Information Fusion for Natural and Man-Made Disasters

AFOSR Grant F49620-01-1-0371

Final Report

January 31, 2007

Submitted to:
U.S. Air Force Office of Science and Research
875 N. Randolph St. Room 3112
Arlington, VA 22203-1954

Principal Investigator:
Dr. Peter D. Scott
Department of Computer Science and Engineering
University at Buffalo
Buffalo NY 14260-2000

Approved for public release; distribution is unlimited

REPORT DOCUMENTATION PAGE

1. REPORT DATE *(DD-MM-YYYY)*	2. REPORT TYPE	3. DATES COVERED *(From - To)*
31-01-2007	Final Technical Report	4/2001 to 10/2006

4. TITLE AND SUBTITLE

Information Fusion for Natural and Man-Made Disasters

5a. CONTRACT NUMBER

N/A

5b. GRANT NUMBER

F49620-01-1-0371

5c. PROGRAM ELEMENT NUMBER

N/A

6. AUTHOR(S)

Scott, Peter D. PhD.

5d. PROJECT NUMBER

6609

5e. TASK NUMBER

N/A

5f. WORK UNIT NUMBER

N/A

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)

CUBRC
4455 Genesee St.
Buffalo, NY 14225

8. PERFORMING ORGANIZATION REPORT NUMBER

6609-01

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)

U.S. Air Force Office of Science and Research
875 N. Randolph St. Room 3112
Arlington, VA 22203-1954

10. SPONSOR/MONITOR'S ACRONYM(S)

AFOSR

11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION/AVAILABILITY STATEMENT

13. SUPPLEMENTARY NOTES

14. ABSTRACT

Prompt effective response to emergencies created by major natural and man-made disasters requires highly coordinated efforts from emergency responders and decision-makers across multiple disciplines, jurisdictions, and hierarchical levels of responsibility and authority. A critical task is creation of coherent, comprehensive and accurate situation assessment which can guide decision-making and resource allocation. The raw materials for this situation assessment are prior domain knowledge, incoming reports from sensors and human observers, a situational assessment logic, and a disciplined data fusion paradigm. To date, data fusion research in this domain has focused on the levels of data fusion below the critical situation assessment level, namely signal and object assessment. This project studies the use of comprehensive data fusion, including situation and impact assessment, in the response to natural and man-made disasters. In particular, the early response phase is emphasized, in which casualty mitigation is the core goal. New methodological approaches and their deployment in an earthquake simulator test bed are the products of this research.

15. SUBJECT TERMS

Information Fusion, Emergency Management, Disaster, Situation Assessment, Models, Optimization, Casualty Management

16. SECURITY CLASSIFICATION OF:

a. REPORT	b. ABSTRACT	c. THIS PAGE
UNCLAS	UNCLAS	UNCLAS

17. LIMITATION OF ABSTRACT

UU

18. NUMBER OF PAGES

19a. NAME OF RESPONSIBLE PERSON

Mr. Michael D. Moskal

19b. TELEPHONE NUMBER *(Include area code)*

716-631-6923

Acknowlegments

I gratefully acknowledge the efforts of the team of faculty, consultants and graduate students from CUBRC, the University at Buffalo, the University of Virginia and beyond who worked on this project. Over its five year lifetime, these scholars gave unstintingly of their time and expertise to support a common vision of scientific excellence and technical relevance. Whatever achievements may be noted in this work are theirs, wherever we came up short I take responsibility. I am particularly indebted to Dr. James Llinas, the original Principal Investigator for the project, who gave me the opportunity to carry on this important work. I also acknowledge the support given us by our sponsor AFRL, and the guidance of our program director Dr. John Tangney.

Many sections of this final report were distilled from reports and publications authored by project team members cited in Section 4 Personnel Supported by the Grant, and I thank them for the opportunity to use their work here. All should be considered co-authors of this report.

Peter D. Scott   January 31, 2007

AFOSR F49620-01-1-0371

1. Table of Contents

2. Program Goals and Achievements

Here an executive summary of the principal goals, objectives, achievements and new findings is presented. Extended technical development of each item can be found in Section 6 of this report.

2.1 Goals

The primary goals and program objectives of this project have remained unchanged since their initial articulation in the project proposal presented by Dr. James Llinas to the AFOSR in February 2001 [6.10-20]. The three-fold primary objectives consist in: 1. Development, validation and documentation of a quantitative engineering methodology for L2-L3 fusion; 2. Design and implemention of DIRE (acronym designating our DIsaster Response Environment), a scalable reusable digital simulation test bed for studying higher level fusion the in the context of response to natural and man-made; and 3. Production of work products with transition paths to high priority Air Force problem domains.

Secondary objectives include formal problem encoding and the production of a domain-specific disaster ontology; design of a centralized fusion node and distribution of centralized fusion functionality to a hierarchical distributed fusion network. Additional secondary objectives developed as the project evolved include the incorporation of disaster-response-specific cognitive work analysis taxonomies within the fusion ontology, formulation of architectural specifications, and the discovery and response to secondary heterogeneous disaster events within the primary scenario (such as a toxic chemical spill event within the initial response phase to an earthquake with damage and casualties).

2.2 Achievements

The dual thrusts by which these goals were pursued were: 1. The development of a comprehensive methodological framework whose scope ranged from broad design issues such as fusion architecture and reasoning scheme to implementation issues such as visualization and asssociation algorithms; 2. The development of a test bed in which to explore the properties of the methods described.

2.2.1 New Methodologies

AFOSR F49620-01-1-0371

New frameworks presented in a problem domain as inherently multi-layer and extended as data fusion have many interrelated elements. Here we list only those we consider the most notable. The details of these findings, and many others, are contained in Section 6 of this report.

1. A new belief-based argumentation system logical framework for abductive reasoning in high level fusion. This method is an extension of the Probabilistic Argumentation System of Kohlas and colleagues in which a belief representation of uncertainty is employed.

3. A disaster ontology (DisReO) is defined and delineated.

4. A cognitive work analysis of the distster response scenario is delineated, and employed jointly with a disaster ontology to define a user-centric approach to data fusion system design.

5. A new logistic hospital model is developed which bypasses previous limitations in accurately predicting residual future capacity during an early-phase emergency response.

6. A new dispatch algorithm for transportation resources is developed in which aggregates of targets are employed to partition the service space, leading to more effective dispatch service in the presence of a high degree of uncertainty.

7. A new routing approach and related algorithms for transportation resources in presented. The novelty is the combination of mutliple disparate solutions and routing efficiency, a suitable approach in scenarios in which the fastest routes may prove to be blocked or slowed.

8. A visualization scheme based on the new principle of dynamic iconography is developed for situation awareness in scenarios where low latency is required such as disaster response.

9. Adjudication and track confidence updating are integrated into the data fusion system design, the significance of these mechanisms and the procedure for doing this is detailed.

10. A new three-layer general fusion architecture is presented. This flexible scalable hybrid scheme integrates layers of the Dual Node, Blackboard and Intelligent Agent schemes in a natural way.

2.2.2 The DIRE Test Bed

1. An HLA/RTI-compliant distributed simulation environment for earthquake scenarios has been designed and tested. This system is suited to the testing and evaluation of data fusion schemes, and testing and evaluation of the phenomenological models for constituent objects such as hospitals and emergency personnel behaviors.

2. A secondary incident generator has been implemented within DIRE. The current focus is on a secondary Hazmat incident, but other secondary incidents such as fire or flood could be easily implemented.

3. The principal methodological recommendations of this project have been coded into DIRE.

4. Results from the test bed suggest that high level fusion is critical to realizing significant benefit from incorporating data fusion in the emergency response environment. Use of L0/L1 fusion without L2/L3 fusion may in fact degrade overall emergency response effectiveness compared to no fusion at all.

3. Personnel Supported by the Grant

3.1 Faculty

Dr. Aidong Zhang, Computer Science and Engineering Dept, University at Buffalo

Dr. Ann Bisantz, Industrial Engineering Department, University at Buffalo

Dr. Don Brown, Systems Engineering Department, University of Virginia

Dr. Henry Hexmoor, Computer Science Department, University of Arkansas

Dr. T. Kesavadas, Mechanical Engineering Department, University at Buffalo

Dr. George Lee, MCEER Earthquake Center, University at Buffalo

Dr. Eric Little, Depts. of Education and Philosophy, D'Youville College

Dr. James Llinas, Industrial Engineering Department, University at Buffalo

Dr. Chris Rump, Industrial Engineering Department, Bowling Green University

Dr. Peter Scott, Computer Science and Engineering Dept., University at Buffalo

3.2 Post-Doctoral Fellows

Dr. William Frank, Industrial Engineering Dept., University at Buffalo

Dr. Michael Tong, Industrial Engineering Dept., University at Buffalo

3.3 Consultants

Dr. Chris Bowman, Data Fusion and Neural Networks

Dr. Ron Eguchi, Imcat Inc.

Dr. Galya Rogova, Encompass Consulting

3.4 Professional staff

Justin Stile, Systems Engineering Department, University of Virginia Charlottesville

Mike Moskal, CUBRC

James Scandale, CUBRC, University at Buffalo

Adam Stotz, CUBRC

Dorothy Tao, MCEER, University at Buffalo

3.5 Graduate Students

Rajendra Agrawal, Dept. Mechanical Engineering, University at Buffalo

Santosh Basapur, Dept. Industrial Engineering, University at Buffalo

Jae Young Choi, Industrial Engineering, University at Buffalo

Zhaofan Ding, Dept. Computer Sci. and Eng., University at Buffalo

Rucha Gokhale, Industrial Engineering, University at Buffalo

Santosh George, Dept. Industrial Engineering, University at Buffalo

Rucha Gokhale, Dept. Industrial Engineering, University at Buffalo

Qiang Gong, Industrial Engineering, University at Buffalo

Venkatraghavan Gourishankar, Dept. Mechanical Engineering, University at Buffalo

Arun Jotshi, Dept. Industrial Engineering, University at Buffalo

Ameya Kamerkar, Dept. Mechanical Engineering, University at Buffalo

Jae-Jun Kim. Industrial Engineering, University at Buffalo

Young-Seok Kim, Dept. Mechanical Engineering, University at Buffalo

Feng Lin, Industrial Engineering, University at Buffalo

Carlos Lollett, Dept. Computer Sci. and Eng., University at Buffalo

Naveen Manchikatla, Industrial Engineering, University at Buffalo

Matthew Mandiak, Dept. Mechanical Engineering, University at Buffalo

Natalia Mazaeva, Industrial Engineering, University at Buffalo

Nishant Mishra, Industrial Engineering, University at Buffalo

Rashmi Mudiyanur, Dept. Computer Sci. and Eng., University at Buffalo

Jomon Paul, Industrial Engineering, University at Buffalo

Sanjay Rawat, Industrial Engineering, University at Buffalo

Jay Robertson, Dept. Systems Engineering, University of Virginia

Daniel Robinson, Dept. Systems Engineering, University of Virginia

AFOSR F49620-01-1-0371

Charles Shah, Dept. Mechanical Engineering, University at Buffalo

Pengfei Yi, Dept. Industrial Engineering, University at Buffalo

4. Theses and Dissertations Produced Under the Grant

4.1 MS Theses

4.1.1 Basapur, Santosh Suresh, *The effect of display modalities on decision making under uncertainty*. A thesis presented to the faculty of the Industrial Engineering Department of the University at Buffalo in partial fulfillment of the requirements for the MS Degree. Abstract available by request from the University at Buffalo Libaries, Capen Libray Thesis Collection.

4.1.2 George, Santhosh K., *Diffusion based FEM simulation and free-form surface characterization for sequential MEMS fabrication processes*. A thesis presented to the faculty of the Industrial Engineering Department of the University at Buffalo in partial fulfillment of the requirements for the MS Degree. Abstract available by request from the University at Buffalo Libaries, Capen Libray Thesis Collection.

4.1.3 Kamerkar, Ameya V., *Touch based interactive nurbs modeler using a force/position input glove.* A thesis presented to the faculty of the Mechanical and Aerospace Engineering Department of the University at Buffalo in partial fulfillment of the requirements for the MS Degree. Abstract available by request from the University at Buffalo Libaries, Capen Libray Thesis Collection.

4.1.4 Lollett, Carlos, Sensor *fusion for mobile robots,* A project presented to the faculty of the Mechanical and Aerospace Engineering Department of the University at Buffalo in partial fulfillment of the requirements for the MS Degree.

Abstract: Mobile robot is a concept that is usually intuitive understood. A mobile robot should be able to interact autonomously with its environment. It means that a mobile robot should be aware of its environment. Robots can know about its environment through devices called sensors.

A sensor is a device that transforms an energy that come from the environment to a useful form that provide environment information, like the case where an infrared sensor detect infrared radiation coming from the environment using it to measure distance.

Human being as a system is also capable to get information from his environment through his senses. We get information from our environment seeing, hearing, touching, smelling and tasting. Sometimes, the same kind of information can be derived from two or more sense. If it is possible to combine the information from several sensors in a synergic way, it is also possible to enhance the environment understanding. That process is called sensor fusion.

In the field of robotics, sensor fusion is a technique for interpreting data from disparate robot sensors to form a unified "picture" of what is happening in the robot's world [2].

In this project a simple setup of three ultrasound sensor was used to implement a robot navigation algorithm. Ultrasound sensors were used to measure distance to obstacles. Having different spatial orientation, the information from several sensors can be use to estimate the angle of regular walls.

In order to obtain the orientation of a straight line, at least two points P1(x1,y1) and P2(x2,y2) as shown in Fig. 1. The robot faces the line with an angle alpha.
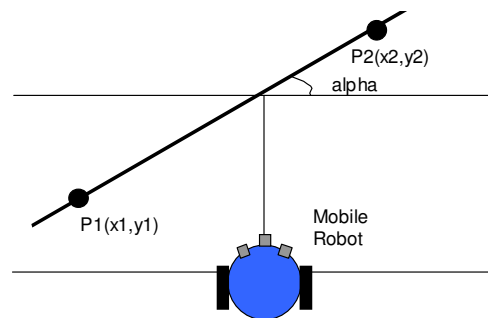
Figure 1. Mobile Robot faces an obstacle.

The alignment between the robot and the wall can be used in the decision making process. In order to avoid the obstacle the robot should try to align its path parallel to the wall. Once the obstacle is cleared it can resume its original goal.

Two approaches to the problem were used: Wall Orientation and Progressive Clearing. In Wall Orientation, the robot goes as close as it can to the wall and then aligns itself to a parallel direction to the wall axis. In Progressive Clearing, the robot tends to stop earlier and then check for the sensor that shows the clearest path, rotating accordingly.

4.1.5 Mandiak, Matthew *Haptics enabled virtual assembly application for enhanced product design*, A thesis presented to the faculty of the Mechanical and Aerospace Engineering Department of the University at Buffalo in partial fulfillment of the requirements for the MS Degree.

Abstract: The use of virtual environments offers en dless possibilities to an engineer in a manufacturing setting. This thesis deals w ith the development of a virtual assembly package to aid in the product development cycle. A framew ork for a user interface is described which will allow e ngineers to design, manufactur e and assemble in a virtual environment. Therefore, prototypes can th en be created virtua lly without expending significant resources or money. In this wo rk, a haptic interface was implemented to allow for assembly to take place with fo rce feedback. In ad dition, manufacturing statistics were provided through a custom bu ilt interface to guide a user on how well the parts they were assembling were produced. Experiments were then carried out in order to prove that haptics could be used to distinguish assemb ly amongst parts of varying assembly parameters. These experiments th en serve to validate the use of virtual assembly in product design.

4.1.6. Mishra, Nishant, *Capacity and non-steady state generalizations to the dynamic MEXCLP model for distributed sensing networks*, A thesis presented to the faculty of the Mechanical and Aerospace Engineering Department of the University at Buffalo in partial fulfillment of the requirements for the MS Degree. Abstract available by request from the University at Buffalo Libaries, Capen Libray Thesis Collection.

4.1.7 Rawat, Sanjay *A frame work for performance evaluation of multi target tracking systems*, A thesis presented to the faculty of the Industrial Engineering Department of the University at Buffalo in partial fulfillment of the requirements for the MS Degree. Abstract available by request from the University at Buffalo Libaries, Capen Libray Thesis Collection.

4.1.8 Robinson, Daniel *Assessing Casualty Densities Based on Sensor Reports Pursuant to a Large-Scale Disaster*, A project presented to the faculty of the Systems Engineering Department of the University of Virginia Charlottesville in partial fulfillment of the requirements for the MS Degree.

Abstract: One of the newest innovations which is making its way more prevalently into the field of emergency response is information technology. Information technology (IT), in this sense, seeks to turn relevant data into usable information to aid in an emergency response. One of the key elements to useful beneficial IT is to quickly, accurately, and dynamically turn incoming data into usable information.

This project presents a way to statistically analyze incoming casualty reports at specific time intervals to not only estimate casualty densities, but also assess whether or not the casualty densities being observed are within some confidence interval of an expected number of casualties. Simple models of the searching process are developed and used to dynamically analyze an incoming report stream. If the number of casualties is sufficiently different than the expected number, then one might conclude either a secondary event has occurred or the initial estimates were simply wrong.

To test the method a simulation is developed where the region in question will be a 10X10 grid. The total casualty population in the model will be 30,000 and the total number of sensors searching for casualties will be 500. The simulation will go for 10 time steps representing 10 hours of searching. Additionally, for all testing the 80% confidence level will be used to determine error rates.

The largest number of casualties in any one grid location is 764. This is not a fixed parameter but rather a result of the random layout. To assign a number of man hours to search in each region at each time step, the total number of sensors was randomly distributed in each region according

to a uniform distribution. Thus, for the purposes of testing, there is no inherent method to assigning search patterns.

The probability of finding a casualty is a function of the damage in an area. Damage was assigned randomly in the same fashion as the casualty layout with the sum of all damage being 30,000. A function for assigning the probability of finding a casualty is

$$p_{i,t} = \exp\left\{-\left(\frac{D_i}{10\max(Di)}\right)\left(\frac{\sum_{i=1}^{100} P_{i,t}}{P_{i,t}}\right)\right\}$$

This function is simply a way to map the damage and man hour values into an appropriate probability space.

To test the simulation it was run 1000 times on a casualty distribution which was exactly what is expected. A hypothesis test result of not being from the expected distribution will be referred to as an area of interest. A contour plot of the hypothesis test results, or rather the percentage of the time that the simulation classified the search region as an area of interest, is given in figure 2. Figure 1 represents the actual distribution of casualties.

4.2 Ph.D. Dissertations

4.2.1. Choi, Jae Young, *Stochastic scheduling problems for minimizing tardy jobs with application to emergency vehicle dispatching on unreliable road networks*, A dissertation presented to the faculty of the Industrial Engineering Department of the University at Buffalo in partial fulfillment of the requirements for the Ph.D. degree. Abstract available by request from the University at Buffalo Libaries, Capen Libray Thesis Collection.

4.2.2 Gong, Qiang, *Responding to casualties in a disaster relief operation: Initial ambulance allocation and reallocation, and switching of casualty priorities*, A dissertation presented to the faculty of the Industrial Engineering Department of the University at Buffalo in partial fulfillment of the requirements for the Ph.D. degree.

Abstract: This research is concerned with models for response to casualties in a disaster relief operation. Three problems are analyzed. The first is that of initial ambulance allocation to

casualty clusters. The second is that of ambulance reallocation between casualty clusters. The third is that of switching casualty priorities. We briefly describe each contribution.

The first problem analyzes a deterministic ambulance allocation model for a post-disaster relief operation. Casualties in a natural disaster, e.g., earthquake, tend to be numerous and distributed in space, typically forming clusters. Due to the geographic separation of the clusters it is not practical to switch ambulances between clusters frequently after the rescue starts. Thus it is critical to allocate the correct number of ambulances to each cluster at the beginning of the rescue process. We formulate a deterministic model which depicts how a cluster grows after a disaster strikes. Based on the model and given a number of ambulances, we develop methods to calculate critical time measures, e.g. completion time for each cluster. Then we present two iterative procedures to optimize the makespan and the weighted total flow time, respectively. Our methods are illustrated via a case study, which is based on an earthquake in Northridge, California. The main conclusion is that the optimal ambulance allocation can be significantly dependent upon the desired performance measure.

The second problem analyzes the ambulance reallocation problem on the basis of a discrete time policy. The benefits of redistribution include providing service to new clusters and fully utilizing ambulances. We consider the objective of minimizing makespan. The complication is that the distance between clusters needs to be factored in when making an ambulance reallocation decision. Our model permits consideration of travel distance between clusters.

The third problem is concerned with servicing casualties with different priorities. We formulate a two-priority, preemptive, single-server queueing model. Each customer is classified into either a high priority class or a low priority class. The arrivals of the two priority classes follow independent Poisson processes and service time is assumed to be exponentially distributed. A queue-length-cutoff method is considered. Under this discipline the server responds only to high priority customers until the queue length of the other class exceeds a threshold L . After that the server switches to handle only the low priority queue. Steady-state balance equations are established for this system. Then we introduce two-dimensional generating functions to obtain the average number of customers for each priority class. We then focus on the preemptive resume case. We develop methodologies to obtain the optimal cutoffs for the situation when the

weights of both queues are constant (i.e., not a function of queue length) and the situation when the weights change linearly with the queue lengths.

4.2.3 Jotshi, Arun, Search *for Immobile Entities on a Network,* A dissertation presented to the faculty of the Industrial Engineering Department of the University at Buffalo in partial fulfillment of the requirements for the Ph.D. degree.

Abstract: We consider the problem of searching for immobile friendly entities on an undirected network. The time to search the entity is a random variable, whose probability density function (pdf) depends upon the path and also upon any information we have regarding the location of the entities on the network. The objective function that we consider is expected search time. We seek a path choice that minimizes this objective. A specific application is a disaster scenario (natural or man-made), in which the searcher is an ambulance and the entity is a casualty. Solving the search game on a network with an arbitrary starting point for the searcher is an interesting problem which, to our knowledge, has not yet been investigated. Minimizing the expected search time differs from arc-covering problems, e.g. the Chinese Postman Problem (CPP), in the way that here the objective is not to find the minimum length tour that covers all the links at least once, but instead to minimize the expected time to find the entity. This problem is also different from search problems considered in the literature, since the entity is neither an evader nor a cooperator and there is no information regarding the location of the entity except for a region of interest within which the entity is believed to be found. We plan to address several aspects of this particular class of search problems. In our analysis we assume that the entity is of infinitesimal size, i.e., it is only found when the searcher is directly over it. The number of entities is given by a Space Poisson random variable. The number of entities we are trying to find is driven by the capacity of the searcher, which in our case is the ambulance.

The dissertation is divided into two parts. Part 1 is dedicated to the problem of optimal search for the first entity. Part 1 assumes that the searcher capacity is one. We introduce a heuristic algorithm to deal with the search process given that there is exactly one entity on the network and it is equally likely to be at any point on the network. We later prove that this path is also optimal for the situation when a number of entities are present and these entities are uniformly distributed. We also show that the search process given non-uniform distribution of entities

across the network is a special case of the uniformly distributed entities. Finally, the case with re-optimization is briefly considered.

4.2.4 . Kim, Young-Seok, *Fingertip digitizer A real-time, fingertip-mounted haptic sensing system for active, dynamic, and viscoelastic touch*. A dissertation presented to the faculty of the Mechanical and Aerospace Engineering Department of the University at Buffalo in partial fulfillment of the requirements for the Ph.D. degree.

Abstract: The capability of human beings to perform skilled tasks often depends on their ability to touch, grasp, and manipulate tools and control objects. An example where this is especially true is in the 3D digitizing industry, where a stiff probe has to make contact with the object under study. However, this tool-based interface has a major drawback---the outcome excludes the benefits of the unique feeling that comes from the direct finger contact on an object. This dissertation introduces a new approach to finger touch interface. Based on the sensing methodology for dynamic human touch, called Active Touch paradigm, a sensory-enhanced virtual environment is proposed where both man and machine perfectly share the haptic stimuli. With this interface, overall work performance can be enhanced by the machine's digital power and the human's instinctive exploratory capability. In the present work, this concept is implemented through the invention and validation of a new dynamic fingertip digitizing device called the Fingertip Digitizer . The unique approach presented here adds a new perspective to the science of conventional passive human touch by adding active and dynamic aspects to it. The specific aims of the present work are as follows: (1) develop a fingertipmounted hardware capable of tactile digitizing, (2) investigate dynamic features of fingertip characteristics in tactual tasks, and (3) develop touch-based applications of multimodal sensory feedback using the new Fingertip Digitizer. All of the above aims were successfully completed. First, a fingertip-mounted digitizer, capable of capturing both static and dynamic phenomena at the finger tip, was developed. Second, the fingertip behavior during the dynamic tactual activities was investigated with the consideration of the fingertip's dynamic and viscoelastic behavior during active touch. Finally, three applications of the Fingertip Digitizer were developed: (1) Touch Painter & Touch Canvas : a 2D touch interface for intuitive drawing, (2) Tactile Tracer : a 3D touch interface for object digitizing, and (3) Touch Model Verifier : a verification methodology for comparing the

haptic stimuli from the real physical objects and its corresponding virtual object through a haptic device.

4.2.5. Kim, Jae-Jun, *Design of hardware/algorithm for enhancement of driver/vehicle performance using a virtual environment*. A dissertation presented to the faculty of the Mechanical and Aerospace Engineering Department of the University at Buffalo in partial fulfillment of the requirements for the Ph.D. degree. Abstract available by request from the University at Buffalo Libaries, Capen Libray Thesis Collection.

4.2.6 Paul, Jomon, *Study of effects of facility damage on hospital capacity estimates and location-allocation planning for management of natural disasters*, A dissertation presented to the faculty of the Industrial Engineering Department of the University at Buffalo in partial fulfillment of the requirements for the Ph.D. degree.

Abstract: Estimation of the impact of damage to the hospitals due to a natural disaster is very important since it allows for planning prior to and shortly after the disaster strikes. These estimates could also be used to plan new facilities and capacity reallocation between existing facilities. The dissertation is divided into three parts.

In the first part, the impact from facility damage due to a natural disaster like an earthquake or hurricane to the hospital capacity estimates is estimated. A recent paper by Yi et al. (2005) contains a generic hospital simulation model. This model is extended to incorporate a facility damage component and estimate the corresponding effect on patient waiting times and capacity estimates. A hospital, unlike many other service organizations, is more affected by non structural damage than structural damage. Non structural components in the hospital like power, water and medical resources, if damaged, can render the hospital useless.

In the second part the effect of capacity reductions on planning of the hospital facility location and capacity allocation in a region prone to natural disaster is incorporated. Two basic models are developed and analyzed for hospital location and capacity allocation. The focus is on an area prone to natural disasters. The first model seeks to locate hospitals and allocate capacities so that the mean travel distance for patients to hospitals is minimized over a variety of disaster

scenarios. The second model seeks to reallocate capacity among hospitals so as to maximize the system's effectiveness to the forthcoming disaster event.

In the third part the effect of damage to the transportation network on the hospital location and capacity allocation problem is studied. Various scenarios of road damage are simulated for the earthquake and hurricane disaster. The results are demonstrated via examples and case studies. Once a disaster strikes, people tend to move out of their current location so as to reach a better and safer location. This displaced population is mainly the noninjured and low severity people. The Roads become congested leading to increased travel times. This directly affects the disaster relief meted out to the casualties. This factor is incorporated in the capacity reallocation model via a simulation model.

4.2.7 Yi, Pengfei, *Real-time Generic Hospital Capacity Estimation under Emergency Situations*
A dissertation presented to the faculty of the Industrial Engineering Department of the University at Buffalo in partial fulfillment of the requirements for the Ph.D. degree.

Abstract: Hospitals are an integral part of a society's readiness to respond to man-made and natural disasters. Capacity planning greatly enhances the capability and effectiveness of treatment provided to the injured resulted from a disaster. The real-time capacity estimates for the hospitals presented in the disaster region can be used for patient/ambulance routing, resource planning and emergency operations management. Clearly case-specific models based on average or steady-state conditions are insufficient in such dynamic environment. Hence, a methodology to handle the generic, real-time, and dynamic phenomena has been developed to provide accurate capacity estimation.

This research has addressed three major requirements that are not mentioned in previous research. First, the methodology needs to be generic so that it can represent a large range of hospitals with various sizes and capabilities. Second, in addition to long-term performance, the dynamic nature of both hospital operations and patient arrivals in a disaster needs to be captured. Third, the capacity estimation has to be made accurately in real time to ensure its usefulness for disaster relief efforts.

All of the above issues are critical for rescue management, however, none has been addressed before. To meet this challenge, several steps are taken. First of all, a generic parametric simulation model is developed to take into account the hospital resources, capability, operational efficiency, and types of injuries. The model is capable of representing a variety of hospitals by their characteristics. Then factorial simulation experiments are designed to cover a large range of hospitals. To ensure real-time applications, the simulations are executed off-line and the steady-state performances are regressed into a parametric response surface model by using both linear and non-linear regression.

Based on steady-state regression models, a double exponential parametric metamodel is developed to capture hospitals' dynamic performance during the transient period. This metamodel is further improved to a continuous metamodel that is capable of utilizing the continuous patient arrival rate function. As a reinforcement of the metamodel, a sequential estimation methodology is developed to estimate the dynamic patient arrival rate in real-time.

Finally, the capacity estimation methodology is illustrated in an earthquake setting. Results show viability of the approach and demonstrate promising potential for further analysis of hospitals' dynamic behavior under other emergency situations. More importantly, the developed methodology can be easily applied to other industries such as manufacturing and service.

5. Publications Produced Under the Grant

1.   Rajan Batta, Qiang Gong and Arun Jotshi, "Dispatching and Routing of Emergency Vehicles on Large Scale Networks," Proceedings of the Industrial Engineering Research Conference, Atlanta, GA, 2005.

2.   Ann Bisantz, Galina Rogova and Eric Little, "On the Integration of Cognitive Work Analysis within a Multisource Information Fusion Development Methodology," Proceedings of the 48th Human Factors and Ergonomics Society Annual Meeting, New Orleans, LA, 2004.

3.   C.L. Bowman, The Dual Node Network (DNN) Data Fusion & Resource Management (DF&RM) Architecture, Proceedings of the AIAA 1st Intelligent Systems Technical Conference, Chicago IL, September 20-22, 2004,.

4.   C.L. Bowman, Unifying Data Fusion & Resource Management (DF&RM) Software Development Approaches Using the Dual Node Network (DNN) Architecture, in Unification of Fusion Theories,.

5.   J-Y. Choi, and C.M. Rump, Maximizing the Number of 'Early Enough' Jobs: A Chance-Constrained Stochastic Scheduling Problem. UB Dept. IE Technical Report 2002.

6.   J-Y. Choi, C.M. Rump and G. Rogova, A Dynamic Program for Minimizing the Expected Number of Tardy Jobs with Distinct Exponential Due Dates. Under review by IIE Transactions on Scheduling and Logistics

7.   Q. Gong and R. Batta, "A Queue-Length Cutoff Model for a Preemptive Two Priority M/M/1 System". Submitted to Operations Research.

8.   Q. Gong and R. Batta, "A Allocation of Ambulances to Casualty Clusters in a Disaster Relief Operation". IIE Transactions.

9.   Q. Gong and R. Batta, "A M/M/1 Queue-Length Cutoff Model with Two Priorities," Institute for Operations Research and the Management Sciences Annual Meeting, Denver CO, 2004.

10.   Q. Gong, A. Jotshi and R. Batta, "Dispatch-Routing of emergency vehicles in a disaster environment using data fusion concepts," Proceedings of the 7th International Conference on Multisource Information Fusion, Stockholm, Sweden, 2004.

11.   Joshi, N. Mishra, R. Batta, R. Nagi, Ad Hoc Sensor Network Topology Design for Distributed Fusion: A Mathematical Programming Approach, Proceedings of the 7th International Conference on Information Fusion (ISIF 2004), Stockholm Sweden, June 2004, pp. 836-841.

12.   Jotshi, Q. Gong and R. Batta, "Dispatching and routing of emergency vehicles in disaster mitigation using data fusion," Socio Economic Planning Sciences Journal, accepted for publication, to appear 2006

13.   Jotshi and R. Batta, "Optimum Search for a Target on a Network," Proceedings of the Industrial Engineering Research Conference, Atlanta, GA, 2005, to appear.

14.   Jotshi and R. Batta, "Finding robust paths for routing ambulances in a dynamic disaster environment," Proceedings of the Industrial Engineering Research Conference, New Orleans, LA, 2004.

15.   T. Kesavadas, Y. Kim, Automated Dynamic Symbology for Level 2 and 3 Fusion, Proceedings of the Conference on Visualization and the Common Operational Picture (VizCOP), September 14-17 2004, Canadian Forces College, Toronto, Canada to appear.

16.   Y. Kim, T. Kesavadas, Automated Dynamic Symbology for Visualization of High Level Fusion, Proceedings of the 7th International Conference on Information Fusion (ISIF 2004), Stockholm Sweden, June 2004, pp. 944-950.

17.   E. Little, G. Rogova, Formal ontology and Higher Level Fusion, submitted to Information fusion, Elsevier.

18.   E. Little, A Proposed Methodology for The Development of Application-Based Formal Ontologies, Proceedings of the 2003 Hamburg Symposium on Ontologies

19.   Eric Little. "A Proposed Methodology for Application-Based Formal Ontologies," Proceedings of the Workshop on Reference Ontologies vs. Application Ontologies, 15-18 Sept., University of Hamburg, CEUR-WS, 2004.

20.   Eric Little,Galina Rogova, A. Boury-Brisset, "Theoretical Foundations of Threat Ontology (ThrO) for Data Fusion Applications", TR-2005 -269, 2005.

21.   Eric Little, Galina Rogova. "Ontology Meta-Model for Building A Situational Picture of Catastrophic Events," in Proceedings of the 8th International Conference on Multisource Information Fusion, Philadelphia, PA, 2005.

22.   Eric Little and Galina Rogova, "Ontology Meta-Model for Building A Situational Picture of Catastrophic   Events," Proceedings of the 8th International Conference on Information Fusion, Philadelphia, PA, 2005, to appear.

23.   J. Llinas, C.L. Bowman, G. Rogova, A. Steinberg, E. Waltz, F. White, Revisions and Extensions to the JDL Data Fusion Model, Proceedings of the 7th International Conference on Information Fusion (ISIF 2004), Stockholm Sweden, June 2004, pp. 1218-1230.

24.   Q. Lu, P. Scott, Active model-based object recognition employing foveal imagery and multiresolution feature sets, Proceedings of the 2001 IEEE Conference on Artificial Neural Networks in Engineering, November 2001, St. Louis MO. pp. 608-613.

25.   Y. Kim and T. Kesevadas, "Automated dynamic symbology for visualization of high level fusion," Proceedings of the 7th International Conference on Multisource Information Fusion, Stockholm, Sweden, 2004.

26.   J. Llinas, "Information Fusion for Natural and Man-Made Disasters," Proc. 5th International Conference on Information Fusion, Annapolis, MD, USA, 2002.

27.   J. Llinas, E. Hansen, Updates, Issues and Questions, Third Workshop on Critical Issues in Information Fusion, Java Center N.Y., September 29- October 1 2004

28.   Qiang Lu and Peter Scott, "3-D View-based active object recognition employing foveal imagery," International Journal of Intelligent Systems, invited paper, 2005, to appear.

29.   Q. Lu, P. Scott, Active model-based object recognition employing foveal imagery and multiresolution feature sets, Proceedings of the 2001 IEEE Conference on Artificial Neural Networks in Engineering, November 2001, St. Louis MO. pp. 608-613.

30.   M. Mandiak, P.P. Shah, Y. Kim and T. Kesavadas, "Development of an integrated GUI framework for post-disaster data fusion visualization," Proceedings of the 8th International Conference on Information Fusion, Philadelphia, PA, 2005, to appear.

31.   J. Paul, P. Yi, S. George,., and L. Lin, "Transient modeling in simulation of hospital operations for emergency response," Prehospital and Disaster Medicine, 21(4) 223-236, 2006.

32.   J. Robertson, D, Brown, Developing a Modular Simulation for the Assessment of Response to Manmade and Natural Disasters, Proceedings of the 22nd Annual Systems Dynamics Conference, Keble College, Oxford, England , July 25 - 29, 2004.

33.   G. Rogova, Reliability in Information Fusion: Literature Survey, Proceedings of the 7th International Conference on Information Fusion (ISIF 2004), Stockholm Sweden, June 2004.

34.   Galina Rogova, Peter Scott, Carlos Lollett, Rashmi Mudiyanur, "Reasoning about situations in the early post-disaster response environment," Proceedings of the 9th International Conference on Information Fusion, Florence, Italy, 2006.

35.   Galina Rogova, Peter Scott and Carlos Lollett, "High level fusion for post-disaster casualty mitigations operations," Proceedings of the 8th International Conference on Information Fusion, Philadelphia, PA, 2005.

36.   Galina Rogova, Peter Scott and Carlos Lollett, "Distributed Fusion: Learning in multi-agent systems for time critical decision making," in E. Shabazian, G. Rogova, and P. Valen, Data Fusion for Situation Monitoring, Incident Detection, Alert and Response Management, FOI Press, 2005, pp.123-152.

37.   Galina Rogova, "Higher Level Fusion: Issues and Design Approaches, in: Data Fusion Technologies for Harbor Protection, E. Shahbazian, M. DeWeert, G. Rogova, (eds), Springer Verlag Publishers, Berlin

38.   Peter Scott and Galina Rogova, "Data fusion framework for early-phase disaster response," invited paper, Proceedings of the National Academy of Sciences Workshop on Using Information Technology to Enhance Disaster Management, 2005, Washington, DC.

39.   P. Scott and G. Rogova, "Crisis Management in a Data Fusion Synthetic Task Environment," in: Proceedings of the 7th Annual Conference on Multisource Information Fusion, Stockholm, Sweden, 2004.

40.   Peter Scott and Galina Rogova, "Data fusion framework for early-phase disaster response," invited paper, Presented at the National Academy of Sciences Workshop on Using Information Technology to Enhance Disaster Management, 2005, Washington, DC.

41.   P.P. Shah, M. Mandiak, Y.S. Kim and T. Kesavadas "Runtime simulation for post-disaster data fusion simulation," Submitted to The Simulation Journal.

42.   G. Srimathveeravalli, N. Subramanian and T. Kesvadas, "A scenario generation tool for the DDF simulation   testbeds," Proceedings of the Winter Simulation Conference, Washington DC, 2004.

43.   Steinberg, C.L. Bowman, "Rethinking the JDL Data Fusion Levels", Proceedings of the National Symposium on Sensor and Data Fusion, Johns Hopkins Applied Physics Lab, June 2004, to appear.

44.   P. Yi, . George, J. Paul, and L. Lin, "Hospital Capacity Planning for Emergency Management in Disaster Mitigation," Socio Economic Planning Sciences, accepted for publication, to appear 2006..

45.   Yi, P., George, S. and Lin, L. 2004: "Real-time Hospital Capacity Estimation by Off-line Simulation and Metamodeling," Proceedings of the 2004 Industrial Engineering Research Conference (IERC), May 15-17, 2004, Houston, TX.

46.   Yi, P., George, S. and Lin, L. 2005: Sequential kernel estimation on dynamic patient arrival rates at hospitals after an earthquake," under review by Computers and Industrial Engineering.

6. Technical Summary of Results

6.1 Overview

Prompt effective response to emergencies created by major natural and man-made disasters requires highly coordinated efforts from emergency responders and decision-makers across multiple disciplines, jurisdictions, hierarchical levels of responsibility and authority. A critical task is creation of coherent, comprehensive and accurate situation assessment which can guide decision-making and resource allocation. The raw materials for constructing this situation assessment are prior domain knowledge, incoming reports from sensors and human observers, a situational assessment logic, and a disciplined data fusion paradigm. To date, data fusion research in this domain has focused on the levels of data fusion below the critical situation

assessment level, namely signal and object assessment. This project is concerned with the integration of comprehensive data fusion, including situation and impact assessment, in the response to natural and man-made disasters. In particular, the early response phase is emphasized, in which casualty mitigation is the central goal. New methodological approaches and their deployment in an earthquake simulator test bed called DIRE (DIsaster Relief Environment) are the products of this research. Here we briefly highlight some of the principal themes in the work to be be discussed in more detail in the following subsections.

Fusion of information from disparate sources and sensors can only proceed effectively witin the framework of a *lingua franca*, a common ontology with which to calibrate meaning and value.This report includes results of research into a metaphysically-based ontology for improved understanding of post-earthquake disaster environments, with extended applications to other kinds of urban disaster environments containing significant numbers of casualties (e.g., terrorist attacks and conventional or unconventional urban warfare activities). Significant attention has been paid to designing the ontology's uppermost levels as well as domain-specific (i.e., lower) levels in order to produce the framework for an overarching model of disaster environments, which can positively impact on the functions of decision-makers who are observing and managing those environments. In particular, our attention has been focused on methods for using ontologies to detect and model the dynamic properties of casualty clusters and their relations to other items in the environment such as the earthquake event itself, hospital and ambulatory services, building, road and bridge damage, and tertiary disaster events. Given the daunting complexity of earthquake disaster environments, it was necessary to focus our methodologies on items such as these, in order to provide a manageable problem space within which to work.

User-centric data fusion system design requires a common ontology, but not an arbitrarily extended one, rather one focused on the key domain processes to be served such as rescue operations. WIthout embedding the ontology into a work domain framework to formulate scope and constraint, the essential elements of information those processes require will not be exposed and the fusion system left without deep roots in user needs. This research explored the means by which methods in cognitive engineering, namely, work domain analyses, could provide input to the development of advanced information processing, or multisensor information fusion,

algorithms. Specifically, a work domain analysis of an emergency management environment (in a post-earthquake context) was performed, and linked abstraction hierarchy models representing the emergency management and response system, the physical environment (e.g., buildings, transportation systems, civilians), and other goal directed agents (e.g., civilian responders and volunteers) were created. Outputs from that analysis (information requirements) were input to the design of the information processing algorithms, providing guidance as to the nature of information required by decision makers, which could be computed through fusion capabilities. This ongoing work thus presents an example of an integrated cognitive engineering/multisensor fusion methodology. One focus within cognitive systems engineering is the systematic description of aspects of the work domain comprising the environment in which human operators must act and make decisions. Specifically, models and techniques in work domain analysis have been developed which capture the complexities and constraints of the work domain that serve to shape and constrain the behavior of domain practitioners. An important output from such an analysis is the provision of information requirements for system controllers and decision makers.

Hospitals are an integral part of a society's critical functions to respond to man-made and natural disasters. Effective hospital capacity planning can significantly enhance the capability and effectiveness of treatment for emergency patients with injuries resulting from a disaster. This information can be used for patient/ambulance routing, resource planning, and emergency operations management. Here we develop a generic simulation model that is capable of representing the operations of a wide range of hospitals in an earthquake disaster situation. From results of our simulations, generalized regression equations are fitted to obtain steady-state hospital capacities. A parametric metamodel is then developed to predict transient capacity for multiple hospitals in the disaster area in a timely manner, as demanded by emergency operations management.

The Dispatcher-Router is a simulation model of the two functions of dispatching ambulances ─ picking up casualties and the subsidiary one of calculating the best route (to either a casualty or a medical treatment center). The dispatcher is also a decision point in the simulation where an improved estimate of casualty location and severity, derived from the information fusion module(s) is injected back into the simulation. Thus the simulation can be run either with or without the aid of fusion, providing one rough measure of the effect of the availability of fused

estimates. The only function of the Router is to provide the quickest route from a source location to a destination. This calculation must take into account the effects of the disaster (such as damaged transportation infrastructure or geographic areas which must be avoided due to chemical or biological hazard). Here we present a new algorithms for the dispatch of ambulances to clusters of casualties determined hy high-level fusion, and their routing by an novel approach in which the possibility of severe roadway damage and congestion dictates the need for multiple alternative routes as distinct from one another as possible within the constraint of short travel time.

The L0/L1 signal and object level data fusion scheme must take into account key properties of the emergency response problem domain: multiple distributed reporting jurisdictions, multiple heirarchical distributed decision-makers, high uncertainty due to the large volume of reports from untrained observers, high error rates due to stressful reporting circumstances, compromised communications systems, damaged and congested transport system. Key elements in the proposed solution are distributed fusion nodes, online track confidence estimation and updating, and a flexible adjudication process which permits the backflow of corrective information in order to maintain consistency of the situation assessment among the decision-makers.

In order to use the data fusion products, decision-makers must be presented with a graphical user interface permitting them to grasp the essential elements of information, rapidly put them in context in a process called sense-making, and candidate various alternative courses of action. Maintaining a large amount of highly dynamic post-disaster fused data is a daunting task, and its visualization is even more difficult to achieve with the paradigm of common geo-referencing systems. In this project we have developed a post-disaster monitoring interface that runs in a fusion-based simulation with High Level Architecture/Run Time Infrastructure (HLA/RTI). In our visualization system, damage and recovering activities are presented in a fast GIS vector map with convenient data and display manipulation. All data that comes from the data fusion federates is displayed at run-time and stored for further analysis. In addition, the pattern of time-aggregated data has enabled dynamic visualization, which includes the morphing of the casualty clusters. This feature provides an effective way to keep track of a region so that a user can easily be aware of the emerging trends. A unique approach to multiple views by the integration of 2D and 3D displays of the fused data is also described.

Finally we turn to L2/L3 fusion, ie. situation and impact assessment. The process of building a situational picture comprises dynamic generation of hypotheses about the states of the environment and assessment of their plausibility via reasoning about situational items, their aggregates at different levels of granularity, relationships between them, and their behavior within a specific context. In some cases, assessment of plausibility of more complex hypotheses may require hierarchical processing, which includes not only reasoning about situational items and relationships between them but also includes relationships between hypotheses and assessments of plausibility of lower level hypotheses. An important component of situation assessment is causal inference aimed at discovery of underlying causes of observed situational items, their attributes and their behavior. Discovery of underlying causes of observed situations is the goal of abductive reasoning or "inference for best explanations". For example, in the early post-earthquake response phase, reasoning about situations is contingent on the assumption that most reported casualties and structural damage are the results of the primary earthquake shock incident and reported subsequent secondary incidents such as fire, flood, aftershocks and Hazmat events. However some secondary incidents such as toxic spills may not be known for a long period of time. At the same time rapid discovery of such incidents is very important since they may have devastating consequences if not responded to quickly. These unknown secondary incidents are usually manifested by unexpected properties and behavior of situational items inconsistent with the current set of beliefs about the state of the world and therefore belief update may be required. Usually belief update methods give priority to this new information and its consequences and abandon some old beliefs to preserve consequences. In the post disaster environment observations and knowledge about situational items, their behavior and relationships are uncertain and, therefore it is necessary to account for this uncertainty while updating the current set of beliefs. In the uncertain environment the principle of priority of new information may not work even in a highly dynamic environment. In the uncertain dynamic environment belief update can be carried out by first seeking some explanations or underlying causes of these inconsistent observations and incorporating these explanations, if found, into a new set of beliefs. Possible explanations can be found as the result of abduction comprising generation of hypotheses about the underlying causes of these inconsistent observations and reasoning about plausibility of such hypotheses. In this project we have developed a method for

belief-based argumentation incorporating these features and applied it to the earthquake simulator DIRE.

6.2 The Disaster Response Environment (DIRE) Test Bed

The Northridge earthquake of January 17 1994 struck the San Fernando Valley at 4:30AM. Classed as a moderate earthquake of magnitude 6.7, this event caused severe casualty and property damage due to the high population density of this area, which is located within the Los Angeles CA city limits. 72 died and over 1,000 were admitted to hospital, with an additional 9,000 treated and released [6.2-9]. Over 12,000 structures were severely damaged, 11 major roadways were closed due to bridge collapses and other structural failures [6.2-10].

This historic event was selected as the basis for the test bed constructed to exercise and test our emergency response data fusion methodology. The disaster domain topology is available in detailed geographic files, the event is well documented, and there is a wealth of data concerning the consequences of the earthquake: casualties and structural damage. Thus the ground truth for our synthetic task environment DIRE is derived from HAZUS using initialization data that reference this Northridge earthquake. Attributes reported as probability distributions by HAZUS are made definite by performing the indicated probability experiments in order to establish a deterministic ground truth.

The overall software architecture chosen for implementation of this environment is the High Level Architecture (HLA) developed by the Defense Modeling and Simulation Office. HLA is a widely adopted standard for distributed heterogeneous simulation in both the military and civilian communities, and its choice is intended to facilitate reusability.

An HLA-compliant simulation system consists of federates, or separate code modules, interacting via a Runtime Infrastructure (RTI) functional interface. The federates in our synthetic task environment include a ground truth generator, report generator, fusion federate, hospital, dispatch/routing, walk-in and visualization federate. Each models actions critical to simulating a mode of activity in the FRP relevant to casualty outcomes. Report generator collects observations and creates reports which are sent to the fusion federate, which implements all data fusion algorithms and publishes the results to subscribing federates. The hospital federate

models the dynamics of hospital medical services, walk-in the dynamics of casualties who choose to seek hospital service without waiting for an ambulance, and dispatch/routing models the assignment of ambulances to casualties and hospitals together with route selection. Visualization delivers visual representations to a human observer or decision maker. Thus for instance ground truth may lay down two casualties at Main and Maple, both of high severity. An observer in the area may incorrectly perceive one severely injured and one lightly injured at that location. A report is sent to fusion, which associates that report with others and perhaps judges there to be two severely injured there. Dispatch/routing then determines how to service these casualties, dispatching an ambulance and directing it to use a specific route (with alternatives) to the casualty and then on to a specific hospital.

6.2.1 Domain knowledge

In order to realistically simulate an earthquake event and nominate a particular data fusion system to operate within it, two categories of domain knowledge are necessary. First, the data, models and consequences corresponding to the ground truth of the earthquake itself are needed. How many casualties are created, of what severities and with what geographic distribution? How many hospitals are damaged and to what degree, how many bridges and gas lines? Second, the assets and operational procedures of, and relationships between, the emergency response organizations must be known. Without the first there is no disaster state to be understood, and without the second there is no awareness possible of what the responders need to understand, no knowledge of what constitutes situation awareness for those users.

Domain knowledge concerning the human and structural damage created by the earthquake is derived from HAZUS, a GIS-based natural hazards estimation tool developed by FEMA. By specifying a geographical region, epicenter and severity corresponding to the Northridge earthquake, we can produce a disaster state resembling that event. The input and lay down distributional parameters can be varied to produce a range of earthquake states based on that real world event.

The phenomena of interest to a data fusion system, which processes need to be understood, what constitutes knowledge, must be defined in terms of the user goals and constraints. The end-user of a data fusion capability in this crisis management setting is the first response command

network. Domain knowledge of the first response system, how it is structured and how it works, is essential to the fusion design process and realistic modeling of the synthetic environment in which it runs.

Surprisingly, the history of organized multi-agency multi-jurisdictional response to natural disasters is rather short. Following a series of devastating wildfires in 1970 Firescope was formed in California to address the lack of coordination, information sharing or communication standards in wildfire-fighting. There followed the Incident Control System ICS and Multi-Agency Control System MACS, further extending common standards for agencies across the state for a range of natural disasters [6.2-6]. In 1994 the state created the Standard Emergency Management System SEMS, organizing all California's emergency offices into a single hierarchical, modular response team [6.2-7]. The organizational structure, operational procedures and goals of SEMS inform our synthetic environment and form the framework for defining higher level data fusion hypotheses.

6.2.2 Objects in the synthetic environment

Rescue and medical management of existing casualties, together with efforts to mitigate risk of additional casualties, are the dominant goals of early stage disaster response. The objects we model in the synthetic environment are those necessary for these activities to unfold, and the modeled attributes of those objects are those instantiating capabilities linked to the casualty-reduction goals and determining their effectiveness. This does not constitute a comprehensive earthquake simulation in the sense of [6.2-4]. Only those objects and attributes most relevant to the FRP goals, and thus most useful in gauging data fusion effectiveness, are represented in the synthetic environment.

Human objects include casualties, police, emergency medical personnel (EMPs) and Hazmat teams. Casualties are attributed by their ID number, physical description, severity and location. Severities are determined from HAZUS data, as are locations (HAZUS reports location at the granularity of census tract, casualties are then randomly placed within the tract as part of our ground truth lay down). Police serve as observers, cruisers moving from initial lay down locations according to a SEMS-based predetermined damage survey plan. EMPs drive ambulances and deliver medical services as they pick up and transport casualties to hospitals.

They also serve as observers. Hazmat teams respond to hazardous chemical spills caused by rupture of a Hazmat transporter vehicle in a roadway accident secondary to the earthquake, or rupture of a Hazmat storage vessel in the damage zone.

Structural objects include hospitals, roadways, bridges and tunnels. Each is attributed by ID number and location. Damage level is associated with hospitals, bridges and tunnels, and link travel times with roadways. Hospital damage, for instance, degrades capacity of that facility. Other structural objects, such as commercial and residential buildings, are not included in the environment. Their damage effects are seen indirectly through the distribution of casualties, which is sufficient for the FRP.

Hazmat objects include ruptured Hazmat roadway transporters and ruptured stationary Hazmat storage tanks. They are attributed by ID number, location, type of hazardous material, and spread of that material.

6.2.3 Reports and Level 1 fusion

Immediately after the primary shock, ground truth is laid down for all objects in the environment. Casualties are characterized and situated, structures tagged by damage level, ambulance and police cruiser initial numbers and locations set.

Initially none of this information is available to the responders. Over logical time reports begin arriving at regional Emergency Operations Centers (EOCs) from observers in the environment: police, EMPs and civilians. Report types include casualty reports and structural damage reports. For instance, an ambulance driver might report a group of casualties at a certain location as the ambulance heads towards hospital with a full load. A civilian may report that a bridge appears to be severely damaged. Note that civilians are not objects in the environment *per se*. They simply serve as the implied sources of certain reports. Locations and times are selected randomly and casualties or structural damage nearby reported by these civilian reporters.

As in the real world, in this environment reports are uncertain and no observers are completely reliable. The report generation process utilizes confused elements of ground truth to model reports. Each element of ground truth being reported on is subject to a confusion matrix before it is enters a report.

The communications links are also assumed unreliable. Associated with each report is probability of reporting failure and probability of reporting delay. These reports are fused to determine the probabilities of the corresponding object-oriented hypotheses. Association is done through ID numbers, physical descriptions and locations. Associated reports are fused using Bayes algorithms. Theses results are then used by the situation assessment module discussed in the next 2 sections.

6.2.4 Design of situation assessment process

The purpose of dynamic situation assessment is to develop probable explanations of the situation based on prior knowledge and incoming transient information. A Situation Assessment (SA) is a stored representation of relations between objects obtained through fusion [6.2-12]. The result of situation assessment is a coherent composite picture of the current situation along with a short prediction of the situation (estimated risk in the case of SA for man-made and natural disasters) to be used by decision makers. In the case of multiple decision makers the situation assessment processes have to deliver a consistent situational picture relevant to each decision maker.

Assessment of the post-disaster situation has specific characteristics, which define requirements for situation assessment architecture and processes. Among these characteristics are:

1. Noisy and uncertain dynamic environment with insufficient *a priori* statistical information

2. Geographically distributed damage

3. Geographically distributed uncertain sources of information often of low reliability .

4. Large amount of heterogeneous information

5. Resource and time constraints

6. High cost of error

7. Multiple decision makers with multiple goals and information requirements

8. Multiple agencies in multiple jurisdictions

These specific domain characteristics call for a multi-agent distributed dynamic situation assessment process, which has to be adaptive to resource and time constraints, new and uncertain environments and reactive to uncertain inputs. This process also has to accommodate heterogeneous information (both symbolic and numeric).

The situation assessment process exploits reports on casualties and damage of essential facilities, databases, maps, information on prior similar situations, preliminary risk assessment based on historical data and event modeling, and results of domain-specific simulations and models (hospital model, walk-in model, etc.) for creating a dynamic situation picture. The produced situation picture provides the critical characteristics of the state in relation to particular goals, capabilities and policies of the decision makers to serve their ultimate goals, which are to serve the maximum number of casualties, save the maximum number of lives , and reduce risk of additional casualties.

There are three essential components of situation assessment process design. The first component is the Cognitive Work Analysis (CWA) [6.2-13], which is a systems-based approach to the analysis, design and evaluation of systems allowing a description of the set of relationships between generic decision tasks, generic activities and available resources. CWA methodology is designed for evaluation of the decision makers' needs to provide understanding of what content various decision makers require from a situation picture and what information should be represented and formulate possible hypotheses about relevant states of the environment.

The second essential component of the situation assessment process is ontological analysis of the specific problem, which denominates the elements of the situation assessment process in terms specific for the disaster domain: objects, attributes, inter-relations, and the dynamic transformations among these objects and relations occurring over time [6.2-14].

The third component is a formal situation assessment ontology for catastrophic events, which studies what exists, what can be categorized, and whose goal is to capture the most basic structures of relevant objective reality by developing accurate and comprehensive formal systems that transparently model existing places, times, entities, properties, and relations [6.2-15]. The formal ontology framework is necessary to provide a formal structure for ontological

analysis of specific type of post-disaster situation, and to assure a certain level of reusability of the designed domain-specific ontology in a different application domain.

The combination of CWA and ontological analysis within the framework of a formal situation assessment ontology is intended to provide sufficient information about the goals, hypothesis, types of objects, relations between them, and processes to support domain specific generation of situational hypotheses and high-level reasoning about these situational hypothesis. The choice of any particular reasoning methods is defined by the domain requirements, the amount of information available, and the level and type of uncertainty of this information. Figure 1 shows the process of situation assessment design.

Situation in the first phase post-earthquake scenario consists of a set of elementary situations and their compositions. Elementary situation nominations are based on the results of the CWA and correspond to essential elements of information required by decision makers for taking actions. Among elementary situations to be considered are Communication system situation, Transportation system situation, Hazmat situation (secondary threat), Casualties situation at different levels of aggregation, Hospital situation at different levels of aggregation, Ambulance situation, and Resource situation. Each elementary situation, when considered at a certain point of time (current or future), is an event, represented by a set of hypothesis with confidence levels, by risk associated with this situation, and a set of attributes with their values characterizing this situation. Over time, each situation is also a process, which is characterized by behavior of its attributes.

6.2.5 Situation assessment node

The situation assessment design process architecture is presented in Fig 1. Information is evaluated to produce a consistent decision state estimate, which is presented for the system application (either the next automated steps or for presentation to a user).

6.2.5.1 Preprocessing

Situation assessment further processes and aggregates information about objects obtained as the result of level 1 fusion of reports on casualties and facilities damage. The results of situation assessment depend heavily on the quality of results of the Level 1 fusion processes. Although

report fusion runs constantly, fused information on each particular casualty or structure cannot enter the situation assessment process until the quality of this information is sufficient to ensure the quality of the resulting situation assessment. At the same time the situation assessment process cannot wait until the stream of reports about a certain casualty or structure is complete. Waiting may result in unacceptable decision latency, leading to either wasted resources or lost lives.

This situation calls for preprocessing, which can be implemented as decision making under a time constraint. Report fusion on a certain object should be stopped and the results passed to the situation assessment process either when the quality of the level 1 process is acceptable or a certain deadline has been reached. The quality of the level 1 estimation can be assessed, for example, by comparing the confidence level of estimates with a time-varying threshold.

6.2.5.2 Situational state estimation

Formally, let Ω be a set of possible states of the environment, $\Omega^k \subset \Omega$ a set of possible states of the environment relevant to decision maker k, and $\{S^k(t)\} \subseteq \Omega^k$ is a situational picture relevant to decision maker k at time t. Set $\{S^k(t)\}$ is represented by a set of pairs $\{H_i^k(t), Bel_i^k(t)\}$, where $\{H_i^k(t)\}$ are hypotheses of decision maker k at time t and $\{Bel_i^k(t)\}$ are corresponding levels of confidence into each hypothesis.
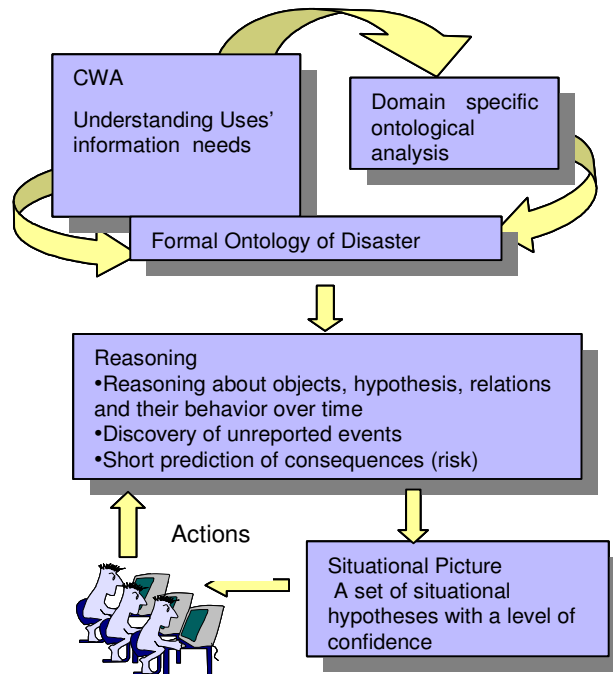
Figure 1. Situation assessment design process

6.2.5.3 Modeling framework

The modeling framework selected for our system represents a combination of domain specific models such as a hospital model and a dynamic dispatch/routing model, with time-dependent belief networks. Time-dependent belief networks (BN) are graphical models representing causal and belief relations among random variables and give an option to update those beliefs upon arriving new information. Belief networks provide intuitive and causal representations of real-world applications, and are supported by a rigorous theoretical foundation [6.2-16]. They allow expert knowledge and empirical observations to be combined (fused reports in our case), and to provide efficient uncertainty representation, which make them applicable for situation assessment. BNs consist of two parts: a directed acyclic graph representing qualitative relations between random variables, and a set of a priori and conditional beliefs which quantify these dependencies. Building the graphical representation and modeling a priori and conditional beliefs present the major challenges of BN. In our system, the graphical representation is derived from the situation assessment ontology [6.2-17].

A priori and conditional beliefs in BNs in most cases are expressed in the framework of probability theory (as Bayesian Networks) and learned from historical data and from expert knowledge and databases. In the natural and man-made disaster context, numeric historical data is sparse, the uncertainty, vagueness, and imprecision of attributes, properties, and relations are high, and many relations are represented in vague symbolic linguistic ways (*high, close, soon*, etc). All this makes the task of deriving useful model probabilities very difficult. One of an attractive ways of dealing with this highly uncertain and incomplete environment is to consider here a combination of a qualitative belief network with a Bayesian network. [6.2-18] This combination provide probabilistic reasoning in a qualitative way when numerical probabilities are not available.

6.2.5.4 Belief change

An important component of situation assessment is checking consistency of the situation picture in a Belief Change process. Situational state estimation at time *t* consists of updating the estimates obtained at *t-1*. It starts at *t=1* with updating and revising the a priori situation estimation, which is based on domain knowledge about initial risk. For example, in the case of the earthquake, the initial situation assessment is based on modeling, seismic and geophysical information about the severity of the earthquake, knowledge about building structure and vulnerabilities, population densities, and other specific and relevant information concerning the disaster zone.

New information obtained at time *t* drives changes to the state estimates obtained at time *t-1*. Traditionally, the nature of information combination in such cases is considered non-symmetrical and new information is given priority to existing information while accounting for reliability of this new information (see, e.g. [6.2-19], [6.2-20], [6.2-21]). However, in the distributed case, we need to consider separately two process, *belief revision* and *belief update*, which treat priority of incoming information differently [6.2-22].

The belief revision process modifies existing estimates at *t-1* based on new information obtained at time *t* to refine the situation assessment at time *t-1*. i.e., belief revision refers to a static situation, although it can be used in a dynamic situation when referred to locally stable conditions. The belief update process, on the other hand, modifies existing estimates at *t-1* based

on new information obtained at time *t* to build a new situation assessment at time *t.* Belief revision decides what beliefs (old or new) should be discarded to accommodate new information. Revision in the static case is based on conditioning while reliability of all beliefs have to be taken into account (see, e.g.[6.2-23]). New information may be discarded if it contradicts either domain knowledge or totally reliable previous information.

In dynamic situations, incoming information describes the changed situation and the nature of belief combination is not symmetrical. In such situations *belief update* has to be considered. In belief update an agent's beliefs should be adjusted to be consistent with a priori knowledge as well as knowledge concerning new events which occurred in the changing problem environment. Belief update attempts to decide what changes in the world led to this new information. Here incoming information is given higher priority provided that its reliability is taken into account. Transition from *Bel(t-1)* to *Bel(t)* should obey the principle of minimal change of previous beliefs to make it compatible with the new information. In dynamic situations, a Kalman-like approach to belief update can be adopted ("model-based" BR) [6.2-24]. In this case revision consists of a prediction step based on a selected model of the evolution of the world and a revision step, in which predicted state of the world is modified based on incoming information while taking into account its reliability. Incoming information can be rejected if this new state deviates too far from the predicted state. In our system the consistency check will be based on a priori risk estimation, relation between fused reports of different types and database information, and may be different for different elementary situations.

6.2.5.5 Decision state estimation

First response phase casualty mitigation operations are under severe time and resource constraints, and timely decision making and swift action are required. At the same time the cost of false alarms can be very high since valuable resources might be diverted from the location where it later becomes clear that they are critically needed. The cost of waiting for additional information, or cost of additional computation delay, has to be justified by the benefits of achieving a more accurate situation assessment. Therefore, as in the preprocessing step described in Section 3.1, the result of aggregation and situation assessment should be determined to be of a minimum threshold quality before being allowed to be used by other processes or passed on to

decision makers. The state of acceptable quality is known as a "decision state". The process of decision state estimation requires criteria for defining situation quality. One of the ways of dealing with this problem is to select a set of pivotal situational hypothesis and then to define a quality of the situation containing this hypothesis by a time-dependent confidence level associated with this hypothesis. The nomination of the pivotal hypotheses and a time-dependent confidence level can be obtained as the result of CWA. In certain situations, when decisions based on the resulting decision state estimations have very serious consequences, a sensor management process can be employed. For example, a highly reliable sensor, perhaps a policeman or structural engineer, can be tasked to observe the situation in question. The situation assessment processing logic is illustrated in Figure 2.
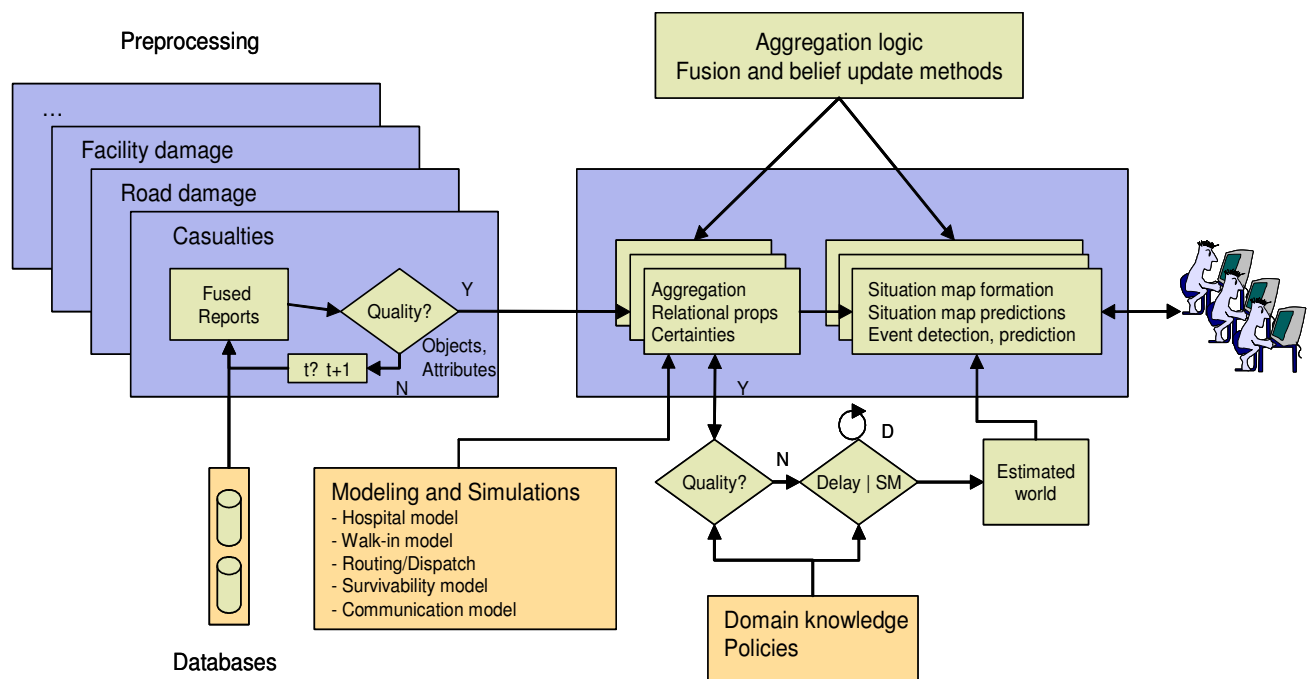
Figure 2. Situation assessment processing

To conclude, a synthetic task environment in the context of natural and man-made disasters has been constructed to explore data fusion system design and performance. Specifically, the first response phase of an event similar to the Northridge CA 1994 earthquake is modeled. Secondary Hazmat spills caused by the primary earthquake shock are included. Key features of the environment include incorporation of higher-level and distributed fusion capabilities, and

AFOSR F49620-01-1-0371

surveillance for secondary incidents, which may need to be inferred rather than directly reported to the fusion nodes.

6.2.6 DIRE initialization parameters

A DIRE simulation run is initiated by setting scenario parameters which determine the root of the random seed tree that determines all probabilitistic realizations, the selected intensity and location of the earthquake epicenter, the fraction of casualty reports which do not include positive casualty ID, etc. For ease in setting and checking, these initialization parameters are gathered in a single initialization file. A sample initialization file is listed below:

```
[general]               // first section of ini-file
RandomSeed = 9763       // RandomSeed          Dimensionless Number
SimLength = 3600        // SimLength                seconds
ZoomLevel = 0.09        // zoomlevel                Dimensionless Number
ShowDisplay = false     // showdisplay              flag
ScreenLog = ScreenLogFile.txt   //                  filename
Juris3Pol = 0.1   // OtherPercent                   percent
StudyAreaEastingMin = 899724.72     //              UTM coord
StudyAreaEastingMax = 926972.03     //              UTM coord
StudyAreaNorthingMin = 3784930.26  //               UTM coord
StudyAreaNorthingMax = 3806621.68  //               UTM coord
MaxPolSpeed = 20       // policespeed         mph
MaxAmbSpeed = 40      // ambspeed             mph
CasSeenPol = 1.0    // probpolicesee          probability
CasSeenAmb = 0.8    // probambsee             probability
ModCasLaydown = false    // resetcas           flag

[PreCalculation]
PreCalcWalkin = true     // bcalcwalkins             filename
PreCalcPolCasRept = polreports.mdb  // brecalcpolice  filename
PreCalcCivCasRept = civreports.mdb  // bcivcalls     filename

[PositionReporting]
IntervalPolGPS = 0            // PolTimeInterval     seconds
IntervalAmbGPS = 0           // AmbTimeInterval     seconds

[Shapefiles]
AmbShapefile = Ambulances  // AmbFileName          ASCII
PolShapefile = OrigAppPolice  // PolFileName       ASCII
CasShapefile = OrigAppCasualty  // CasFileName     ASCII
RoadLayerShapefile = nridgelatlong  // Roadfile     ASCII

[DeniedReports]
NoPolAmbCasualtyID = 0.75        // NoIDPercent      percent
NoCivReporterID = 0.75  // NoCivIDPercent           percent
NoPolAmbReporterID = 0.05       // NoAgentIDPercent   percent

[Limits]
PolAmbErrStdDevLocRept = 0.1  // STDErrOnLoc         meters
```

```
CivErrStdDevCasLocRept = 50 // STDErronCivRepLoc      meters
PolAmbErrStdDevCasLocRept = 25      // STDErronRepLoc meters
PercentDelayedRepts = 0.3      // RpDlPercent          percent
ReportDelayMean = 120  // RpAvrgDelay                  seconds
ReportDelayStdDev = 10  // STDErronDelay               seconds
CivCallLimitMean = 200  // UpperLimitCiv              calls/sim-interval
CivCallLimitStdDev = 20 // LimitCivStdDev              calls/sim-interval
PolAmbCallLimitMean = 65000  // UpperLimitEmerg        calls/sim-interval
PolAmbCallLimitStdDev = 1       // LimitEmergStdDev    calls/sim-interval


[Reporting]
PolViewDistance = 50    // polviewdistance             meters
AmbViewDistance = 50    // ambviewdistance             meters
RoadDlTime = 1800       // RoadDlTime
RoadDelayGain = 1.15    // RoadDlScale                 percent
RoadDelayStdDev = 0.01  // roaddelaystddev             percent
PolFalseRepts = 0.1            // PercentpolfalseReps  percent
AmbFalseRepts = 0.1            // Percentambfalsereps  percent
CivFalseRepts = 0.2            // Percentcivfalsereps  percent


[WalkinCalculation]
Sev2Walkin = 0.35       // Percent2walkin              percent
Sev3Walkin = 0.003           // Percent3walkin         percent


[HazmatPlume]
Origin_X = 908539 // plumeoriginx                      UTM coords
Origin_Y = 3795823      // plumeoriginy                UTM coords
WindDirection = 45      // winddir                 right-hand degrees from north
WindVelocity = 2      // windspeed                    mph
SpillType = 2                  // spilltype          1-bolus | 2-gradual
StartPlume = 15000             // startplume          seconds
PlumeInterval = 150            // plumeinterval        seconds
NewCasThreshold = 1  // newcasthresh                   hazmat concentration
NewCasScale = 2         // newcasmult                  new cas scaling factor


[Shelters]
ShelterStart = 30             // shelterstart
ShelterInterval = 600         // shelterinterval
ShelterDistance = 300         // shelterdist


[ToBeDeleted]
bResetRoads = false           // bResetRoads
Resetdatabases = true         // Resetdatabases
ResetPolsent = true           // ResetPolsent
ResetCivsent = true           // ResetCivsent
AmbShpfile = XYAmbulance      // AmbShpfile
DelayFile = Delay.mdb         // DelayFile
ShapesDatabase = shapes.mdb   // Shapesdb             ASCII
CasInfo = CasInfo.mdb         // CasInfo              ASCII
JurisShapefile = jurisdiction // JurFileName          ASCII
WindVelocityX = 5       // windvelx                    mph
WindVelocityY = 5       // windvely                    mph
SpillRate = 2                  // spillrate
ReleaseDuration = 7200  // duration                    seconds
Dispersion_X = 1               // dispersionx          Dimensionless Number
Dispersion_Y = 1      // dispersiony                   Dimensionless Number
```

Brief descriptions of these parameters:

RandomSeed   'integer declaring what the random seed should be. Zero means choose a random seed

bcalcwalkins   'boolean telling whether to calculate walkins

polreps   'string that contains the filename of the police reports that were precalculated. If NULL then make new reports and save in polreports.mdb

civreports   'string of the filename for precalculated civilian reports. If NULL then recalculate civilian calls and save in civreports.mdb

PolTimeInterval   'number giving the time interval between police location reports

AmbTimeInterval   'number giving the time interval between ambulance location reports

SimLength   'number giving the total length of the simulation

CasFileName   'casualty shapefile base name

PolFileName   'police shapefile base name'

AmbFileName   'ambulance shapefile base name

RoadFile   'road shapefile base name

NoIDPercent   'percent of reports containing no casualty id

UpperLimitCiv   'upper bound on the number of reports civilians can report

LimitCivStdDev   'standard deviation on upper bound of civilian reports

UpperLimitEmerg   'upper bound on the number of reports emergency personnel can report

LimitEmergStdDev   'standard deviation on upper bound of emergency personnel reports

RpDlPercent   'percent of reports delayed by a certain amt

RpAvrgDelay   'average time of report delay

STDErronDelay   'standard deviation of report delay

STDErronRepLoc   'standard deviation of location given in reports

STDErronCivRepLoc 'standard deviation of location field in civilian reports

STDErrOnLoc   'standard deviation of spatial error

OtherPercent   'percent of police from other juris

ambviewdistance   'distance ambulances can view

polviewdistance   'distance police can view

RoadDlScale	'scale factor for road delay each 'roadtimeinterval'
roaddelaystddev	'standard deviation for random noise in road delay changes (mean is zero)
RoadDlTime	'time interval between which road delay scales
PercentpolfalseReps	'percentage of police reports that are false
Percentambfalsereps	'percentage of ambulance reports that are false
Percentcivfalsereps	'percentage of civilian reports that are false
Percent2walkin	'percentage of severity 2 that walkin
Percent3walkin	'percentage of severity 3 that walkin
NoAgentIdPercent	'percentage of reports that are sent without the police or ambulance ID
NoCivIDPercent	'percentage of civilian reports sent with casualty ID missing
zoomlevel	'level for zooming into movement (smaller is closer in)
showdisplay	' tells whether or not to show the display. To show the display input 'true', else input 'false'
probpolicesee	'the probability a policeman sees a casualty
probambsee	'the probability an ambulance sees a casualty
policespeed	'maximum speed a policeman can travel
ambspeed	'maximum speed an ambulance can travel
plumeoriginx	'x coordinate in UTM of plume origin
plumeoriginy	'y coordinate in UTM of plume origin
winddir	'direction of the wind for the plume
windspeed	'wind speed of the plume
spilltype	'1-bolus spill and 2-gradual release
startplume	'this is the time, in seconds, after the onset of the disaster that the plume begins
plumeinterval	'this is the time between plume updates in seconds
shelterstart	'this is the time when the shelters start affecting travel delays around them
shelterinterval	'this is the time between shelter updates of road link delays around them

shelterdist   'roads within this distance from a shelter have their delay affected
                     (meters)

6.2.7 DIRE message formats

In this section are listed all message formats governing interactions within the DIRE federation.
They are grouped by message source: which federate is publishing the message. Each distinct
message format is also identified by its destination, ie. which federate is subscribed to that
message.

A – **Message   Source:**   ReportGenerator

        Destination: L1Fusion

Casualty observation

```
SourceDestination      // "RGtoDF01:Casualty Observation"
Report_Type            // 2 digit integer            unsigned int
Reporter_ID            // 5 digit integer            unsigned long
JurisdictionID         // 1 digit integer            unsigned int
Time                   // seconds                     unsigned long
SrcDstCount            // counter for this Int-name  unsigned long
TrackID                // recnum * 100 + FileID       unsigned long

Location_X             // UTM – easting value        std::string
Location_Y             // UTM – northing value       std::string
CensusTractID          // Tract where found          std::string
NearestNode            // Roadway intersection ID    std::string
Casualty_ID            // injured person             std::string
Casualty_Age           // 1 digit integer            unsigned int
Casualty_Race          // 1 digit integer            unsigned int
Casualty_Sex           // 1 digit integer            unsigned int
InjuryType             // 1 digit 0..4               unsigned int
Severity               // 1 digit integer            unsigned int
----------------------------------------------------------------------
```

Medical facility damage

```
SourceDestination      // "RGtoDF02:Medical Facility Damage"
Report_Type            // 2 digit integer            unsigned int
Reporter_ID            // 5 digit integer            unsigned long
JurisdictionID         // 1 digit integer            unsigned int
Time                   // seconds                     unsigned long
SrcDstCount            // counter for this Int-name  unsigned long
TrackID                // recnum * 100 + FileID       unsigned long

Location_X             // UTM – easting value        std::string
Location_Y             // UTM – northing value       std::string
Facility_ID            // Medical facility           std::string
Severity               // 1 digit integer            unsigned int
```

```
--------------------------------------------------------------------
Roadway damage

SourceDestination        // "RGtoDF03:Roadway Damage"
Report_Type              // 2 digit integer          unsigned int
Reporter_ID              // 5 digit integer          unsigned long
JurisdictionID           // 1 digit integer          unsigned int
Time                     // seconds                  unsigned long
SrcDstCount              // counter for this Int-name unsigned long
TrackID                  // recnum * 100 + FileID     unsigned long

BridgeID                 // Bridge                   std::string
LinkID                   // road section             std::string
Severity                 // 1 digit integer          unsigned int


--------------------------------------------------------------------
Casualty pickup

SourceDestination        // "RGtoDF04:Casualty Pickup"
Report_Type              // 2 digit integer          unsigned int
Reporter_ID              // 5 digit integer          unsigned long
JurisdictionID           // 1 digit integer          unsigned int
Time                     // seconds                  unsigned long
SrcDstCount              // counter for this Int-name unsigned long
TrackID                  // recnum * 100 + FileID     unsigned long

Location_X               // UTM - easting value      std::string
Location_Y               // UTM - northing value     std::string
NearestNode              // Roadway intersection ID  std::string
Casualty_ID              // injured person           std::string
Casualty_Age             // 1 digit integer          unsigned int
Casualty_Race            // 1 digit integer          unsigned int
Casualty_Sex             // I digit integer          unsigned int
InjuryType               // 1 digit 0..4             unsigned int
Severity                 // 1 digit integer          unsigned int


--------------------------------------------------------------------

Casualty delivery

SourceDestination        // "RGtoDF05:Casualty Arrival"
Report_Type              // 2 digit integer          unsigned int
Reporter_ID              // also hospital ID         unsigned long
JurisdictionID           // 1 digit integer          unsigned int
Time                     // seconds                  unsigned long
SrcDstCount              // counter for this Int-name unsigned long
TrackID                  // recnum * 100 + FileID     unsigned long

Location_X               // UTM - easting value      std::string
Location_Y               // UTM - northing value     std::string
NearestNode              // Roadway intersection ID  std::string
Casualty_ID              // injured person           std::string
Casualty_Age             // 1 digit integer          unsigned int
Casualty_Race            // 1 digit integer          unsigned int
```

```
Casualty_Sex           // 1 digit integer           unsigned int
InjuryType             // 1 digit 0..4              unsigned int
Severity               // 1 digit integer           unsigned int


------------------------------------------------------------------------


Police location

SourceDestination      // "RGtoDF06:Police Location"
Report_Type            // 2 digit integer           unsigned int
Reporter_ID            // 5 digit integer           unsigned long
JurisdictionID         // 1 digit integer           unsigned int
Time                   // seconds                   unsigned long
SrcDstCount            // counter for this Int-name  unsigned long
TrackID                // recnum * 100 + FileID      unsigned long

Location_X             // UTM – easting value        std::string
Location_Y             // UTM – northing value       std::string


------------------------------------------------------------------------

Ambulance Location

SourceDestination      // "RGtoDF07:Ambulance Location"
Report_Type            // 2 digit integer           unsigned int
Reporter_ID            // 5 digit integer           unsigned long
JurisdictionID         // 1 digit integer           unsigned int
Time                   // seconds                   unsigned long
SrcDstCount            // counter for this Int-name  unsigned long
TrackID                // recnum * 100 + FileID      unsigned long

Location_X             // UTM – easting value        std::string
Location_Y             // UTM – northing value       std::string



------------------------------------------------------------------------


Medical Facility Capacity

SourceDestination      // "RGtoDF08:Medical Facility – Capacity"
Report_Type            // 2 digit integer           unsigned int
Reporter_ID            // 5 digit integer           unsigned long
JurisdictionID         // 1 digit integer           unsigned int
Time                   // seconds                   unsigned long
SrcDstCount            // counter for this Int-name  unsigned long
TrackID                // recnum * 100 + FileID      unsigned long

Severity_1             // 2 digit integer           unsigned int
Severity_2             // 2 digit integer           unsigned int
Severity_3             // 2 digit integer           unsigned int


------------------------------------------------------------------------

Casualty Treatment Delay

SourceDestination      // "RGtoDF09:Casualty Treatment Delay"
Report_Type            // 2 digit integer           unsigned int
Reporter_ID            // 5 digit integer           unsigned long
```

```
JurisdictionID          // 1 digit integer              unsigned int
Time                    // seconds                      unsigned long
SrcDstCount             // counter for this Int-name    unsigned long
TrackID                 // recnum * 100 + FileID         unsigned long

Delay_1                 // 4 digit integer              unsigned int
Delay_2                 // 4 digit integer              unsigned int
Delay_3                 // 4 digit integer              unsigned int
```

Ambulance Idle

```
SourceDestination       // "RGtoDF10:Ambulance Idle"
Report_Type             // 2 digit integer              unsigned int
Reporter_ID             // 5 digit integer              unsigned long
JurisdictionID          // 1 digit integer              unsigned int
Time                    // seconds                      unsigned long
SrcDstCount             // counter for this Int-name    unsigned long
TrackID                 // recnum * 100 + FileID         unsigned long

Location_X              // UTM – easting value          std::string
Location_Y              // UTM – northing value         std::string
NearestNode             // Roadway intersection ID      std::string
Position_ID             // Location ID                  std::string
OnBoard_2               // number of patients – Sev 2   unsigned int
OnBoard_3               // number of patients – Sev 3   unsigned int
```

Ambulance Stuck

```
SourceDestination       // "RGtoDF11:Ambulance Stuck"
Report_Type             // 2 digit integer              unsigned int
Reporter_ID             // 5 digit integer              unsigned long
JurisdictionID          // 1 digit integer              unsigned int
Time                    // seconds                      unsigned long
SrcDstCount             // counter for this Int-name    unsigned long
TrackID                 // recnum * 100 + FileID         unsigned long

Location_X              // UTM – easting value          std::string
Location_Y              // UTM – northing value         std::string
NearestNode             // Roadway intersection ID      std::string
LinkID                  // road section                 std::string
OnBoard_2               // number of patients – Sev 2   unsigned int
OnBoard_3               // number of patients – Sev 3   unsigned int
```

Travel Delay

```
SourceDestination       // "RGtoDF12:Travel Delay"
Report_Type             // 2 digit integer              unsigned int
Reporter_ID             // 5 digit integer              unsigned long
JurisdictionID          // 1 digit integer              unsigned int
```

```
Time                    // seconds                    unsigned long
SrcDstCount             // counter for this Int-name   unsigned long
TrackID                 // recnum * 100 + FileID        unsigned long

LinkID                  // road section                std::string
Severity                // 00..100                     unsigned int
```

Cluster Ident

```
SourceDestination       // "RGtoDF13:Cluster Ident"
Report_Type             // 2 digit integer             unsigned int
Reporter_ID             // 5 digit integer             unsigned long
JurisdictionID          // 1 digit integer             unsigned int
Time                    // seconds                     unsigned long
SrcDstCount             // counter for this Int-name   unsigned long
TrackID                 // recnum * 100 + FileID        unsigned long

ClusterID               // integer                     unsigned int
CellSide                // length of cell (meters)      unsigned long
CellCount               // number of cells in cluster   unsigned int
    CellType            // 1 => Non-boundary            unsigned int
                        // 2 => Boundary

    CellCenter_X        // X-Coord of cell center       unsigned long


    CellCenter_Y        // Y-Coord of cell center       unsigned long


    Sev2CasCount        // Cas severity 2 Cell Count    unsigned long


    Sev3CasCount        // Cas severity 3 Cell Count    unsigned long
```

B – Message   Source: L1Fusion

**   Destination: EstimateDirector**

Casualty Observation
```
SourceDestination       // "DFtoED01:Casualty Observation"
Report_Type             // 2 digit integer             unsigned int
Reporter_ID             // 5 digit integer             unsigned long
JurisdictionID          // 1 digit integer             unsigned int
Time                    // seconds                     unsigned long
SrcDstCount             // counter for this Int-name   unsigned long
TrackID                 // recnum * 100 + FileID        unsigned long

Location_X              // UTM - easting value         std::string
Location_Y              // UTM - northing value        std::string
CensusTractID           // Tract where found           std::string
LocErrCovX              // Location Error Covariance    double
LocErrCovY              // Location Error Covariance    double
NearestNode             // Roadway intersection ID      std::string
Casualty_ID             // injured person              std::string
```

```
Casualty_Age              // 1 digit integer          unsigned int
Casualty_Race             // 1 digit integer          unsigned int
Casualty_Sex              // I digit integer          unsigned int
Severity                  // 1 digit integer          unsigned int
SevProbVect               // Severity probabilities    double[4]
ReportCountP              // Police                    unsigned int
ReportCountA              // Ambulance                 unsigned int
ReportCountC              // Civilian                  unsigned int
CumAssocProb              // Cumulative Association     double


--------------------------------------------------------------------
```

Medical Facility Damage

```
SourceDestination         // "DFtoED02:Medical Facility Damage"
Report_Type               // 2 digit integer          unsigned int
Reporter_ID               // 5 digit integer          unsigned long
JurisdictionID            // 1 digit integer          unsigned int
Time                      // seconds                   unsigned long
SrcDstCount               // counter for this Int-name  unsigned long
TrackID                   // recnum * 100 + FileID     unsigned long


Location_X                // UTM – easting value       std::string
Location_Y                // UTM – northing value      std::string
Facility_ID               // medical facility          std::string
Severity                  // 1 digit integer          unsigned int


--------------------------------------------------------------------
```

Roadway Damage

```
SourceDestination         // "DFtoED03:Roadway Damage"
Report_Type               // 2 digit integer          unsigned int
Reporter_ID               // 5 digit integer          unsigned long
JurisdictionID            // 1 digit integer          unsigned int
Time                      // seconds                   unsigned long
SrcDstCount               // counter for this Int-name  unsigned long
TrackID                   // recnum * 100 + FileID     unsigned long


BridgeID                  // bridge                    std::string
LinkID                    // road section              std::string
Severity                  // 1 digit integer          unsigned int
--------------------------------------------------------------------
```

Casualty Pickup

```
SourceDestination         // "DFtoED04:Casualty Pickup"
Report_Type               // 2 digit integer          unsigned int
Reporter_ID               // 5 digit integer          unsigned long
JurisdictionID            // 1 digit integer          unsigned int
Time                      // seconds                   unsigned long
SrcDstCount               // counter for this Int-name  unsigned long
TrackID                   // recnum * 100 + FileID     unsigned long


Location_X                // UTM – easting value       std::string
Location_Y                // UTM – northing value      std::string
NearestNode               // Roadway intersection ID   std::string
Casualty_ID               // injured person            std::string
Casualty_Age              // 1 digit integer          unsigned int
```

```
Casualty_Race           // 1 digit integer              unsigned int
Casualty_Sex            // I digit integer              unsigned int
Severity                // 1 digit integer              unsigned int


--------------------------------------------------------------------


Casualty Delivery
SourceDestination       // "DFtoED05:Casualty Arrival"
Report_Type             // 2 digit integer              unsigned int
Reporter_ID             // 5 digit integer              unsigned long
JurisdictionID          // 1 digit integer              unsigned int
Time                    // seconds                      unsigned long
SrcDstCount             // counter for this Int-name    unsigned long
TrackID                 // recnum * 100 + FileID        unsigned long


Location_X              // UTM - easting value          std::string
Location_Y              // UTM - northing value         std::string
NearestNode             // Roadway intersection ID      std::string
Hospital_ID             // hospital                     std::string
Casualty_ID             // injured person               std::string
Casualty_Age            // 1 digit integer              unsigned int
Casualty_Race           // 1 digit integer              unsigned int
Casualty_Sex            // I digit integer              unsigned int
Severity                // 1 digit integer              unsigned int


--------------------------------------------------------------------


Police Location

SourceDestination       // "DFtoED06:Police Location"
Report_Type             // 2 digit integer              unsigned int
Reporter_ID             // 5 digit integer              unsigned long
JurisdictionID          // 1 digit integer              unsigned int
Time                    // seconds                      unsigned long
SrcDstCount             // counter for this Int-name    unsigned long
TrackID                 // recnum * 100 + FileID        unsigned long
SrcDstCount             // counter for this Int-name    unsigned long
TrackID                 // recnum * 100 + FileID        unsigned long

Location_X              // UTM - easting value          std::string
Location_Y              // UTM - northing value         std::string


--------------------------------------------------------------------
Ambulance Location

SourceDestination       // "DFtoED07:Ambulance Location"
Report_Type             // 2 digit integer              unsigned int
Reporter_ID             // 5 digit integer              unsigned long
JurisdictionID          // 1 digit integer              unsigned int
Time                    // seconds                      unsigned long
SrcDstCount             // counter for this Int-name    unsigned long
TrackID                 // recnum * 100 + FileID        unsigned long

Location_X              // UTM - easting value          std::string
Location_Y              // UTM - northing value         std::string


--------------------------------------------------------------------
```

AFOSR F49620-01-1-0371

Medical Facility Capacity

```
SourceDestination        // "DFtoED08:Medical Facility Capacity"
Report_Type              // 2 digit integer          unsigned int
Reporter_ID              // 5 digit integer          unsigned long
JurisdictionID           // 1 digit integer          unsigned int
Time                     // seconds                  unsigned long
SrcDstCount              // counter for this Int-name unsigned long
TrackID                  // recnum * 100 + FileID     unsigned long


Severity_1               // 2 digit integer          unsigned int
Severity_2               // 2 digit integer          unsigned int
Severity_3               // 2 digit integer          unsigned int
```

Casualty Treatment Delay

```
SourceDestination        // "DFtoED09:Casualty Treatment Delay"
Report_Type              // 2 digit integer          unsigned int
Reporter_ID              // 5 digit integer          unsigned long
JurisdictionID           // 1 digit integer          unsigned int
Time                     // seconds                  unsigned long
SrcDstCount              // counter for this Int-name unsigned long
TrackID                  // recnum * 100 + FileID     unsigned long


Delay_1                  // 4 digit integer          unsigned int
Delay_2                  // 4 digit integer          unsigned int
Delay_3                  // 4 digit integer          unsigned int
```

Ambulance Idle

```
SourceDestination        // "DftoED10:Ambulance Idle"
Report_Type              // 2 digit integer          unsigned int
Reporter_ID              // 5 digit integer          unsigned long
JurisdictionID           // 1 digit integer          unsigned int
Time                     // seconds                  unsigned long
SrcDstCount              // counter for this Int-name unsigned long
TrackID                  // recnum * 100 + FileID     unsigned long


Location_X               // UTM – easting value      std::string
Location_Y               // UTM – northing value     std::string
NearestNode              // Roadway intersection ID  std::string
Position                 // location ID              std::string
OnBoard_2                // number of patients – Sev 2 unsigned int
OnBoard_3                // number of patients – Sev 3 unsigned int
```

Ambulance Stuck

```
SourceDestination        // "DFtoED11:Ambulance Stuck"
Report_Type              // 2 digit integer          unsigned int
Reporter_ID              // 5 digit integer          unsigned long
JurisdictionID           // 1 digit integer          unsigned int
```

```
Time                    // seconds                  unsigned long
SrcDstCount             // counter for this Int-name unsigned long
TrackID                 // recnum * 100 + FileID     unsigned long

Location_X              // UTM – easting value       std::string
Location_Y              // UTM – northing value      std::string
NearestNode             // Roadway intersection ID   std::string
LinkID                  // road section              std::string
OnBoard_2               // number of patients – Sev 2 unsigned int
OnBoard_3               // number of patients – Sev 3 unsigned int


------------------------------------------------------------------------
```

Travel Delay

```
SourceDestination       // "DFtoED12:Travel Delay"
Report_Type             // 2 digit integer           unsigned int
Reporter_ID             // 5 digit integer           unsigned long
JurisdictionID          // 1 digit integer           unsigned int
Time                    // seconds                   unsigned long
SrcDstCount             // counter for this Int-name  unsigned long
TrackID                 // recnum * 100 + FileID      unsigned long

LinkID                  // road section              std::string
Severity                // 00..100                   unsigned int


------------------------------------------------------------------------
```

Cluster Ident

```
SourceDestination       // "DFtoED13:Cluster Ident"
Report_Type             // 2 digit integer           unsigned int
Reporter_ID             // 5 digit integer           unsigned long
JurisdictionID          // 1 digit integer           unsigned int
Time                    // seconds                   unsigned long
SrcDstCount             // counter for this Int-name  unsigned long
TrackID                 // recnum * 100 + FileID      unsigned long

ClusterID               // integer                   unsigned int
CellSide                // length of cell (meters)    unsigned long
CellCount               // number of cells in cluster unsigned int
     CellType           // 1 => Non-boundary          unsigned int
                        // 2 => Boundary
     CellCenter_X       // X-Coord of cell center     unsigned long


     CellCenter_Y       // Y-Coord of cell center     unsigned long


     Sev2CasCount       // Cas severity 2 Cell Count  unsigned long


     Sev3CasCount       // Cas severity 3 Cell Count  unsigned long
```

C – Message   Source: EstimateDirector

**   Destination: DispatcherRouter**

Casualty Observation

```
SourceDestination        // "EDtoDP01:Casualty Observation"
Report_Type              // 2 digit integer           unsigned int
Reporter_ID              // 5 digit integer           unsigned long
JurisdictionID           // 1 digit integer           unsigned int
Time                     // seconds                   unsigned long
SrcDstCount              // counter for this Int-name  unsigned long
TrackID                  // recnum * 100 + FileID      unsigned long

Location_X               // UTM - easting value        std::string
Location_Y               // UTM - northing value       std::string
NearestNode              // Roadway intersection       std::string
Casualty_ID              // injured person             std::string
Severity                 // 1 digit integer           unsigned int
SevProbVect              // Severity probabilities     double[4]
```

Roadway Damage

```
SourceDestination        // "EDtoDP04:Roadway Damage"
Report_Type              // 2 digit integer           unsigned int
Reporter_ID              // 5 digit integer           unsigned long
JurisdictionID           // 1 digit integer           unsigned int
Time                     // seconds                   unsigned long
SrcDstCount              // counter for this Int-name  unsigned long
TrackID                  // recnum * 100 + FileID      unsigned long

LinkID                   // road section              std::string
Severity                 // 1 digit integer           unsigned int
```

Medical Facility Capacity

```
SourceDestination        // "EDtoDP05:Medical Facility Capacity"
Report_Type              // 2 digit integer           unsigned int
Reporter_ID              // 5 digit integer           unsigned long
JurisdictionID           // 1 digit integer           unsigned int
Time                     // seconds                   unsigned long
SrcDstCount              // counter for this Int-name  unsigned long
TrackID                  // recnum * 100 + FileID      unsigned long

Severity_1               // 2 digit integer           unsigned int
Severity_2               // 2 digit integer           unsigned int
Severity_3               // 2 digit integer           unsigned int
```

Travel Delay

```
SourceDestination        // "EDtoDP06:Travel Delay"
Report_Type              // 2 digit integer           unsigned int
Reporter_ID              // 5 digit integer           unsigned long
JurisdictionID           // 1 digit integer           unsigned int
```

```
Time                    // seconds                       unsigned long
SrcDstCount             // counter for this Int-name  unsigned long
TrackID                 // recnum * 100 + FileID          unsigned long

LinkID                  // road section                  std::string
Severity                // 00..100                        unsigned int

---------------------------------------------------------------------
```

Casualty Treatment Delay

```
SourceDestination       // "EDtoDP07:Casualty Treatment Delay"
Report_Type             // 2 digit integer                unsigned int
Reporter_ID             // 5 digit integer                unsigned long
JurisdictionID          // 1 digit integer                unsigned int
Time                    // seconds                       unsigned long
SrcDstCount             // counter for this Int-name  unsigned long
TrackID                 // recnum * 100 + FileID          unsigned long

Delay_1                 // 4 digit integer                unsigned int
Delay_2                 // 4 digit integer                unsigned int
Delay_3                 // 4 digit integer                unsigned int

---------------------------------------------------------------------
```

Ambulance Idle

```
SourceDestination       // "EDtoDP08:Ambulance Idle"
Report_Type             // 2 digit integer                unsigned int
Reporter_ID             // 5 digit integer                unsigned long
JurisdictionID          // 1 digit integer                unsigned int
Time                    // seconds                       unsigned long
SrcDstCount             // counter for this Int-name  unsigned long
TrackID                 // recnum * 100 + FileID          unsigned long

Location_X              // UTM – easting value            std::string
Location_Y              // UTM – northing value           std::string
NearestNode             // Roadway intersection           std::string
Position                // Location ID                    std::string
OnBoard_2               // number of patients – Sev 2 unsigned int
OnBoard_3               // number of patients – Sev 3 unsigned int

---------------------------------------------------------------------
```

Ambulance Stuck

```
SourceDestination       // "EDtoDP09:Ambulance Stuck"
Report_Type             // 2 digit integer                unsigned int
Reporter_ID             // 5 digit integer                unsigned long
JurisdictionID          // 1 digit integer                unsigned int
Time                    // seconds                       unsigned long
SrcDstCount             // counter for this Int-name  unsigned long
TrackID                 // recnum * 100 + FileID          unsigned long

Location_X              // UTM – easting value            std::string
Location_Y              // UTM – northing value           std::string
NearestNode             // Roadway intersection           std::string
LinkID                  // road section                  std::string
```

```
OnBoard_2              // number of patients - Sev 2 unsigned int
OnBoard_3              // number of patients - Sev 3 unsigned int


------------------------------------------------------------------
```
Cluster Ident

```
SourceDestination      // "EDtoDP10:Cluster Ident"
Report_Type            // 2 digit integer           unsigned int
Reporter_ID            // 5 digit integer           unsigned long
JurisdictionID         // 1 digit integer           unsigned int
Time                   // seconds                   unsigned long
SrcDstCount            // counter for this Int-name  unsigned long
TrackID                // recnum * 100 + FileID      unsigned long


ClusterID              // integer                   unsigned int
CellSide               // length of cell (meters)    unsigned long
CellCount              // number of cells in cluster unsigned int
     CellType          // 1 => Non-boundary          unsigned int
                       // 2 => Boundary
     CellCenter_X      // X-Coord of cell center     unsigned long


     CellCenter_Y      // Y-Coord of cell center     unsigned long


     Sev2CasCount      // Cas severity 2 Cell Count  unsigned long


     Sev3CasCount      // Cas severity 3 Cell Count  unsigned long
```

D – Message   Source: EstimateDirector

**   Destination: MedicalFacility**

Medical Facility Damage

```
SourceDestination      // "EDtoMF01:Medical Facility Damage"
Report_Type            // 2 digit integer           unsigned int
Reporter_ID            // 5 digit integer           unsigned long
JurisdictionID         // 1 digit integer           unsigned int
Time                   // seconds                   unsigned long
SrcDstCount            // counter for this Int-name  unsigned long
TrackID                // recnum * 100 + FileID      unsigned long

Location_X             // UTM - easting value       std::string
Location_Y             // UTM - northing value      std::string
Facility_ID            // medical facility          std::string
Severity               // 1 digit integer           unsigned int


------------------------------------------------------------------
```
Casualty Delivery

```
SourceDestination      // "EDtoMF02:Casualty Delivery"
Report_Type            // 2 digit integer           unsigned int
```

AFOSR F49620-01-1-0371

```
Reporter_ID          // 5 digit integer          unsigned long
JurisdictionID       // 1 digit integer          unsigned int
Time                 // seconds                  unsigned long
SrcDstCount          // counter for this Int-name unsigned long
TrackID              // recnum * 100 + FileID     unsigned long

Hospital_ID          // hospital                 std::string
Casualty_ID          // injured person           std::string
Casualty_Age         // 1 digit integer          unsigned int
Casualty_Race        // 1 digit integer          unsigned int
Casualty_Sex         // I digit integer          unsigned int

Severity             // 1 digit integer          unsigned int
```

Hospital Location

```
SourceDestination    // "EDtoMF03:Hospital Location"
Report_Type          // 2 digit integer          unsigned int
Reporter_ID          // 5 digit integer          unsigned long
JurisdictionID       // 1 digit integer          unsigned int
Time                 // seconds                  unsigned long
SrcDstCount          // counter for this Int-name unsigned long
TrackID              // recnum * 100 + FileID     unsigned long
LocCount             // Number of hospLocs        unsigned int
HospLocs:
```

```
    Hospital_ID      // Medical facility         std::string
    Location_X       // UTM – easting value      std::string
    Location_Y       // UTM – northing value     std::string
    BedCount         // Size of Facility         std::string
```

F – Message   Source: MedicalFacility

**   Destination: ReportGenerator**

Medical Facility Capacity

```
SourceDestination    // "MFtoRG01:Medical Facility Capacity"
Report_Type          // 2 digit integer          unsigned int
Reporter_ID          // 5 digit integer          unsigned long
JurisdictionID       // 1 digit integer          unsigned int
Time                 // seconds                  unsigned long
SrcDstCount          // counter for this Int-name unsigned long
TrackID              // recnum * 100 + FileID     unsigned long

Severity_1           // 2 digit integer          unsigned int
Severity_2           // 2 digit integer          unsigned int
Severity_3           // 2 digit integer          unsigned int
```

Casualty Treatment Delay

AFOSR F49620-01-1-0371

```
SourceDestination        // "MFtoRG02:Casualty Treatment Delay"
Report_Type              // 2 digit integer            unsigned int
Reporter_ID              // 5 digit integer            unsigned long
JurisdictionID           // 1 digit integer            unsigned int
Time                     // seconds                    unsigned long
SrcDstCount              // counter for this Int-name   unsigned long
TrackID                  // recnum * 100 + FileID       unsigned long

Delay_1                  // 4 digit integer            unsigned int
Delay_2                  // 4 digit integer            unsigned int
Delay_3                  // 4 digit integer            unsigned int
```

G – Message   Source: **DispatcherRouter**

                       Destination: **ReportGenerator**

Ambulance Route
```
SourceDestination        // "DPtoRG01:Ambulance Route"
Report_Type              // 2 digit integer            unsigned int
Reporter_ID              // 5 digit integer            unsigned long
JurisdictionID           // 1 digit integer            unsigned int
Time                     // seconds                    unsigned long
SrcDstCount              // counter for this Int-name   unsigned long
TrackID                  // recnum * 100 + FileID       unsigned long


Ambulance_ID             // ambulance                  std::string

Casualty_ID              // injured person             std::string


Hospital_ID              // Hospital ident number      std::string


RouteType                // to cas (0), to hosp (1)    unsigned int

SegCount                 // number of segments         unsigned int

Segment[SegCount]        // Array of segments          array

Segment:
```

```
  LinkID                 // link (8-10 digits)         std::string
  NodeID                 // node following link        std::string
```

H – Message   Source: **L1Fusion**

                       Destination: **L2Fusion**

Casualty observation
```
SourceDestination        // "DFtoL201:Casualty Observation"
Report_Type              // 2 digit integer            unsigned int
Reporter_ID              // 5 digit integer            unsigned long
```

```
JurisdictionID          // 1 digit integer         unsigned int
Time                    // seconds                 unsigned long
SrcDstCount             // counter for this Int-name unsigned long
TrackID                 // recnum * 100 + FileID    unsigned long

Location_X              // UTM – easting value      std::string
Location_Y              // UTM – northing value     std::string
CensusTractID           // Tract where found        std::string
LocErrCovX              // Location Error Covariance double
LocErrCovY              // Location Error Covariance double
Casualty_ID             // injured person           std::string
Casualty_Age            // 1 digit integer          unsigned int
Casualty_Race           // 1 digit integer          unsigned int
Casualty_Sex            // I digit integer          unsigned int
InjuryType              // 1 digit 0..4             unsigned int
Severity                // 1 digit integer          unsigned int
SevProbVect             // Severity probabilities   double[4]
ReportCountP            // Police                   unsigned int
ReportCountA            // Ambulance                unsigned int
ReportCountC            // Civilian                 unsigned int
CumAssocProb            // Cumulative Association    double
FalseAlarmProb          // False Alarm Probability   double
-----------------------------------------------------------------------
```

Casualty Pickup

```
SourceDestination       // "DFtoL202:Casualty Pickup"
Report_Type             // 2 digit integer          unsigned int
Reporter_ID             // 5 digit integer          unsigned long
JurisdictionID          // 1 digit integer          unsigned int
Time                    // seconds                  unsigned long
SrcDstCount             // counter for this Int-name unsigned long
TrackID                 // recnum * 100 + FileID     unsigned long

Location_X              // UTM easting              std::string
Location_Y              // UTM northing             std::string
Casualty_ID             // injured person           std::string
NearestNode             // Roadway intersection     std::string
InjuryType              // 1 digit 0..4             unsigned int
Severity                // 1 digit integer          unsigned int
```

J – Message   Source: L2Fusion
                    Destination: ReportGenerator

Cluster Ident

```
SourceDestination       // "L2toRG01:Cluster Ident"
Report_Type             // 2 digit integer          unsigned int
Reporter_ID             // 5 digit integer          unsigned long
JurisdictionID          // 1 digit integer          unsigned int
Time                    // seconds                  unsigned long
SrcDstCount             // counter for this Int-name unsigned long
TrackID                 // recnum * 100 + FileID     unsigned long

ClusterID               // integer                  unsigned int
CellSide                // length of cell (meters)  unsigned long
```

AFOSR F49620-01-1-0371

```
CellCount                 // number of cells in cluster unsigned int
    CellType              // 1 => Non-boundary        unsigned int
                          // 2 => Boundary
    CellCenter_X          // X-Coord of cell center    unsigned long


    CellCenter_Y          // Y-Coord of cell center    unsigned long


    Sev2CasCount          // Cas severity 2 Cell Count  unsigned long


    Sev3CasCount          // Cas severity 3 Cell Count  unsigned long
```

K – **Message**   **Source:**   **ReportGenerator**

                         Destination: **MedicalFacility**

Hospital Location

```
SourceDestination       // "RGtoMF01:Hospital Location"
Report_Type             // 2 digit integer          unsigned int
Reporter_ID             // 5 digit integer          unsigned long
JurisdictionID          // 1 digit integer          unsigned int
Time                    // seconds                  unsigned long
SrcDstCount             // counter for this Int-name  unsigned long
TrackID                 // recnum * 100 + FileID     unsigned long

LocCount                // Number of hospLocs        unsigned int
HospLocs:               // hospital locations        vector
```

```
    Location_X          // UTM – easting value       std::string
    Location_Y          // UTM – northing value      std::string
    Hospital_ID         // Medical facility          std::string
    BedCount            // size of facility          std::string
```

L – **Message**   **Source:**   **EstimateDirector**

                         Destination: **Visualization**

Casualty Observation
```
SourceDestination       // "EDtoVZ01:Casualty Observation"
Report_Type             // 2 digit integer          unsigned int
Reporter_ID             // 5 digit integer          unsigned long
JurisdictionID          // 1 digit integer          unsigned int
Time                    // seconds                  unsigned long
SrcDstCount             // counter for this Int-name  unsigned long
TrackID                 // recnum * 100 + FileID     unsigned long

Location_X              // UTM – easting value       std::string
Location_Y              // UTM – northing value      std::string
LocErrCovX              // Location Error Covariance double
LocErrCovY              // Location Error Covariance double
```

```
NearestNode              // Roadway intersection ID    std::string
Casualty_ID              // injured person             std::string
Casualty_Age             // 1 digit integer            unsigned int
Casualty_Race            // 1 digit integer            unsigned int
Casualty_Sex             // I digit integer            unsigned int
Severity                 // 1 digit integer            unsigned int
SevProbVect              // Severity probabilities      double[4]
ReportCountP             // Police                     unsigned int
ReportCountA             // Ambulance                  unsigned int
ReportCountC             // Civilian                   unsigned int
CumAssocProb             // Cumulative Association       double


----------------------------------------------------------------------
```
Ambulance Location

```
SourceDestination        // "EDtoVZ02:Ambulance Location"
Report_Type              // 2 digit integer            unsigned int
Reporter_ID              // 5 digit integer            unsigned long
JurisdictionID           // 1 digit integer            unsigned int
Time                     // seconds                    unsigned long
SrcDstCount              // counter for this Int-name   unsigned long
TrackID                  // recnum * 100 + FileID       unsigned long


Location_X               // UTM – easting value         std::string
Location_Y               // UTM – northing value        std::string


----------------------------------------------------------------------
```
Medical Facility Capacity

```
SourceDestination        // "EDtoVZ03:Medical Facility Capacity"
Report_Type              // 2 digit integer            unsigned int
Reporter_ID              // 5 digit integer            unsigned long
JurisdictionID           // 1 digit integer            unsigned int
Time                     // seconds                    unsigned long
SrcDstCount              // counter for this Int-name   unsigned long
TrackID                  // recnum * 100 + FileID       unsigned long


Severity_1               // 2 digit integer            unsigned int
Severity_2               // 2 digit integer            unsigned int
Severity_3               // 2 digit integer            unsigned int


----------------------------------------------------------------------
```
Cluster Ident

```
SourceDestination        // "EDtoVZ04:Cluster Ident"
Report_Type              // 2 digit integer            unsigned int
Reporter_ID              // 5 digit integer            unsigned long
JurisdictionID           // 1 digit integer            unsigned int
Time                     // seconds                    unsigned long
SrcDstCount              // counter for this Int-name   unsigned long
TrackID                  // recnum * 100 + FileID       unsigned long


ClusterID                // integer                    unsigned int
CellSide                 // length of cell (meters)     unsigned long
CellCount                // number of cells in cluster unsigned int
     CellType            // 1 => Non-boundary           unsigned int
                         // 2 => Boundary
```

```
        CellCenter_X        // X-Coord of cell center     unsigned long


        CellCenter_Y        // Y-Coord of cell center     unsigned long


        Sev2CasCount        // Cas severity 2 Cell Count  unsigned long


        Sev3CasCount        // Cas severity 3 Cell Count  unsigned long
```

Medical Facility Damage

```
SourceDestination          // "EDtoVZ05:Medical Facility Damage"
Report_Type                // 2 digit integer            unsigned int
Reporter_ID                // 5 digit integer            unsigned long
JurisdictionID             // 1 digit integer            unsigned int
Time                       // seconds                     unsigned long
SrcDstCount                // counter for this Int-name   unsigned long
TrackID                    // recnum * 100 + FileID        unsigned long

Location_X                 // UTM – easting value          std::string
Location_Y                 // UTM – northing value         std::string
Facility_ID                // medical facility             std::string
Severity                   // 1 digit integer             unsigned int
```

Roadway Damage

```
SourceDestination          // "EDtoVZ06:Roadway Damage"
Report_Type                // 2 digit integer            unsigned int
Reporter_ID                // 5 digit integer            unsigned long
JurisdictionID             // 1 digit integer            unsigned int
Time                       // seconds                     unsigned long
SrcDstCount                // counter for this Int-name   unsigned long
TrackID                    // recnum * 100 + FileID        unsigned long

BridgeID                   // bridge                       std::string
LinkID                     // road section                 std::string
Severity                   // 1 digit integer             unsigned int
```

Police Location

```
SourceDestination          // "EDtoVZ07:Police Location"
Report_Type                // 2 digit integer            unsigned int
Reporter_ID                // 5 digit integer            unsigned long
JurisdictionID             // 1 digit integer            unsigned int
Time                       // seconds                     unsigned long
SrcDstCount                // counter for this Int-name   unsigned long
TrackID                    // recnum * 100 + FileID        unsigned long
```

```
Location_X                 // UTM – easting value        std::string
Location_Y                 // UTM – northing value       std::string
```

Cluster Ident

```
SourceDestination       // "EDtoVZ08:Cluster Identification"
Report_Type             // 2 digit integer          unsigned int
Reporter_ID             // 5 digit integer          unsigned long
JurisdictionID          // 1 digit integer          unsigned int
Time                    // seconds                  unsigned long
SrcDstCount             // counter for this Int-name unsigned long
TrackID                 // recnum * 100 + FileID     unsigned long

ClusterID               // integer (0-based)         unsigned long
PrevCount               // previously-related clust  unsigned int
PrevClusters            // IDs of prev clusters       vector
     ClusterID          // ID of related cluster      unsigned long
Association             // type = 0 – same            unsigned int
                        //        1 – merge
                        //        2 – split
CellSide                // length of cell (meters)    unsigned long
CellCount               // number of cells in cluster unsigned int
VizCells                // cells in cluster           vector
     CellType           // 1 => Non-boundary          unsigned int
                        // 2 => Boundary
     CellCenter_X       // X-Coord of cell center     unsigned long
     CellCenter_Y       // Y-Coord of cell center     unsigned long
     Sev1CasCount       // Cas severity 1 Cell Count  unsigned long
     Sev2CasCount       // Cas severity 2 Cell Count  unsigned long
     Sev3CasCount       // Cas severity 3 Cell Count  unsigned long
     Sev4CasCount       // Cas severity 4 Cell Count  unsigned long
     AvgSeverity        // average casualty severity  double
```

M – Message   **Source:**   **ReportGenerator**

                 Destination: **DispatcherRouter**

Hospital Location

```
SourceDestination       // "RGtoDP01:Hospital Location"
Report_Type             // 2 digit integer          unsigned int
Reporter_ID             // 5 digit integer          unsigned long
JurisdictionID          // 1 digit integer          unsigned int
Time                    // seconds                  unsigned long
SrcDstCount             // counter for this Int-name unsigned long
TrackID                 // recnum * 100 + FileID     unsigned long

LocCount                // Number of hospLocs        unsigned int
HospLocs:
```

AFOSR F49620-01-1-0371

```
    Location_X           // UTM – easting value      std::string
    Location_Y           // UTM – northing value     std::string
    Hospital_ID          // Medical facility         std::string
    BedCount             // size of facility         std::string
```

N – Message   **Source:**   **ReportGenerator**

                    Destination: **Visualization**

Hospital Location

```
SourceDestination        // "RGtoVZ01:Hospital Location"
Report_Type              // 2 digit integer          unsigned int
Reporter_ID              // 5 digit integer          unsigned long
JurisdictionID           // 1 digit integer          unsigned int
Time                     // seconds                  unsigned long
SrcDstCount              // counter for this Int–name unsigned long
TrackID                  // recnum * 100 + FileID     unsigned long

LocCount                 // Number of hospLocs        unsigned int
HospLocs:
```

```
    Location_X           // UTM – easting value       std::string
    Location_Y           // UTM – northing value      std::string
    Hospital_ID          // Medical facility          std::string
    BedCount             // size of facility          std::string
```

P – Message   **Source:**   **ReportGenerator**

                    Destination: **L2Fusion**

Casualty Counts

```
SourceDestination        // "RGtoL201:CT Casualty Counts"
Report_Type              // 2 digit integer          unsigned int
Reporter_ID              // 5 digit integer          unsigned long
JurisdictionID           // 1 digit integer          unsigned int
Time                     // seconds                  unsigned long
SrcDstCount              // counter for this Int–name unsigned long
TrackID                  // recnum * 100 + FileID     unsigned long

TractCount               // Number of Census Tracts   unsigned int
CTinfo:
```

```
    Tract_ID             // Census Tract ID           std::string
    Severity1Count       // Expected # Casualties     unsigned int
    Severity2Count       // Expected # Casualties     unsigned int
    Severity3Count       // Expected # Casualties     unsigned int
    Severity4Count       // Expected # Casualties     unsigned int
    Severity0Count       // Expected # Casualties     unsigned int
```

AFOSR F49620-01-1-0371

```
Q – Message        Source:         L2Fusion

                   Destination:    EstimateDirector


Cluster Ident

SourceDestination      // "L2toED01:Cluster Identification"
Report_Type            // 2 digit integer          unsigned int
Reporter_ID            // 5 digit integer          unsigned long
JurisdictionID         // 1 digit integer          unsigned int
Time                   // seconds                  unsigned long
SrcDstCount            // counter for this Int-name unsigned long
TrackID                // recnum * 100 + FileID     unsigned long

ClusterID              // integer (0-based)         unsigned long
PrevCount              // previously-related clust  unsigned int
PrevClusters           // IDs of prev clusters      vector
    ClusterID          // ID of related cluster     unsigned long
Association            // type = 0 - same           unsigned int
                       //        1 - merge
                       //        2 - split
CellSide               // length of cell (meters)   unsigned long
CellCount              // number of cells in cluster unsigned int
VizCells               // cells in cluster          vector
    CellType           // 1 => Non-boundary         unsigned int
                       // 2 => Boundary
    CellCenter_X       // X-Coord of cell center    unsigned long
    CellCenter_Y       // Y-Coord of cell center    unsigned long
    Sev1CasCount       // Cas severity 1 Cell Count unsigned long
    Sev2CasCount       // Cas severity 2 Cell Count unsigned long
    Sev3CasCount       // Cas severity 3 Cell Count unsigned long
    Sev4CasCount       // Cas severity 4 Cell Count unsigned long
    AvgSeverity        // average casualty severity double
```

6.3 Work Analysis and domain ontology

Effective high-level data fusion for emergency response demands an intimate integration of domain knowledge, sensor models and their data, goal heirarchies, human perception and judgment. Where in this varying terrain is the fusion system designer to find a foothold? What are useful organizing principles of the design process?

An important observation is that a successful data fusion system should be designed from a user-centric perspective [6.3-1]. However effective technically, any decision support system which is not grounded in the realities of the users, their goals, constraints, and their practices will fail in significant ways. For instance, an emergency communication system covering multiple

AFOSR F49620-01-1-0371

jurisdictions whose design does not pay careful attention to interoperability issues will be of limited value. The difficult part of the interoperability problem may not be technical, but something as basic as the use of the same key term differently in different jurisdictions, or the lack of a common communications protocol among the emergency responders in separate areas. The author recalls hearing an interesting story at a workshop, in which the communications and information processing coordinator of a regional emergency response task force told of his experience with a new multi-million dollar system he had just installed. It was designed to permit police, fire personnel, hazmat teams, radio equipment technicians, city and county field engineers from many distinct jurisdictions, some 80 jurisdictions in all, to talk to one another seamlessly during an emergency. The first time it was turned on, during an incident of multiple wild fires propogating in the area, responders in zones unaffected thusfar made frequent calls to operators in affected areas to find out what was going on. Their curiosity overloaded the system, preventing critical communications from getting through. The system was turned off shortly after its first turn-on. Interoperability without thorough work domain analysis to guide its use can be a detriment rather than a benefit.

A second related observation is that to be user-centric, the requisite domain knowledge must include a thorough understanding of the work environment of the users of the system. The activities and environments that support their goal-seeking determine the essential elements of information that data fusion need supply, and the requisite properties of that information: its required resolution, confidence, and timeliness.

These considerations lead to a emergency response data fusion design philosophy anchored on deployment of an explicit domain ontology, and on the products of that branch of human factors engineering referred to as cognitive work analysis. In the subsections which follow, this approach will be developed.

6.3.1 Cognitive Work Analysis in Information Fusion System Design

Consider the means by which methods in cognitive engineering, namely, work domain analyses, could provide input to the development of advanced information processing, or multisensor information fusion, algorithms. Specifically, a work domain analysis of an emergency management environment (in a post-earthquake context) was performed, and linked abstraction

hierarchy models representing the emergency management and response system, the physical environment (e.g., buildings, transportation systems, civilians), and other goal directed agents (e.g., civilian responders and volunteers) were created. Outputs from that analysis (information requirements) were input to the design of the information processing algorithms, providing guidance as to the nature of information required by decision makers, which could be computed through fusion capabilities. This work thus presents an example of an integrated cognitive engineering/multisensor fusion methodology.

6.3.1.1 Work Domain Models

One focus within cognitive systems engineering is the systematic description of aspects of the work domain comprising the environment in which human operators must act and make decisions. Specifically, models and techniques in work domain analysis have been developed which capture the complexities and constraints of the work domain that serve to shape and constrain the behavior of domain practitioners. Thus, work domain models serve as purposeful, albeit bounded, descriptions of a portion of reality the system of interest. Work domain models, such as the abstraction hierarchy (AH) [6.3-2] , [6.3-3] can be used to identify information relevant to display design (e.g., information requirements and relevant constraints) regardless of which tasks or activities are being carried out within the context of the intended purposes of the system. In AH models, higher levels of abstraction represent the system in terms of its purpose and functions, whereas lower levels represent the system in terms of its physical implementation. An important output from such an analysis is the provision of information requirements for system controllers and decision makers.

6.3.1.2 Multisensor Information Fusion

Multisensor information (data) fusion, is an engineering discipline which uses techniques from signal processing, statistics, numerical methods, and artificial reasoning to formally combine information (numeric data) from numerous, disparate, and uncertain sources (sensors), in order to provide information for higher level reasoning (i.e., human decision making; [6.3-5]). Information processed through data fusion algorithms typically provides an improved estimate (i.e., one with less statistical uncertainty) than can be ascertained from a single sensor or source.

A classic problem in multisensor data fusion is the tracking of a moving object (e.g., an aircraft) based on data coming from multiple sensors such as radar and infrared imaging [6.3- 4] .

A framework for classifying data fusion problems is contained with the data fusion process model created by the Joint Directors of (Military) Laboratories working group on data fusion. Briefly, this process model describes four hierarchical levels of data fusion. Level 1 (L1-"object refinement") processing focuses on producing estimates of an entities position, velocity, attributes, and identity; Level 2 processing (L2-"situation refinement") develops descriptions of relationships among entities and events in context; Level 3 processing (L3-"threat refinement") predicts information about future states; and Level 4 processing provides meta control over the other levels of fusion processing [6.3-5]. Although focused on computational and processing methods, rather than human perception and awareness of such states, the first three levels have a loose correspondence to the three levels of situation awareness described by [6.3-6]: perception of the elements in the environment, including the status, attributes, and dynamics of relevant elements (Level 1 SA); comprehension of the current situation, through synthesis of disparate level 1 elements in terms of their relationship to operator goals (Level 2 SA); and projection of the future actions or states of the elements (Level 3 SA). Certainly, the products of fusion processes can provide operators with information relevant to multiple levels of situation assessment.

Within the field of data fusion, L1 algorithms have been well studied, particularly within the military domain. However, challenges in developing higher level algorithms which can exploit the estimates produced by L1 algorithms still remain, particularly in identifying the parameters which define a "situation" of interest, and are considered key problems for the field. There is current interest in developing processing algorithms which address the challenge of L2 fusion.

Our current project has explored the means by which modeling techniques such as work domain analysis can be used as input to the development of L2 and L3 fusion algorithms. One approach to the development of such algorithms is to formally study and define the nature of situational and impact estimates, based on normative theories regarding what constitutes a "situation." A complementary approach would be to seek within a given domain of interest for information needs which can be nominated as potential higher level fusion problems. Aspects of these

information needs, as well as data fusion algorithms developed to produce the implied estimates, could then be inspected to identify classes of higher level fusion problems and algorithms, which could be understood within a normative framework of situations.

In our project, formal methods in ontological reasoning (see Section 6.3.3 below) are being applied with the goal of producing logically defensible situational constructs. An ontology is a logically structured, conceptual representation of all independently existent items (e.g., physical objects, relations, processes, events) that make up the fabric of reality. At the same time, work domain models are being constructed in order to understand the information needs of decision makers, including needs which could be classified as Level 2 fusion estimates. While the methodologies and contributions of the ontological modeling are beyond the scope of this report, it should be noted that concepts identified in the ontological modeling are being mapped onto concepts identified in the work domain modeling to insure consistency across and within the two modeling constructs. The goals of the work domain modeling are thus primarily to indicate what information products needed to be created, rather than to directly influence display design. This outcome is consistent with recommendations regarding the use of work domain modeling in supporting the identification of information that needs to be obtained, or sensed. The remainder of the report documents the results to date of this approach.

6.3.2 Case Study

The potential role of work domain analysis in informing the development of L2 estimates were explored within the context of a multi-year applied project in the emergency management domain. Essentially, a post-disaster emergency management for an earthquake disaster context was used as a complex and realistic testbed within which a variety of multisource data fusion challenges could be addressed, including the development of methodologies and algorithms related to L2 fusion.

6.3.2.1 The Emergency Management Domain

Post-disaster emergency management is a complex environment poses challenges both to human decision makers which operate within it, as well as attempts to provide automated support

through methodologies such as information fusion. In particular, the following complexities are present:

1. Noisy and uncertain dynamic environment with insufficient *a priori* statistical information

2. Geographically distributed damage

3. Large amount of heterogeneous information

4. Resource and time constraints

5. High cost of error

6. Multiple decision makers with multiple goals and information requirements

7. Heterarchic-hierarchic organization of decision makers

8. Multiple agencies in multiple jurisdictions, at different hierarchical levels (e.g., federal, state, county, locality)

Emergency management and response is an environment in which teams of operators in the field (e.g, fire fighters, building inspectors) utilize skills and local resources to address problems (e.g., trapped civilians, traffic re-direction), and provide status information as well as resource requests to higher level operational response centers (e.g., fire department or city emergency operations centers). The role of these higher level centers is to monitor the current state of the situation, coordinate and provide requested resources, and plan for future resource needs and overall action plans. In the event that local (e.g., city) resources are or will likely be unable to meet the needs of those in the field, requests are made (based on mutual aid agreements and other disaster management agreements and plans) to other municipalities (e.g., other city fire departments, the county). Likewise, if county resources are over committed, requests are made to the state, and so on.

6.3.2.2 Work Domain Model

The work domain model was constructed based on a variety of information sources, including documents detailing emergency management plans; accounts based on data collected during and following two major earthquakes in California; interviews with emergency management personnel at various jurisdictional levels in California (state, county, city) as well as an official with the Federal Emergency Management Agency; and observations of a full scale emergency management exercise at the city level (conducted over four hours in a fully staffed, city emergency operations center). Nodes in the work domain model were annotated with citations, quotes, and explanatory notes based on the information sources used in their development, to support traceability and justification of the model. In emergency management, as in many complex systems, there are multiple domains of interest to be modeled. Similar to work domain models in of military domains [6.3-7], there are domains to be represented: the controlled system (e.g., that of emergency management and response), the environment (e.g., the earth, weather, building, civilians); and other intentional agents (in this case, civilian responders and volunteers who are only loosely linked to the emergency management system). As with most published work domain models, five levels of abstraction were represented. For the emergency management system, at the highest level, goals such as casualty management (recovery of injured, and prevention of further injuries) were represented. At the level of abstract function, balances of economic resources, physical resources, authority, and risk vs. benefit were represented. Nodes at the general function level represented the emergency management functions of planning and intelligence; operations; logistics; management, and finance. Nodes at the physical function level included entities which support the generalized functions, both mobile (e.g., police cars, fire engines, locations), fixed location (e.g., hospitals, dispatch centers), and human (physicians, building inspectors). Nodes at the physical form level represent the physical attributes of these entities (e.g., their locations, operational states, and capacities).

For the environment model, no functional purpose was modeled. At the abstract function level, laws of physics relevant to earth shaking and building stability would be represented. At the general function level, processes of earthquakes, structural stability/collapse, as well as weather processes, and combustion were noted. At the physical function level, nodes represented the functioning of the civil, transportation, and communications infrastructures, as well as the population, earth, and atmosphere. Finally, at the physical form level nodes included the location and operation status of communication systems, building, utility systems, and transportation

networks; location and status hazardous materials, location and state of the civilian population, and the state of geographical features (e.g., hillsides, fault lines) & vegetation; and the wind, humidity, moisture, and temperature conditions. Interactions between nodes at the level of physical form (both within and across domains) were modeled explicitly as links between the nodes. For instance, there was a link between the environmental node "state of utility systems" and the emergency responses system node "location, state, and load on fixed resources" because the capacity and capabilities of hospitals can be impacted by the operational status of water and electricity [6.3-8].

Modeling Outcomes

Information needs were identified and associated with nodes and interactions in the work domain model. For instance, information needs associated with the physical form node "mobile human resources" correspond to the locations, assignments, and instance specific capabilities of the resources (e.g., the location and response status of an ambulance, information regarding its ability to care for pediatric patients). At the abstract function level, information was required regarding the state of resource balances (e.g., for medical, fire/rescue, and police resources). While many of the needs identified correspond more closely to L1 concepts (e.g., the location and severity of casualties) other needs identified could be classified as L2 or L3 fusion problems, because they involve interpretation and projection of data. Examples of these information needs identified to date include:

1. Regions of causalities with input both from casualty reports and predictive models based on earthquake parameters, time of day, and building construction types

2. Risks for secondary hazards (e.g., hazardous materials spills, fires) based on utility locations, earthquake parameters & damage/spill reports

3. Areas of impeded transportation based on predicted traffic patterns, road configurations, earthquake parameters, and damage reports

4. Resource balance assessments (e.g., available vs. potentially required medical personnel, building inspection teams, search and rescue equipment) based on predicted needs and availabilities within each operational area

These information needs are being provided to researchers involved with the design of the L2 algorithms.

In summary, methodologies in multisource fusion may provide the means to produce important information for the consumption of human decision makers. Particularly, L2 and L3 fusion processes can support processes of higher level situation awareness, if the needs of human decision makers, and the outputs from the fusion processes, can be aligned. Methodologies in cognitive work analysis, namely, work domain analysis, may provide a means for systematically identifying such information needs which can then be supplied through fusion processes.

6.3.3 Disaster Ontology

As noted in the preceeding section, the codex for assessing user goals, needs and practices is built on an understanding of the work dynamics and the domain ontology. Having discussed the use of work domain analysis, we turn our attention to the construction of an effective and efficient ontology. This work was aimed at producing a metaphysically-based ontology for improved understanding of post-earthquake disaster environments, with extended applications to other kinds of urban disaster environments containing significant numbers of casualties (e.g., terrorist attacks and conventional/unconventional urban warfare activities). Significant attention has been paid to designing the ontology's uppermost levels as well as domain-specific (i.e., lower) levels in order to produce the framework for an overarching model of disaster environments, which can positively impact on the functions of decision-makers who are observing and managing those environments.

In particular, our attention has been focused on methods for using ontologies to detect and model the dynamic properties of casualty clusters and their relations to other items in the environment such as the earthquake event itself, hospital and ambulatory services, building, road and bridge damage, and tertiary disaster events. Given the daunting complexity of earthquake disaster environments, it was necessary to focus our methodologies on items such as these, in order to provide a manageable problem space within which to work.

6.3.3.1 Existing Ontology Tools and Their Applicability to Higher-Level Fusion Processing in Disaster Environments.

This section will discuss the general issues surrounding ontology construction for improved situation and threat assessment (STA) of complex states of affairs such as disaster environments. The section will address current ontology tools being used for situation assessment and some of the concerns and pitfalls these tools must overcome to provide a sufficient framework for reasoning about disaster environments. In doing so, we will begin to consider certain theoretical and methodological issues that are important in ontology construction for STA applications.

Situation and Threat Assessment (STA).

The purpose of situation and threat assessment (STA) processing in higher-level fusion applications is to infer and approximate the critical characteristics of the state of an uncertain environment in relation to particular goals and information requirements of decision-makers [6.3-22] . The process of building current and predicted situational pictures involves reasoning about various relationships between objects of interest within a particular context. Some of these relations will be relatively simplistic in nature (e.g., spatial relations such as 'next to', 'located at', 'near') and can therefore be handled appropriately using L1 techniques associated with common metrics for object identification, object location, individual object tracking, etc. Ontological modeling at this level of fusion has been quite successful, since one is primarily concerned with individuated and discreet objects, whose properties and behaviors can be measured and understood with relative ease and utilized through existing technologies [6.3-5].

However, other kinds of threat items, especially those kinds of natural and man-made threats important for situation and threat assessment processing, will be complex in nature and will therefore require a sufficiently complex ontological model to understand their composition, organization, inter- and intra-relationships, behaviors, temporal unfoldings, etc. It is here where the focus of fusion processing shifts from *observing* discreet objects and their immediately discernable properties and behaviors, at L1, to *inferring* relations between these objects, which are seldom immediately observable, but rather require understanding a plethora of abstract and concrete relationships that exist between L1 objects.

In earthquake environments, for example, it will be of use to understand basic L1 items such as where discreet items such as casualties, hospitals, ambulances, police, and fire/rescue personnel are located, how they are moving, what they are reporting, etc. It is equally important, however,

to understand related information that must be inferred from the knowledge of these discreet items. Important questions necessary for disaster management can include examples such as:

1. Where do emergency personnel need to be dispatched in the near future?

2. What is the status of related disaster victims, such as casualty clusters?

3. Is there inferential evidence of tertiary disasters in a given area?

4. Can one predict certain kinds of injuries/fatalities due to certain features of the disaster environment such as damage to certain building/bridge/road types due to features such as their construction materials, time of day of the event, or other extenuating circumstances?

In these cases, it is not enough to understand discreet objects, their particular properties and behaviors. Instead, it is necessary to understand their existence as members of collections or wholes, which often possess a certain structured set of characteristics, leading to the parts or members to act in concert with one another and share spatial, temporal, intentional, or causal relations with one another, to name a few. Ontologies are capable of providing an understanding of items as *relational entities*, which amount to the general subject matter of higher-level fusion processing.

Ontology and STA.

Understanding the complexities of disaster environments amounts to understanding complex kinds of relations including spatial relations, temporal relations, causal relations, etc. The value of ontologies (particularly of the realist ilk) is that, if constructed accurately and consistently, they can provide a priori information on complex relational states of affairs in the world, by providing a logically structured model of normal, or known, portions of reality as they stand to one another. In turn, an ontological model of a given domain would then provide a backdrop for reasoning within dynamic and uncertain situations where knowledge is imperfect, items are unknown, reports are unclear or unreliable. The framework of an ontology can then be utilized by reasoning systems to produce new information from a situation by comparing purported epistemological states of the world *as they appear at a given time, under varying conditions of*

uncertainty, with the ontological description of the world *as it stands independent of the epistemic constraints of the given environment*.

Information fusion systems represent the world via several distinct levels of information processing aimed at understanding items such as: individuated objects (as bare particulars), collections of objects, relations between objects, and psychological (i.e., intentional, goal-directed) states of agents responsible for subsequent behaviors and activities (see Fig. 1), all of which, in turn, correspond to different levels of *ontological granularity*, meaning the distinctions
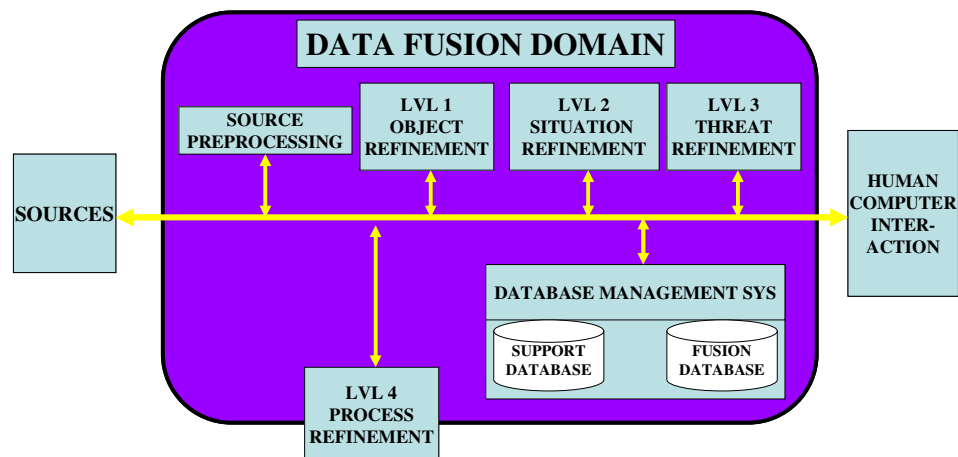
Figure 1: JDL Fusion Model

between coarse- or fine-grained levels of reality, including contextual features. Ontologies provide a formal way of capturing the kinds of informal everyday items over which fusion systems can subsequently reason. Examples include the distinctions between existentially independent physical objects, existentially dependent non-physical objects (attributes, properties), spatial relations, temporal relations, and the like.

Because the items within higher-level fusion (and situation assessment) are relational by nature

Taxonomy of Relations

Relation
- Internal — Identity
- External — Founded / Unfounded
- Grounded — Logical Relations
- Intentional
  - Real
    - Presentational — Satisfied / Not Satisfied
    - Representational — Satisfied / Not Satisfied
  - Fictional
- Existential
  - Mutually Dependent
    - A & B Constitute Each Other
    - A & B are Founded on Each Other
  - 1-Sided Dependence
    - A is Constituted by B
    - A is Founded on B
  - Efficient Causality

(i.e., are not

Figure 2: Relation-Types of Interest for Higher Level Fusion.

singular, discreet units), it is increasingly important to perform a rigorous ontological analysis of the kinds of entities, entity attributes, events, behaviors, and contextual settings which compose the subject matter of situation assessment. situation assessment can be enhanced through an ontological decomposition of threats that includes: designing a meta-model for threat, one which captures the integrated part-whole relations between its components, understanding the distinctions between *viable* (i.e., real, immediate) and *potential* (i.e., possible, contingent) threats, and understanding the ways in which *vulnerability* is related to threat components [6.3-25], [6.3-28], [6.3-36].

An ontology of situation assessment must be able to present a description and elucidation of complex relation-types needed for reasoning about the kinds of dynamic entities treated by higher-level fusion. Significant work is currently being done in this regard [6.3-9], [6.3-22], [6.3-25]. The methodology herein varies somewhat from other research this research, in that the

goal here is to present a metaphysically-based ontology that can describe and enumerate the hosts of relation-types that are found in disaster environments, while not succumbing to the current constraints of given ontology languages such as Protégé, the varieties of OWL, or other such approaches that treat common sense spatio-temporal objects in peculiar and non-common sense ways. Little and Our group has produced a candidate list of such items and their corresponding relationtypes, some of which can be directly perceived via processing of data provided by sensors (e.g., internal and external spatial relations) (See Fig. 1). Other types of more complex items and relations, however, require further inferential processing activities, since they often contain abstract information that is not directly perceivable by sensors.

The taxonomy of relations above provides a glimpse of the kinds of complex relations needed for higher level fusion processing. The categories of 'internal' and 'external' relations can be understood relatively well through L1 processing of discreet items and events, as they encompass information, which can be provided by L1 sensing capabilities that can measure things such as distances between discreet items, their discernable independent (i.e., stand-alone, physical) parts, their dependent attributes/properties (angular trajectory, shape, size), etc. Many of the other relations, however, are not so easily processed, since they involve complex relational information, which must be inferred from a given state of affairs. For example, *intentional* relations, which are those kinds of psychological relations between an agent and their surroundings, which in turn form perceptions, beliefs, goals, etc., are composed of many kinds of nested and intertwined sub-relations between physical and nonphysical components. *Causal* relations present another kind of complex relation, where two items (i.e., a cause and a resulting effect) stand to one another in nested spatio-temporal relations, which must often be inferred based on inferential evidence when there is no directly perceived spatial or temporal connection between items. For these reasons, the ontology must be capable of treating these kinds of items in a thorough and consistent fashion, so as to provide an a priori model of certain basic formal relations between spatial, temporal and spatio-temporal items, which can then be used as the backdrop for a posteriori models, which arise out of, and depend upon, human experience, thereby including uncertainty, perspective, reliability, etc.

6.3.3.2 The Trade-off Between Ontological Languages (Expressivity) and Quality of Inferencing Capabilities (Descriptive Robustness).

A large challenge facing the ontology community centers around the relationship between formal expressivity and descriptive robustness. Ontologies are meant to be comprehensive depictions of reality, including all such necessary items required for an understanding of a given domain. They must therefore contain a plethora of things such as: objects, properties, relations, and events, taken from some particular set of domains. However, a problem arises when one attempts to fit a large, complex ontology into a computational framework designed for producing results expeditiously or within a given logical framework. Given the ways in which many contemporary computational systems work, often build upon description logics or frame-based systems, it can pose significant challenges in capturing the kinds of robust ontological relations found within complex situations such as disaster environments. This results in a problem of fit between ontologies as metaphysical constructs and the computational languages used to express rules, axioms, or propositions of the ontology. Often, one is forced to accept a trade-off between the *complexity and quality of the inferencing* (i.e., both the computational aspects of executing the logic, and the "usual" qualities ascribed to logic systems such as completeness, soundness) and the *expressivity of the formal language used to describe items within the ontology*. For example, the choice of a given species of OWL (Lite, DL, Full) has a direct implication on the underlying nature of the consequent inferencing power that results, because the choice of language expressiveness directly implies a style of inferencing and thus the inherent qualities that come with it.

Considering OWL, there is an obvious trade-off when one moves from OWL Full to OWL-DL, or from OWL-DL to OWL Lite. The restrictions placed on each subsequent sublanguage in OWL are such that one sacrifices the ability to express certain important relations between classes, individuals, and properties for the sake of computational efficiency. Since information used by computational systems exists in, and subsequently has limited interactions with, certain formats (e.g., XML), there is a legacy to those existing systems that cannot be overlooked. Description logics have been developed to quickly and effectively run computational systems in ways that more robust first-order logic systems cannot. However, ontologies have traditionally been developed within the context of those robust first-order systems (e.g., KIF), because they possess the kind of desired formal machinery capable of capturing complex kinds of relationships between things. Transforming that ontological information from first-order logics

into description logics often results in the loss of information pertaining to relation types, deep semantic content, etc.

In essence, this is no small issue, since high-level fusion systems seek to understand situational complexes in robust ways. This means that they would require an ontology capable of providing the kind of robust relational descriptions of the world necessary for actually understanding states of affairs within the real world. Watering down the ontological description of the world for the sake of computational efficiency could prove disastrous for fusion systems, as entire segments of a domain may not be included within such a limited ontological description. Yet, fusion systems also need to produce real-time results for decision-makers. This means that they cannot be bogged down by significant time-delays caused by inefficient computational systems.

Researchers on this project were concerned more with providing a sufficiently robust ontology for high-level fusion design, and less interested in applying weak forms of computational tools which would undercut the sophistication of the ontology just to increase processing speed. Doing so would only result in future problems, since another system would be designed which would be incapable of actually doing higher-level fusion applications, but would, instead, be constrained to more basic L1 problems, such as the Protégé, OWL and SAW examples mentioned above. Large-scale ontologies, like those needed for fusion, should not be designed solely in regards to computational efficiency. Instead of building ontologies under the constraint of today's current computational abilities, we should build them to be accurate depictions of the world, somewhat independent of computational constraints. Doing so can set new standards for the computer science community to invent new logics or computational languages that can capture, and process, the kinds of things, processes, relations, attributes, etc., found in complex metaphysically-based formal ontologies.

6.3.3.3 Amending Existing Tools With A Metaphysically-based Upper Ontology for Higher-Level Fusion Processing in Disaster Environments.

This section will report upon specific approaches taken on this project for designing an ontology model that can more effectively treat the kinds of complex relational items found within disaster environments, ultimately providing for improved higher-level fusion processing and situation assessment.

Basic Formal Ontology.

The ontology constructed for this project is part of a larger research agenda entitled the Basic Formal Ontology (BFO). The BFO is an ongoing research project being conducted at both The University at Buffalo and The Institute for Formal Ontology in Medical Information Sciences (IFOMIS) at the University of Saarbrücken, Germany [6.3-24], [6.3-13], [6.3.14], [6.3-15].

Upper- vs. Domain-specific Ontology.

The BFO represents an approach to building large-scale, reusable ontologies for applications in any domain whatsoever, since it is a metaphysically-based approach to ontology design that attempts to faithfully capture both the physical and phenomenological aspects of the world. In this sense, the BFO is designed from both a top-down as well as a bottom-up approach, ensuring that it is metaphysically comprehensive and consistent, while at the same time, being accurate and computationally tractable at the domain-specific levels. It has been argued by Little [6.3-24] that the formal, upper-ontology levels of an ontology are produced by logical reasoning about the metaphysical structure of the world, whereas, in contrast, the domain-specific levels of an ontology are produced by empirical means taken directly from a particular domain of interest (see Fig. 4). By getting the metaphysics correct at the upper-most levels, the ontology can guarantee certain successes which other computationally-based systems could not, since often times computationally-based systems are initially constrained by a specific logic, rule language, or other such computational concern. While computational matters are very important for implementation, it is equally important that the ontology be able to capture all of the relevant data supplied by the domain, and not be overly constrained in its ability to categorize or conceive of a situation, based on the limitations of the system's capabilities. This topic will be covered in more detail later in this report. By getting the domain-specific levels of the ontology correct, one is guaranteed an accurate representation of a particular domain based upon domain expertise in a given field. By conjoining the upper and domain-specific levels, one is able to construct a comprehensive ontology that is both logically, as well as empirically, comprehensive and consistent.
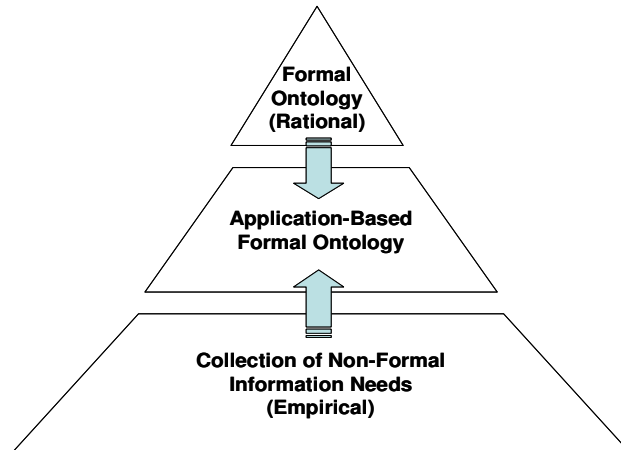
Figure 3: Combining Formal & Domain-Specific Ontological Levels [6.3-26]

SNAP and SPAN.

The BFO is composed of two orthogonally-related sub-ontologies called SNAP and SPAN. SNAP is used to represent spatial objects, independent of their temporal characteristics or attributes (see Fig. 5). SPAN is used to represent temporal objects, independent of their spatial characteristics. SNAP ontological entities represent items that are continuants, meaning they endure over time and maintain their identity, in spite of changes. Examples of such items are: a person whose cellular structure undergoes numerous changes, a body of water whose shoreline can grow or shrink, a nation whose members are continuously gained and lost. At any given point in time, one can gain a "snapshot" of such an object, or group of objects, that exhibits all of its spatial properties in one go. All such items will appear as complex existent (and identifiable) entities, complete with their static properties and numerous kinds of spatial relations (e.g., next to, at location 'x', possessing size 'y', etc.).

Conversely, SPAN items exist as occurrents, meaning they occur only within time, or exist as items that temporally unfold over time (see Fig. 6). Examples of such items are: the function of a person gaining or losing their cells over time, the expansion or contraction of a body of water's shoreline over time, the gaining or losing of members within a nation. SPAN entities cannot be understood in a "snapshot" approach, since at no given point in time are all of its attributes or properties present. The attributes or properties of SPAN items exist over numerous time frames (See Fig. 4).
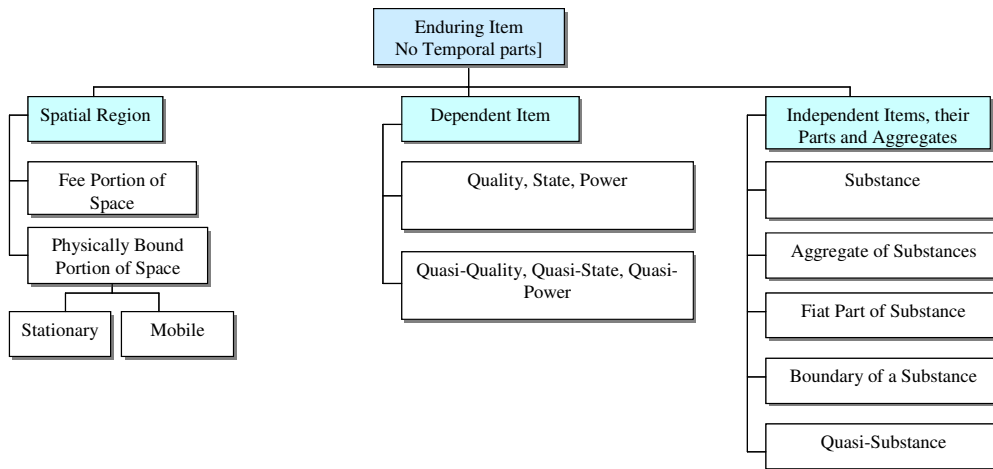
Figure 4: SNAP BFO Upper Ontology Model.

SNAP and SPAN represent an artificial way of parsing the world, since all real objects in the world, particularly those of interest to multisensor information fusion systems, exist as spatio-temporal items. SNAP and SPAN were designed to avoid errors in modeling complex spatio-temporal items, where entities and processes can become confused, resulting in an improper and fallacious formal model of part-whole relations. Confusing entities and processes can result in improper ontological categorization, which in turn, leads to poor inferencing capabilities in corresponding knowledge representation (KR) systems or in ontological queries. An example of the distinction between SNAP and SPAN is between an object and its function.
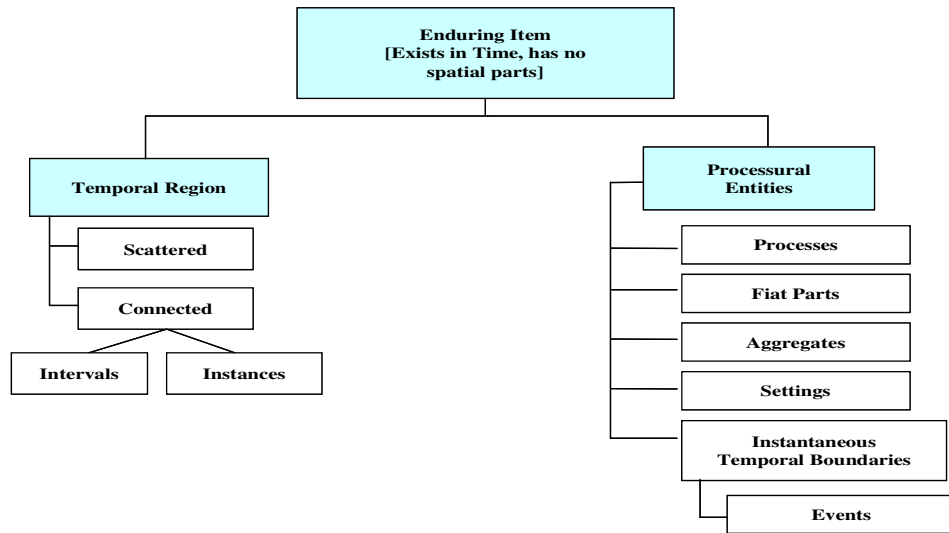
Figure 5: SPAN Upper-Ontology in the BFO

6.3.3.4 Six-Step Methodology for Ontology Construction.

A six-step methodology has previously been outlined by Little [6.3-24]. It is a methodology which addresses ontology construction from both the upper- (i.e., top-down) and the domain-specific (i.e., bottom-up) approach. The six steps are as follows:

Step 1. Develop a sufficiently large and representative lexicon of terms.

Step 2. Develop a set of metaphysically-grounded upper-level (abstract) categories (SNAP & SPAN).

Step 3. Develop a sufficiently large set of domain-specific (lower-level) categories under 2.

Step 4. Diagram complex formal relations between SNAP-SPAN terms/categories.

Step 5. Develop an ontology management tool for knowledge representation and reasoning (KRR).

Step 6. Examine methods for ontology evaluation.

Step 1 involves finding and structuring lexical data from a given domain of interest. This step involves a careful examination of relevant literature, parsing out any and all terms relevant to the description of that domain. All terms are alphabetized into a domain-specific lexicon, which will provide the terminology for the domain-specific construction of the ontology's lower levels. Definitions are then further dissected in order to extract all relevant terms contained within each one. These terms are then divided (and normally color-coded) into SNAP-specific, SPAN-specific and SNAP-SPAN-relational items. To date, this work has been done by hand, but current thought is being given to using text-mining software to facilitate easier construction of the domain-specific lexicon. The issue concerning the use of automated text-mining tools at this phase is that it is unclear how effective these tools will be in capturing all of the relevant terms. One must be cautious that the lexicon's construction, which ultimately winds up representing the domain-ontological terminology, does not rest on faulty ontological assumptions within the text-mining software itself. The lists generated by text-mining tools, at least in the initial trials, would have to be checked by hand to ensure that all relevant terms were captured and subsequently placed into appropriate SNAP and SPAN categories.

Step 2 is a philosophical exercise that involves the general construction of the BFO's upper-level segment. This is an on-going research agenda that involves a lot of reasoning on the part of the ontologist to ensure a proper upper-ontology that is capable of categorizing all of the necessary information within the ontology according to logical laws.

Step 3 amounts to applying the lexicon from step 1 to the upper-ontology segment designed in step 2. Many of the items within the lexicon will share certain class, sub-class, and attribute relations which can be gleaned from their definitions. By applying the lexical items to the BFO's upper-ontology segment, one can build a-cyclical species trees for both SNAP and SPAN items independent of one another.

Step 4 involves drawing transcategorical relations between the independent SNAP and SPAN species trees (which again represent the orthogonal nature of the BFO's ontological structure). These relation-types will represent more complex relation-types than can be shown within the structure of a tree diagram, since tree diagrams normally can only present simple relations such

as is-a or part_of. A list of these kinds of more complex transcategorical relations will be described below.

Step 5 is an implementation-minded step, which involves the trade-off of ontological robustness versus computational tractability. While this topic will be discussed more thoroughly later in this report, it is worth noting the importance of this issue in the ontology's over-all development. Too often in the computer science community, ontologies are constrained by the computational limits of current ontology development tools (e.g., work on structured vocabularies, frame-based or description logic systems, etc.), resulting in ontologies that do not perform adequately in terms of their abilities to structure and draw connections between numerous kinds of complex relational items, such as those in disaster environments. The methodology described here argues that the ontology should be designed with an eye to the state of the art in terms of software applications, but at the same time, the ontology's construction should also be carried out in line with a certain metaphysical and logical robustness, perhaps exceeding immediate implementation in existing software tools. In this sense, the ontology can be a quality theoretical product, which would serve to prompt new and innovative design methodologies in terms of its application in a computationally-tractable software product. This approach seeks to form a more synergistic and productive relationship between the community of formal ontologists and the community of computer scientists than currently exists.

Step 6 is perhaps one of the most challenging steps to fulfill, since it is tied to the question of: "how does one know when they have a quality, or even useful, ontology product?". This question could be approached form numerous angles, but the idea in this methodology is to approach the issue from both the rational as well as empirical position. A good ontology on the rational level would admit of consistency, metaphysical robustness, sound theoretical (philosophical) structure, and a proper logical framework (set theory, mereology, topology, mereotopology, etc.). A good ontology on the empirical level would be the product of testing the system in experimental applications, where one could produce higher-level fusion state estimates for a specified task both with and without the ontology. If the thesis that ontologies are useful for higher-level fusion is correct, there should be a significant gain in both the construction, and implementation, of fusion algorithms for state estimation, belief revision, and the like. It has been argued [6.3-24] that an ontology's success or failure should be determined within the

guidelines of traditional systems engineering methodologies, where there are sets of feedback loops set up from the initial design phase, through the prototyping and construction phase, to termination and disposal. In this sense, the design of the ontology is scrutinized throughout both its development and implementation, leading to an evolving and increasingly better end product in terms of both its internal design, as well as it external (i.e., implemented) behaviors.

6.3.3.5 Inter- and Intra-Relations Needed for Domain Specificity.

The SNAP/SPAN Basic Formal Ontology constructed for this project required a significant amount of research into not only the various kinds of relation-types mentioned in Fig. 2, but the order and structure of those basic relation-types in terms of the kinds of nested sub-relations contained within them. As stated earlier, other ontology attempts such as the SAW Ontology do not appropriately partition their relation-types into suitable sub-classes, capable of treating the kinds of complex situated items in disaster environments. Each relationship characterizing a situation falls into one of two basic categories: inter-class relations and intra-class relations (see Figure 8). Intra-class relations can be thought of as a subclass of internal relations whereas inter-class relations can be thought of as a subclass of external relations (see Fig. 8).   In this sense, we are especially concerned with both the part-relations that exist within a given item or set of items as well as the part-relations that exist between various items, since these kinds of relations will be crucial to proper reasoning about items such as casualty clusters, whose part-relations, members, and spatio-temporal attributes can change over time.
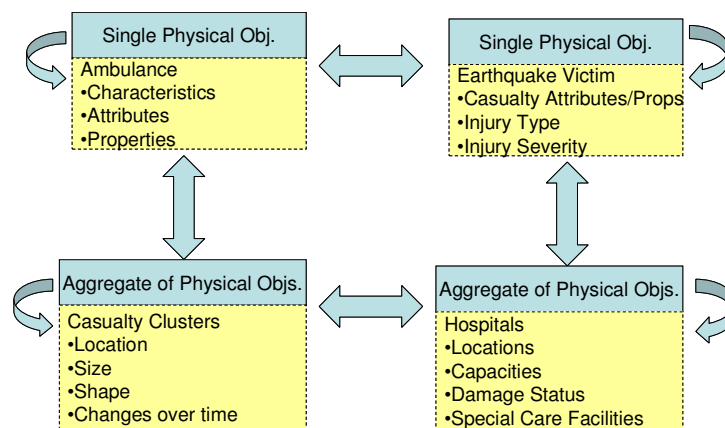

Figure 6: Inter- and Intra-Relations in Dis-ReO.

Intra-relations (i.e., internal relations) are spatial, temporal, or functional relations that exist between the following:

1. physical objects and their respective attributes

2. various attributes of the same object or set of objects

3. individual physical objects and their overarching aggregates

4. sub-groups of related physical items of the same aggregate at different levels of granularity (i.e., macro or micro considerations)

5. aggregates of related events.

Inter-class relations (i.e., external relations) are spatial, temporal, or functional relations that exist between the following:

1. individuated objects of different types

2. individuated objects and aggregates of different types

3. individuated aggregates of different types at the same level of granularity

4. individuated aggregates of different types at different levels of granularity.

The most basic situations can be treated as collections of context-dependent relations between physical items within the same category (e.g., casualties, buildings, ambulances, etc.) or between similar temporal events of the same category (e.g., settings, time periods, discrete events). These basic situations can be defined as either aggregates (clusters), or as wholes, depending upon their metaphysical structure (i.e., the specific ordering of their parts). Structured items, whose part-relations form certain metaphysical connections such as dependence (either one-sided or reciprocal) represent wholes , which can possess an intrinsic put-togethered-ness, thereby resisting conventional modeling as sets (whose members do not form such intrinsic connections). Sets of items represent loose aggregates, where the intra-part relations are not necessary for the

existence of the aggregate. Lists, piles, or collocations of items, which possess no inherent part-structure, but merely co-exist in some space or time, are examples of these. An understanding of basic situations and their aggregates can be obtained by applying a similarity metric in the feature space. The types of features used for aggregation depend on the information needs of a certain user or a group of users. At each subsequent level of granularity, situations are localized situations described by either a clique of aggregates, or more simply by a set of aggregates in a
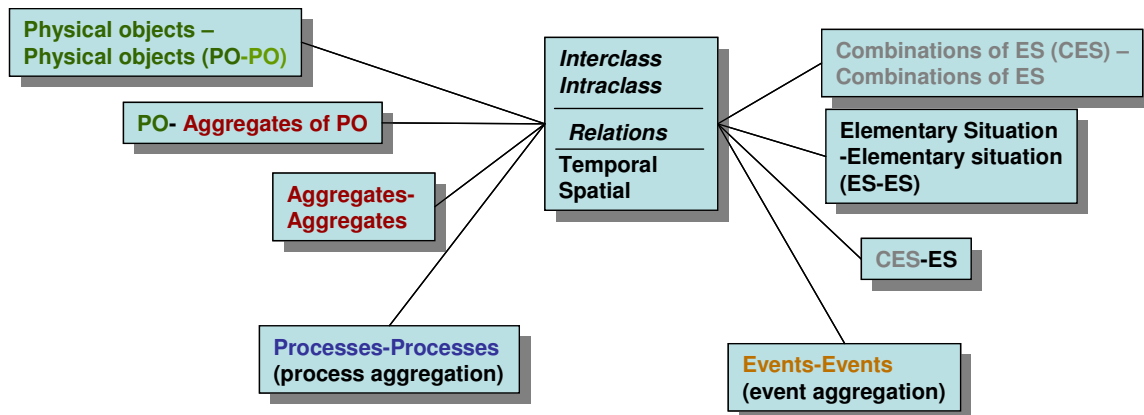
certain region. Events related to aggregates are represented by a significant change of the parameters of the aggregates, discovery of a new aggregate, or the splitting/merging of aggregates at a higher level of granularity.

Figure 7: Individuals and Aggregates of Various Kinds.

Derived intra-class situations are created by a composition of basic intra-class situations at specific levels of granularity. These are called elementary situations. Among elementary situations to be considered are:

1. Communication system situations (capacity vs. demand, location, boundary, dynamics, possible causes of the problems, predicted problems)

2. Transportation system situations,

3. hazmat situations (secondary threat, location, type, dynamics),

4. Casualty situations (location, boundary, severity, injury types, dynamics and causes)

5. Hospital situations (capacity, accessibility, possible damage, capacity prediction)

6. Building situations (location, level of damage, predicted damage)

7. Ambulance situations (location, capacity vs. demand)

Relations between spatial aggregates at various levels of granularity (SNAP relationships) are represented by the mereological categories of direction, size, and distance. Relations between events and processes (SPAN relationships) are defined by time-point and time-interval relationships (Tables 1 and 2). Examples of such relations in disaster situation assessment are:

1. Close to a hospital,

2. Cluster A is larger then before,

3. Cluster B is along the west wind direction

4. Distance between Clusters A and B is smaller than before,

5. Casualty cluster A overlaps with building cluster C.

Table 1. SPAN relations

Relation between time points	Before, At the same time, Start, Finish, Soon, Very soon, Resulting in, Initiating, value of time interval
Relation between time intervals	Disjoint, Joint, Overlap, Inside, Equal

Table 2. SNAP relations

Topology/ mereology	Direction	Size	Distance
Disjoint Joint Overlap Cover Reachable Unreachable Contain A part of	Along Towards East West South North Similar Opposite	Smaller Larger size difference	Not far Far Very far Close Very close distance between clusters centroids

Figure 8: Domain-Specific SNAP and SPAN Relations for Disaster Evvironments.

6.3.4 Ontology and Cognitive Work Analysis (CWA).

Determining the kinds of domain-specificity needed for a quality fusion ontology rests on meeting the needs of decision-makers in terms of their goals and the allocations of tasks/resources to meet those goals. For this reason, a significant portion of work was performed to merge the Dis-ReO Upper-Ontology with an Abstraction Hierarchy, which is a task-related hierarchical model used by cognitive systems engineers to structure information relevant to domain-specific tasks carried out by domain experts and other key personnel working within disaster environments. The merger of the ontology with CWA considerations allowed for an enhancement of both the ontology and the given abstraction hierarchy, in that ontology served as a means to formally categorize elements in the abstraction hierarchy, while the abstraction hierarchy served to provide a consistent and comprehensive amount of domain-specific information relevant to disaster environments. An exemplary segment of the conjoined model

provided for a large categorical structuring of information, which can be further decomposed and utilized for extended understandings of disaster domains (see Fig. 11).
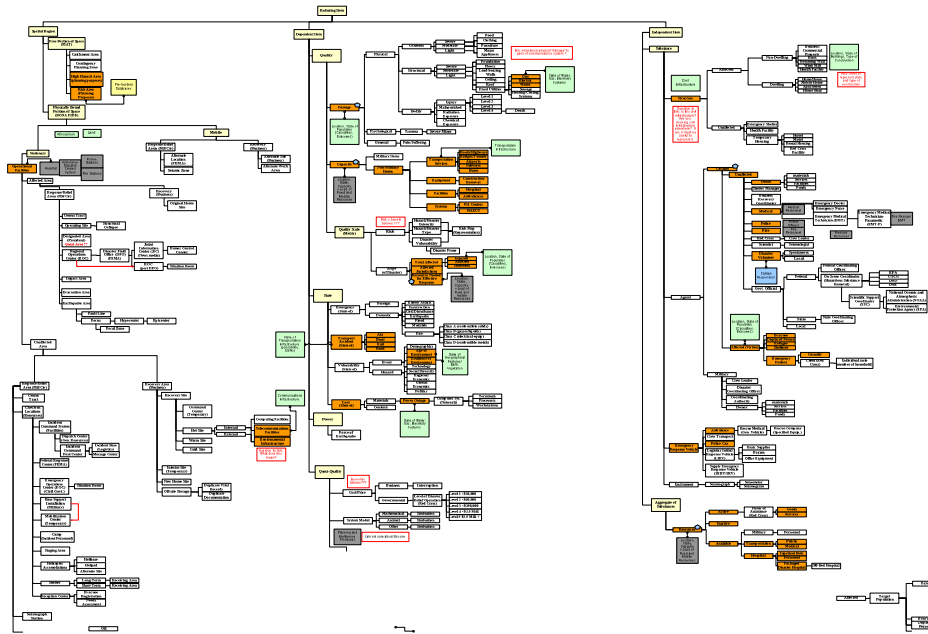
Figure 9: Small Segment of SNAP Dis-ReO Merged With CWA.

This research improves even further on the complexity of relation-types between items in CWA's (yellow boxes in Fig. 11) and the ontology. Utilizing this model within an appropriately sophisticated KRR tool, as previously discussed, would allow for not only an understanding of the disaster domain in terms of objects, object attributes, processes, etc., but also the functions and activities of human agents operating in a goal-directed manner within that environment. More work needs to be done on this to implement it into a software tool for disaster personnel management, but indications appear favorable that this approach will provide for ever improved ontological capabilities for treating the kinds of complex items (including those of human intentions and tasks) involved within disaster environments.

6.4 Hospital Modeling

The principal interest in this project is to explore the role of, and define suitable methods of, data fusion in emergency response. A simulation test bed named DIRE was designed and

AFOSR F49620-01-1-0371

implemented to support this purpose, and is described in Section 6.2 of this report. In order to define MOP's and MOE's by which to measure the efficacy of various data fusion choices and sensitivity to variations of simulated ground truth parameters, it is necessary to implement certain activities within DIRE which do not directly contribute to data fusion per se, but use the products of data fusion to mitigate casualties and achieve other goals related to emergency response.

One such activity is the routing to appropriate hospitals of casualties which data fusion determines to require hospital care. This is a resource management rather than a data fusion activity, one which permits metrics such as casualty service time and fraction of truly critical casualties served to be measured. In order to choose the appropriate hospital to which to route a given patient who has been picked up by an ambulance, the ambulance routing agent should anticipate the ability of each candidate hospital to serve a new patient of the given injury type at the time the patient would be arriving, as well as the expected transit time. The ability of a hospital to serve this patient depends on several factors, including the current level of hospital capacity utilization and a dynamic model for how this will evolve between now and the time the patient in question could arrive there. This is the hospital modeling problem.

In this section we develop a generic hospital simulation model that is capable of representing the operations of a wide range of hospitals in an earthquake disaster situation. From results of our simulations, generalized regression equations are fitted to obtain steady-state hospital capacities. A parametric metamodel is then developed to predict transient capacity for multiple hospitals in the disaster area in a timely manner, as demanded by emergency operations management. Given transient capacity predictions for each hospital, the routing agent can select an appropriate target hospital.

6.4.1 Functioning of the hospital in a disaster

Following a disaster, a hospital emergency room (ER) can expect an increase of three to five times the normal patient volume [6.4-15]. This could easily overwhelm the hospital resources. Hospitals should thus be prepared. As vital community resources, hospitals should thus be prepared in terms of the following requirements resulting from a disaster:

1. To have numbers of personnel, including physicians, registered nurses and other practitioners, that are sufficient to meet resulting needs for emergency care.

2. To meet the sudden surge of emergency patients with temporary additional capacity. (This requires integration of emergency services with other departments of the hospital. For example, in a disaster situation, it is common to convert some available in-patient areas, and even hallways, to ERs, and to use labs in other departments to test ER patients. Close coordination within the hospital will help in the timely treatment of large numbers of new patients.)

3. To conduct resource planning and coordination between the emergency operation center (EOC, normally set up and operated by federal, state and local emergency management agencies) and hospitals in the disaster area.

4. To continually treat those patients who are already under care prior to the disaster.

In order to meet these requirements, well-coordinated relief efforts, in addition to emergency preparedness, are essential functions.

6.4.2 Hospital capacity estimates

Although we cannot predict a disaster with appreciable certainty, emergency preparedness is essential to minimize resulting damage. In a disaster, the EOC generally supervises relief operations. If the damage can be predicted, or estimated immediately after the disaster, relief efforts can be planned and coordinated accordingly. In the case of earthquakes, software tools such as HAZUS [6.4-13] developed by the Federal Emergency Management Agency (FEMA) are helpful in predicting the extent of loss/damage based on geographic location and severity of the earthquake.

Another effort that could greatly assist the EOC is hospital capacity planning. By estimating the available hospital capacity, EOC would be able to make well-informed decisions on where to send patients and how many. Such decisions are based on the proximity of the hospital to the disaster site and its available capacity, as well as the amount of time within which a patient's injuries must be treated. Furthermore, capacity estimates are also useful in the dispatching of

ambulance/rescue vehicles, deploying medical staff, and securing external help and equipment for the hospitals.

Despite the importance of hospital capacity in emergency management, there lacks the research effort. Except for a few reported applications (e.g. [6.4-10], [6.4-21], [6.4-26], [6.4-28]) where simulation was used in hospital studies in general, we found no applications to hospital capacity planning in disaster management. Further, little research has been done to study the hospital functions in a disaster.

Motivated by the needs of hospital capacity estimates in a disaster, this research had the following goals:

1. To model hospital operations in a disaster situation. Since it is important for the EOC to know the status of all hospitals in their disaster area, the proposed model should be capable of representing all such facilities. We use an earthquake as the disaster for this purpose..

2. To develop a methodology for capacity estimates of hospitals. For purposes of capacity estimations, we define hospital capacity as the number of injured patients the hospital can accept in a given time period that the patients must be treated for theire injuries to avoid loss of lives.

A wide range of modeling methods is reported in the literature for representing hospital operations. Deterministic mathematical programming models, such as linear programming (LP) and dynamic programming (DP) were used to optimize resource allocation in hospitals and healthcare systems [6.4-31]. Queuing models can capture the stochastic nature of patient arrivals [6.4-5, 6.4-18]. System dynamics model can describe the dynamic relationships among different hospital sectors [6.4-19]. Discrete event simulations were widely used for modeling the detailed functioning of an individual hospital or a specific section of a hospital [6.4-27], [6.4-29], [6.4-44]. In addition, metamodels are useful for describing a set of similar systems and have been used to evaluate hospitals' efficiency [6.4-6].

Almost all existing hospital operational modeling research directed at capacity estimation has focused on bed capacity. Hill Burton [6.4-38] projected five-year bed demand based on population on accupancy factors. Roemer and Shain [6.4-39] concluded that beds beget patients,

in the sense that beds will ultimately be occupied at approximately the same rate in any hospital even if bed number is increased. Trye et al. [6.4-41] constructed a mathematical model for estimating future bed demand based on two years of inpatient data. Mouza [6.4-42] projected hospital bed requirements based on forecasts of the admission rates after accounting for the structure of the admitted population by gender and age.

In the current context, the challenge lies in the real-time estimation of hospital capacity during disasters. In addition, when studying disaster management, it is important to employ a generic model of all hospitals in the disaster area. Clearly, the time-consuming simulation of individual hospitals will not suffice.

6.4.3 Modeling Methodology

Discrete event simulation is a valuable tool for hospital modeling, lengthy execution is required for obtaining statistically meaningful results . Further, in disaster mitigation all available hospitals in the area need to be modeled, which may vary vastly in number and specification. Most importantly, capacity estimates are required near real time to be useful. Real-time simulation runs are thus infeasible.

Another serious challenge in disaster modeling is the sudden surge of patient arrivals after the disaster, rendering the system a transient behavior. As patient arrival rate significantly affects patient flow time. [6.4-10], high arrival rates leads to excessively long waiting due to overwhelmed hospital resources and facilities. A new methodology is thus needed to obtain capacity estimates that are not only transient in nature, but also applicable for all hospitals in real time.

Such a generic simulation can represent any hospital of interest, with a model that varies ER patient volumes, hospital size and operating efficiency. The simulation is run off-line. According to Giraldo et al. [6.4-14], using a factorial simulation experimental design, one can develop parametric models of hospitals by constructing regression equations that relate hospital performance measures to hospital's characteristics, which are the independent variables.

While parametric regressions can model long-term system performance, transient behavior is captured by a metamodel based on system dynamics. The two sets of models are then combined to determine the temporal behavior of the hospital(s), thus allowing for capacity estimations.

We consider patient waiting times as the response variable of interest. Since they represent how busy the system is, allowable waiting times for treating the injuries of various severities, i.e., survivability of the injury, indicate the hospital's available capacity.

6.4.3.1 Characteristics of hospitals for simulation modeling

The hospitals of interest in an earthquake are those that treat all general types of injury and have ERs and operating rooms. Specialty hospitals such as cancer institutes, psychiatric centers, etc., are not seen as significant contributors to the treatment of earthquake related injuries. Only non-specialty hospitals are thus included in this study.

Number of beds

Table 1 shows the statistics for all US hospitals [6.4-1]. Although hospitals with less than 100 beds constitute 47% of all hospitals in numbers, they only account for 17% of ER visits and 12% of surgeries. Furthermore, they have less than 20% of all the beds and staff. Therefore, if we focus on hospitals that have more than 100 beds, we will include more than 80% of hospitals in terms of capacity. After studying more than 50 hospitals randomly selected from different states across the country, we consider a typical large hospital to have about 500 beds, a medium-sized one to have 300 beds, and a small one with about 100 beds. Knowing these facts, we categorize hospitals into three sizes with 100, 300, and 500 beds. Hospitals within the range can be interpolated from the obtained results.

Table 1

National statistics on hospitals for the year 1999

Beds per Hospital	Hospital	Total Beds	ER visits	Surgery	Physicians & Dentists	RN's	LPN's	Other Personnel
6~24	355	6,356	1,007,020	170,545	1,215	6,474	2,054	25,255
25~49	1,018	37,610	5,589,452	873,181	3,615	31,488	9,543	112,799
50~99	1,377	99,296	11,055,589	2,099,193	7,216	75,772	19,868	26,4215
beds<100	47%	14%	17%	12%	12%	11%	20%	13%
100~199	1,472	209,499	25,600,078	5,835,390	16,929	196,314	34,772	604,275
200~299	727	176,426	20,176,833	5,204,488	18,071	186,601	28,786	554,009
300~399	424	146,703	15,190,729	4,182,690	12,723	164,413	19,538	478,683
400~499	193	85,634	7,573,751	2,335,381	8,159	97,179	12,788	283,622
>500	324	232,342	17,590,973	5,564,085	31,028	263,852	31,694	765,688
beds>100	53%	86%	83%	88%	88%	89%	80%	87%

(Compiled from AHA Hospital Statistics, 2001 edition, table 2, pp.4-5)

Number of operating rooms (OR)

In an emergency situation, a patient is expected to go through the ER, any required Lab testing and, if necessary, the OR. To simplify the model, we only included one lab, which is capable of all tests such as CAT scan, MRI, X-Ray, blood tests, etc., instead of modeling individual labs. Since the OR capacity is fixed, OR becomes the most critical resources in an emergency situation. Our study of over 50 hospitals across the country showed that most hospitals have five to 15 ORs. Therefore we chose 5, 10, and 15 OR's to define this characteristic of the hospitals.

OR efficiency

Even for hospitals with the same number of beds and ORs, the number of patients treated varies widely with various degrees of OR efficiency. Given the number of ORs, we can estimate the hospital's surgical capacity by multiplying the number of surgeries per OR. Thus, a logical measure of efficiency is the number of surgeries a hospital can perform per OR per year. The more surgeries an OR can perform, the more efficient the hospital. By comparing the American

Hospital Association (AHA) hospital data for more than 50 hospitals across the country [6.4-1, 6.4-2, 6.4-3], we found that the number of surgeries per OR in a year ranges from 600 to 1200, with an average around 900. We denote this OR efficiency index with a value of 600, 900, or 1200.

By compiling the recent national hospital data [6.4-1], we found in Fig.1 that annual ER visits are directly proportional to the number of beds. Therefore, it is not necessary to specify ER capacity once the number of beds is known.
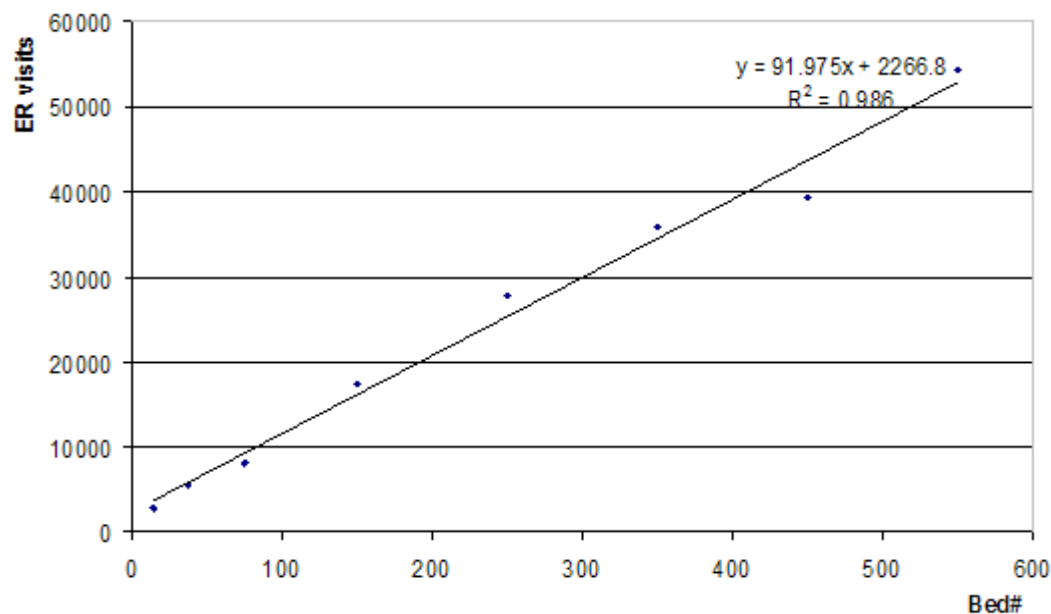
Fig. 1 Relationship between ER visits per year and number of beds

6.4.3.2 Regression approach and metamodel

By assuming a constant patient arrival rate before the earthquake, the system stabilizes to a steady state.. However, increasing the arrival rates to a higher value after the earthquake will result in either a steady state after a certain transient state or system inequilibrium, depending on the patient volume. By using different patient arrival rates in the simulation, we obtained the corresponding post-EQ [EQ = Earthquake] patient waiting times. Regression equations are obtained for these pre-EQ and post-EQ steady-state waiting times. When patient arrival rate is

higher than the service rate, the waiting time will increase continuously in an inequilibrium system. The waiting time will soon exceed the acceptable limits, i.e., the survivability.

Immediately after the earthquake, the system goes through a transient stage to gradually stabilize to the post-EQ steady state. Since we are interested in short-term estimates, the transient state is of utmost importance. In capturing the dynamic behavior of first-order systems, Cochran and Lin [6.4-8] showed that the transient state of a manufacturing system resulting from dynamic events such as machine breakdowns and parts supply shortage can be approximated satisfactorily by an exponential function. We believe the transient state, starting from pre-EQ waiting time to post-EQ waiting time in the hospital due to the sudden patient volume surge in the volume may have a similar exponential behavior due to the similarities of hospitals and manufacturing systems and the fact that both have limited resources. The exact shape of the exponential function depends on hospital parameters and patient arrival rates.

Any earthquake situation is well represented by an initial pre-EQ steady state, the intermediate transient state and the final post-EQ steady state. Combining the steady state regression equations and transient state models, the patient waiting time at any time for any hospital for a given patient arrival rate can be found. The available capacity, then, is indicated by the difference of maximum allowable waiting time, i.e., survivability, and current waiting time. That is, if the waiting time exceeds the survivability, the hospital does not have the necessary capacity to treat the injured patients. Since the patient arrival rate changes during an earthquake, it is necessary to update the arrival rate periodically. Therefore, the capacity estimate is dynamically updated in specified time intervals based on the most recent arrival update.

6.4.4 A generic simulation model of hospitals

Using the simulation software ProModel, we developed a generic hospital model with the partial factorial design described earlier. In a disaster situation, all staff will be called on duty. The efficiency in ER/Lab tends to improve and mostly lab tests of a preliminary nature for faster result will be used. Labs from other departments may also be used. Therefore, the model assumes that Lab and ER do npt restrict capacity. Similarly, human resource and equipment constraints are not considered in this study.

We performed the initial set of simulation experiments with a large number of replications (300 replications) to achieve small variances so that we could validate our model. Then a power analysis [6.4-46] was used to determine the number of replications. The significance level, α, is set to 0.05; power ($1-\beta$) is 0.80; effect sizes for the factors (Bed, OR, Efficiency) is 0.25 and all two factor interactions is 0.10, by assuming that the main effects have more significance than interactions. We calculated the number of replications using the method of independent replications to ensure small variances, with the following formula:

$$[(Z_{\alpha/2})^2 (C.V.)^2] / \Delta^2$$

where Δ is the desirable relative error in percentage (we have assumed a relative error of 2% in our calculations), C.V. is the coefficient of variation obtained from the pilot run (46 replications). After computing the number of replications required for each of the 21 hospital combinations, we chose the largest as the number of replications for all hospital combinations to ensure a narrower confidence interval. The largest value obtained from the above formula is slightly smaller than 100. For simplicity, we used 100 replications for all experimental studies.

A warm-up period is used for the system to stabilize to its steady state. The transient and steady-state outputs are extracted from simulations. The outputs are average waiting times of the patients in the queues of ER, Lab and OR before they receive required medical attention.

Fig. 3 shows the static and dynamic components of a hospital simulation model and the relationship between the simulation model and the metamodel.
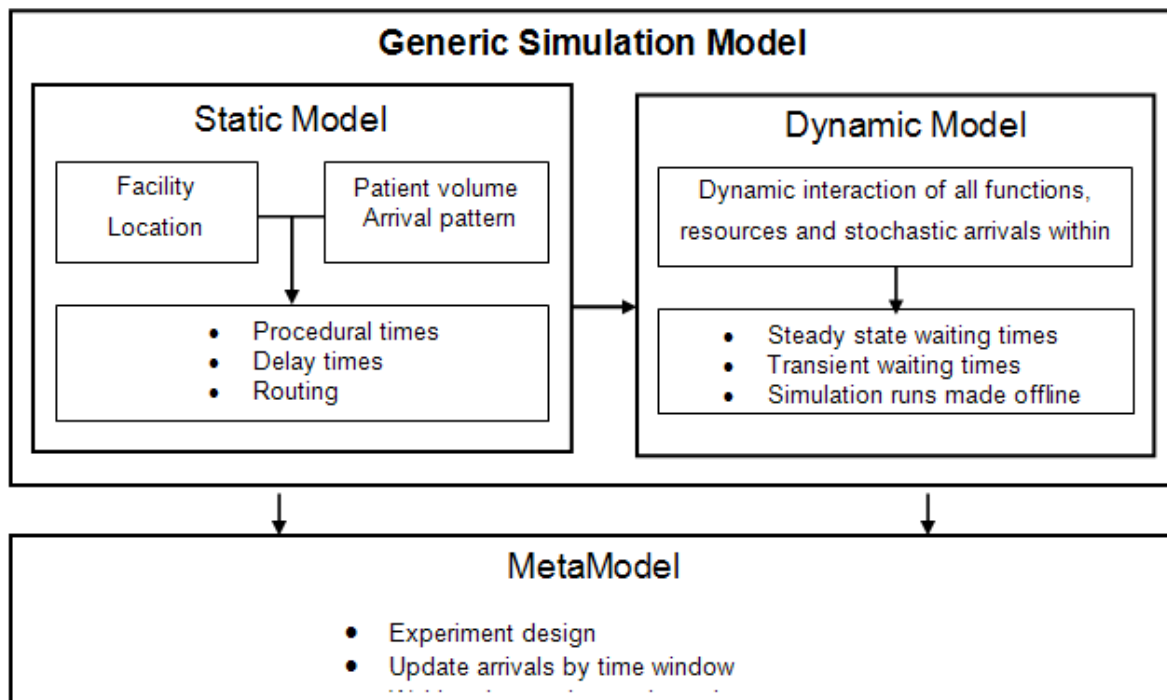
Fig. 3. Generic simulation model and metamodel

6.4.4.1 Pre-EQ simulation model

As ORs are the most critical resource, we classify patients as either OR patients who require surgeries or non-OR patients. For routing these patients we needed define a treatment path based on clinical guidelines. The route i.e., the sequence of locations that a patient goes through within the hospital is determined by patient type and procedure types. In the simulation model, we used the service times required to treat the specified injuries provided by Mercy Hospital and the Erie County Medical Center in Buffalo, New York [6.4-22, 12].

In a real dynamic situation, the rate of patient arrivals changes constantly. However, it is impractical to obtain detailed arrival patterns of all possible hospitals. We assumed that throughout our simulations, for any hospital, there is a constant arrival rate before the earthquake. Therefore, we used national statistics to calculate pre-EQ average daily patient arrivals. The arrivals are assumed to follow a Poisson process. Since annual ER visits are directly proportional to bed size (Fig. 1), it provides the basis to calculate the ER patient arrival rate

before the earthquake. In addition, there are scheduled surgical patients and inpatients who require surgery. The inpatient volumes, though small in proportion, are modeled along with the scheduled surgical patients.

We verified the simulation model by using available historic data to compare the simulated OR utilization with the real utilization for a few hospitals. The results differed by only about 2%.

6.4.4.2 Post-EQ simulation model

Although the post-EQ simulation model does not differ from the pre-EQ model in the physical elements, patient volumes are changed to post-EQ volumes. Therefore, in the post-EQ model, only serious inpatient patients and surgical patients from the ER are assumed to go to the OR.

6.4 4.3 Patient types

Injuries from an earthquake can be broadly classified into injuries directly attributed to the earthquake and those that are not. Patients use varying types of hospital resources for different durations. Therefore, it is necessary to identify patient types in an earthquake.

In an emergency situation such as an earthquake, documentation of patient/treatment data is usually a low-priority activity. Therefore, few injury data are available for analysis. The Northridge, CA earthquake on Jan 17, 1994 and the Loma Prieta, CA earthquake, Oct 17, 1989 are two well documented recent earthquakes [6.4-21, [6.4-7, [6.4-11, [6.4-25, [6.4-4, [6.4-23]. However, due to different criteria and definitions, the data are inconsistent, which makes it difficult to analyze the injuries.

Cheu [6.4-7] reported that during the first day of the Northridge earthquake, approximately 2,400 patients were treated at hospitals. Of these patients, 39% of the injuries were lacerations, 12% were minor cuts, 1-2% were head injury, 8-9% were orthopedic and 1% were burns. In other words, 62% of all emergency room cases were injuries due to the earthquake.

Although physical injuries contribute to a large portion of total ER visits, patients not related to earthquakes also occupy ER resources. Durkin [6.4-11] listed the distribution, by general types, of injuries and medical problems seen by four hospitals' emergency rooms for the first 24 hours after the Northridge earthquake. From Table 2, we can see that the percentage of soft

tissue/orthopedic injuries (58.4%) is consistent with the overall injury percentage (62%) for all hospitals during the earthquake [6.4-7]. The average can be considered approximately 60%.

Table 2

Distribution of all injuries and medical problems at four emergency rooms (Source: Durkin 1995)

Injury or Medical Problems	Hospital A		Hospital B		Hospital C		Hospital D		Total	
	Number	%	Number	%	Number	%	Number	%	Number	%
Soft Tissue/ Orthopedic	45	75	66	64	56	58	81	49	248	58.49
Cardiovascular	8	13	14	14	13	14	25	15	60	14.15
Neuro/ Psychiatric	5	8	2	2	6	6	16	10	29	6.84
Respiratory	0	0	7	7	9	9	12	7	28	6.60
Gastrointestinal	1	2	8	8	0	0	27	16	36	8.49
OBG/ GYN	1	2	3	3	12	13	2	1	18	4.25
Other	0	0	3	3	0	0	2	1	5	1.18

One week's injury data after the Northridge earthquake is available for Northridge Hospital in [6.4-26]. The percentages of respiratory and OB/GYN patients were 6.5% and 4.9%, respectively, which support the distribution in Table 2. Based on these two sources, the average of respiratory and OB/GYN is computed as 6.55% and 4.58%, respectively.

The soft tissue/orthopedic patients can be further divided into several subclasses. Table 3 shows a general breakup of these injuries [6.4-7], [6.4-11], [6.4-25], [6.4-4], 8[.4-23].

Table 3

Physical injury breakup

Physical injury	Loma Prieta EQ	Northridge earthquake (All hospitals first day)	Santa Barbara earthquake	Imperial earthquake	Coalinga earthquake	Average
Minor Cuts & Wounds	27.99%	19.35%	-	-	-	23.67%
Contusion, Abrasion & Laceration	20.66%	62.90%	59.57%	50.66%	60.46%	58.17%
Strain & Sprain	13.32%	-	-	-	-	13.32%
Fracture	20.06%	14.52%	20.21%	16.00%	15.81%	16.25%
Burn	-	1.61%	2.13%	2.67%	0.93%	1.67%
Head Injury	3.29%	3.23%	5.32%	8.00%	9.77%	4.45%

Since these data are from five different earthquakes and are not consistent, the total of the averages exceeds 100%. Since fracture, burn, and head injuries are usually more severe than those other patients, they require more hospital resources, particularly OR. Therefore, as a conservative estimate, we kept the percentages of these patients, and adjusted the other patients' percentage, so that their total is 100%. The result is shown in Table 4.

Table 4

Patient breakup during earthquakes

Sr. No.	Injury types	Percentage	Remarks
1[a]	Cuts, wounds, laceration, contusion, sprain	60*77.5%=46.5%	Type 1
2[a]	Fracture	60*16.3%=9.8%	Type 2
3[a]	Burn	60*1.7%=1.0%	Type 3
4[a]	Head Injury	60*4.5%=2.7%	Type 3
5	Cardiovascular	13.5%	Type 5
6	Neuro/Psychiatric	6.8%	Type 4
7	Respiratory	6.6%	Type 4
8	Gastrointestinal	8.5%	Type 4
9	OB/GYN	4.6%	Type 6

[a] These are Soft Tissue/ Orthopedic cases and constitute 60% of all cases

The patients who have similar medical needs and go through the same treatment procedure were grouped into six categories:

Type 1: Laceration, Abrasion, Contusion, Minor Cuts, Muscle Strain, and Sprains. With minor injuries, these patients do not need the OR, and are released after ER treatment.

Type 2: Fractures, Orthopedic. This type of patient requires Lab (X- ray in general) after ER treatment. Depending on lab results, some of them go to the OR; the others are discharged.

Type 3: Head injury and Burns. These types of patients require immediate treatment. Therefore, they are considered to be the highest severity type. They are routed through ER, Lab, OR, (then ICU when required) and to the "Inpatient" area (also called "beds").

Type 4: Neuro/Psychiatric, Respiratory, Gastrointestinal and Others. These patients go through the same route in hospitals as Type 1 patients. However, they have different processing times compared to Type 1 patients.

Type 5: Cardiovascular. These patients go through the ER and Lab. After diagnosis, some go to the OR, then ICU/CCU and finally to the inpatient area; the rest of them are discharged.

Type 6: OB/GYN. These patients go through the ER, Lab, OR and then the inpatient area.

The percentages of these six types are shown in Table 5.

Table 5

Patient percentage within hospitals during earthquakes

Patient type	Percentages					
	ER	Lab[a]	OR	ICU/ CCU[b]	Inpatient	Discharge
Type 1	46.5					46.5
Type 2	9.8	9.8	2.3		2.3	7.5
Type 3	3.7	3.7	3.7	3.7	3.7	
Type 4	21.9					21.9
Type 5	13.5	13.5	5.9	5.9	5.9	7.6
Type 6	4.6	4.6	4.6		4.6	
Total	100	27.0	16.5	9.6	16.5	83.5

[a] Types 2, 3, 5 and 6 are assumed to undergo Lab tests.
[b] Types 3 and 5 are assumed to go to Intensive Care Unit/ Cardiac Care Unit

Physical injuries (Types 1, 2 and 3) occupy 60% of total ER visits, and of these only 10% are hospitalized (Inpatient) [6.4-20], [6.4-10]. Therefore we can assume that only 6% (Types 1, 2

and 3) of patients require surgery. Since all Type 3 patients require surgery (3.7% overall), the proportion of Type 2 patients who go to the OR is 2.3% (6-3.7=2.3).

Durkin [6.4-11] also reported that within one week after the Northridge earthquake, 7,192 patients had been treated and released from hospital emergency rooms and 1,419 patients had been admitted to hospitals for further treatment. These patients result in a 16.5% admission rate for all ER visits. Therefore, the proportion of Type 5 patients who go to the OR is:

16.5 - 2.3 - 3.7 - 4.6 = 5.9%. In addition, we have to consider inpatients who require surgery due to an emergency medical condition:

Type 7: Emergency inpatients. They are inpatients who must undergo surgery due to an emergency medical condition such as cardiac arrest. Their percentage is relatively small. After the OR, they usually go to ICU/recovery rooms and then return to the inpatient area.

Since waiting times differ significantly between surgical and non-surgical patients, these seven patient types are further classified into OR and non-OR patients. OR patients: Type 3, Type 6 and part of Type 2 and Type 5 who go to the OR. Non-OR patients: Type 1, Type 4 and part of Type 2 and Type 5 who do not go to the OR. Routing of these types of patients within the hospital is shown in Fig. 4.
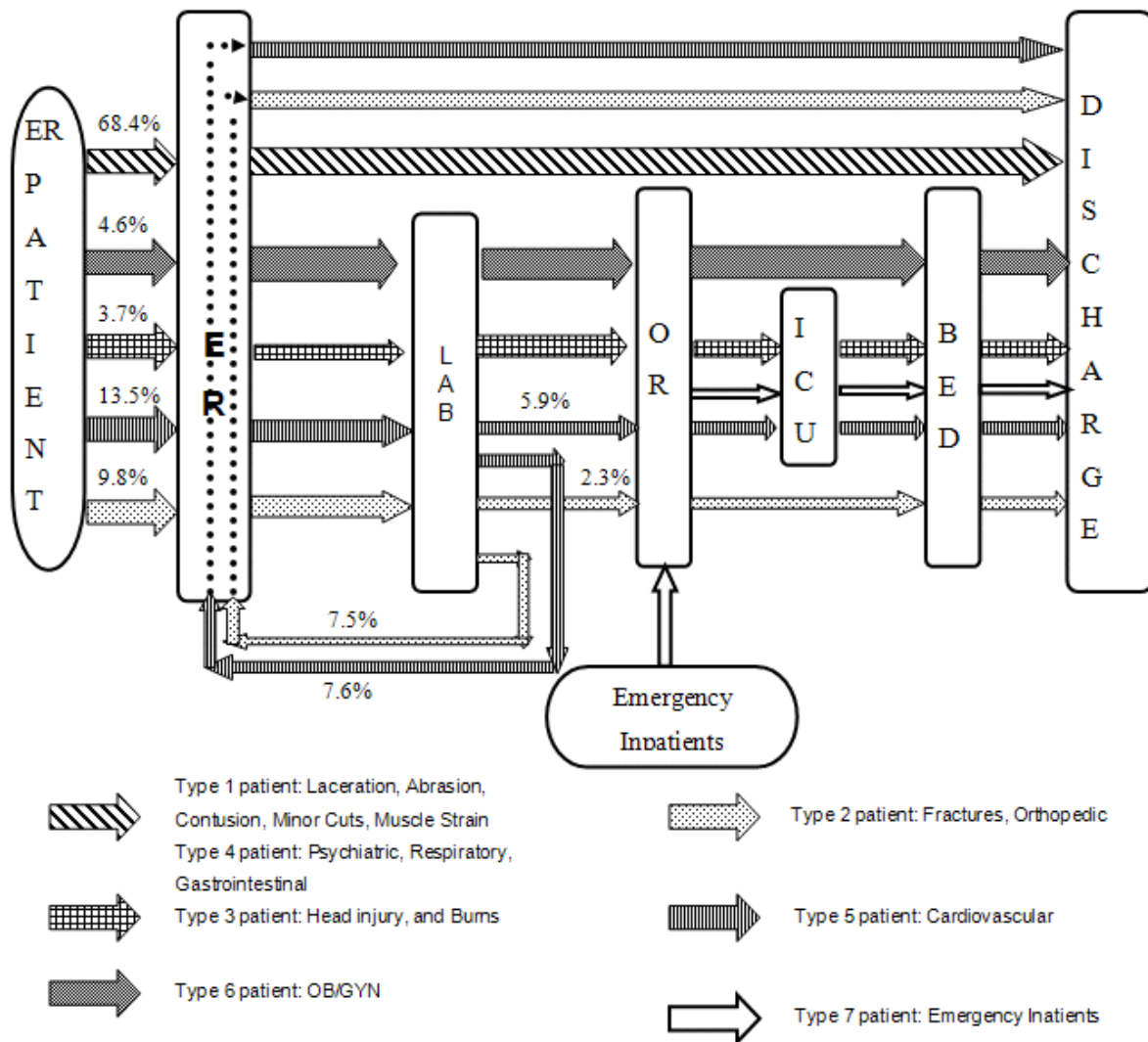
Fig. 4. Patient routings within hospitals during earthquakes

The non-OR patients are taken on a first-come-first-served basis. After the necessary lab tests, they are discharged from the ER. The OR patients follow the same route but after receiving lab tests they go to the OR. Type 2 and Type 6 patients coming out of the OR go to the inpatient area. Due to their medical needs from specific injuries, Types 3, 5 and normal patients go through ICU, then the inpatient area. We have not explicitly modeled medical-surgical and telemetry units, but the time it takes the patients from when they first receive medical attention to their discharge includes time at the important units based on data collected from hospitals.

6.4.4.4 Post-EQ service times and service logic

Based on discussions with hospital ER staff [6.4-24], we assume that the service times in the ER and lab will accelerate by a minimum of 30% after a disaster. In addition, emergency management directives require the hospitals to cancel all regularly scheduled surgeries except medical emergencies. However, surgical times in the OR are unchanged. Therefore, if the patient arrival rate is unchanged and if there is no facility damage, the hospital capacity is virtually expanded.

Although the simulation experiment used a constant patient arrival rate both before and after the earthquake, they were not necessarily the same. The inter-arrival time is considered to be exponentially distributed, i.e., the arrival assumes Poisson process. The service times of the patients follow different distributions also based on the data collected from hospitals. Further, the percentages of all patient types are considered to be constant regardless of patient volume change, and patients are assumed to undergo treatment in a FIFO (First In First Out) order.

6.4.5 Capacity prediction model

After the simulation is run for each hospital in the partial factorial design (Fig. 2), we obtain both OR and non-OR patients' waiting times. As mentioned before, this includes all the waiting time and is thus closely related to survivability. From our interviews with hospital ER staff, a survivability of one hour is allowed for OR patients. Since OR patients' waiting time is generally more critical than that of non-OR patients, we focus on OR patients in the following steps.

6.4.5.1 Pre-EQ steady-state waiting time equation

For each of the 21 different hospital settings in the partial factorial design, we obtain the steady-state pre-EQ waiting time from simulation. Then, a metamodel in regression of these 21 sets of results relates pre-EQ steady-state waiting time to number of beds, number of ORs, and efficiency:

$$T = C_0 + C_1 \cdot B + C_2 \cdot O + C_3 \cdot E + C_4 \cdot B \cdot O + C_5 \cdot B \cdot E + C_6 \cdot O \cdot E \qquad (1)$$

where T = steady-state waiting time before the earthquake

B = number of beds in the hospital

O = number of OR's

E = efficiency index

$C_0, C_1, C_2, C_3, C_4, C_5$ and C_6 are constants. The R-square value is 94.5%, indicating a good fit.

The resulting equation also validates the effect sizes selected during the power analysis for the main factors and only two factor interactions.

6.4.5.2. Post-EQ steady-state waiting time equation

Post-EQ waiting time regression equations have the same form as the pre-EQ waiting time equation. The patient volumes after the earthquake that can be treated may vary from a zero volume of patients from the earthquake to a threshold maximum volume.

If after the earthquake, no EQ patients arrive, there would only be a minimum arrival rate of regular ER patients, resulting in a large capacity available. This patient volume is a hypothetical base case. When any disaster takes place the incoming patient volume would be greater than or equal to this base case.

Using the same regression approach, we obtain the steady-state base case waiting time for any given hospital. The regression of the experimental design showed an R-square value of 90%.

By experimenting on various post-EQ arrival rates, the critical arrival rate is determined. That is, any sustained rate greater than this critical rate would push the system into inequilibrium. We call this situation the critical case, where the system is on the threshold of becoming over-capacitated. We are able to obtain critical patient arrival rates for any hospital, in a regression, with the R-square value being 95%.

In addition to the critical case patient arrival rate, we can also obtain the steady-state critical case waiting time. Then, regression will give us this value for any given hospital. The R-square value for this regression is 85%.

6.4.5.3 Arrival rates between the base case and the critical case

In order to find the relationship between patient arrival rate and the steady-state waiting time under situations between the base case and the critical case, we perform a series of simulations with different arrival rates for each hospital. As an example, results for the 500-Bed, 15-OR and 1,200-efficiency index combination are shown in Fig. 5.
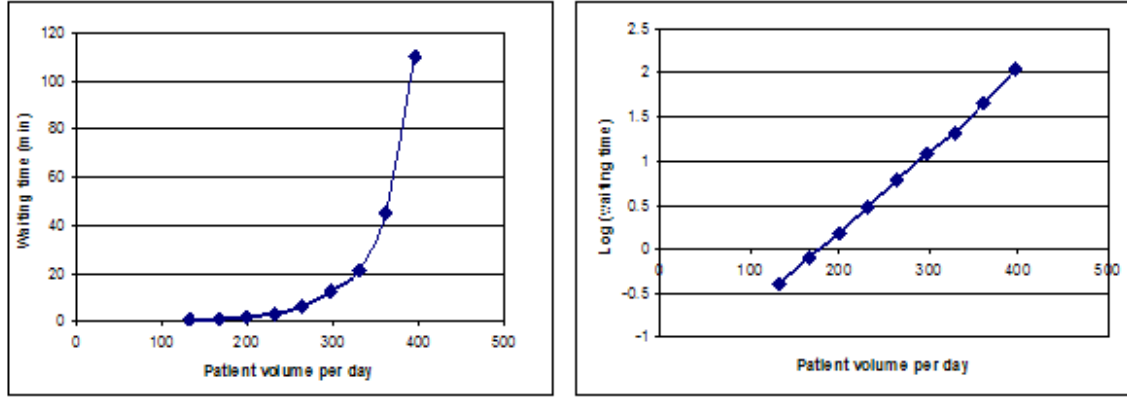
a) Steady state waiting time vs. patient volume   b) Logarithmic scaled steady state waiting time vs. patient volume

Fig. 5. Relationship between steady state patient waiting time and daily patient volume
(For a 500-Bed, 15-OR, 1200-Efficiency combination)

From Fig. 5 (a), waiting time increases exponentially with patient arrival rate. Fig. 5 (b) shows the logarithmic scaled steady-state waiting time. There is a good log-linear relationship between waiting time and patient arrival rate.

Therefore, for any given hospital, the following relationship holds:

$$Log\,(T_s) = a + b*\lambda \qquad\qquad (2)$$

where a and b are constants, T_s = Steady-state waiting time and λ = Patient arrival rate

For any given hospital combination, the above equation corresponds to a straight line in a two-dimensional space of *Log (Steady-state waiting time)* and *Patient arrival rate*. Therefore, the

constants *a* and *b* can be uniquely determined by two points known to lie on the line. These points correspond to the base case and the critical case. It is important to note that *a* and *b* depend on the three basic hospital factors only, which strongly supports our research objective on developing generic hospital models to represent all hospitals in the disaster area.

To further verify this approach, we chose another hospital setting and ran a simulation under a different set of patient arrival rates. The results were fitted into a straight line, and then compared with the calculated line. These two lines are nearly coincident, indicating a general validity of the log-linear model.

The typical transient waiting time behavior between the pre-EQ steady state and post-EQ steady state is shown in Fig. 6, for a hospital with 500-beds, 15-ORs and 1200-efficiency index, when patient arrival rate changes from 132 to 396 per day after an earthquake hits at time 2,000 (simulated minutes).
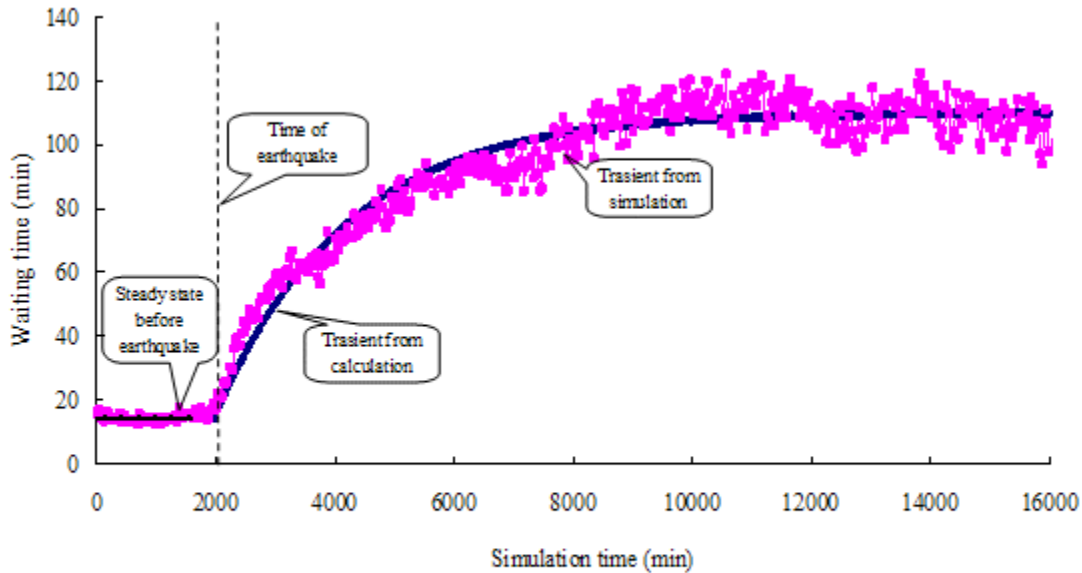
Fig. 6. Transient behavior of waiting time (Earthquake hits at time 2000)

The hospital transient behavior is described by the weighted sum of two exponential functions. The two exponential functions take into account all arrival rates from the base case to the critical case and any state in between. The time earthquake strikes is t_{eq}; the transient waiting time at

clock time t is $T_r(t)$; the steady state waiting time before earthquake is T_i; and the steady-state waiting time corresponding to the patient arrival rate after the earthquake is T_f. Assuming static but different pre-EQ and post-EQ arrival rates, the following equation is obtained,

$$T_r(t) = \mathrm{T}_i + (1-\alpha) \cdot (\mathrm{T}_f - \mathrm{T}_i) \cdot (1 - e^{(\frac{t_{eq}-t}{\tau_1})}) + \alpha \cdot (\mathrm{T}_f - \mathrm{T}_i) \cdot (1 - e^{(\frac{t_{eq}-t}{\tau_2})})$$

For dynamic post-EQ arrival rates, the relation can be generalized as follows. If at time t_1 the waiting time is T_1, the patient arrival rate λ during the transient (within a time interval from t_1 to t_2) is a constant, and the steady-state waiting time corresponding to λ is T_2, then the transient waiting time $T_r(t)$ is given by

$$T_r(t) = T_1 + (1-\alpha)(T_2 - T_1)(1 - e^{\frac{t_1-t}{\tau_1}}) + \alpha(T_2 - T_1)(1 - e^{\frac{t_1-t}{\tau_2}}) \text{ for } t_1 \leq t < t_2$$

where τ_1 and τ_2 are two time constants corresponding to the base case and critical case, respectively. They are approximately linearly related to the time it would take to reach from a pre-EQ steady state to the post-EQ steady state. The higher the value of τ, the longer it would take to reach a new steady state from the previous steady state and vice-versa. Weighting factor α is between 0 and 1. In the base case, $\alpha = 0$, and in the critical case, $\alpha = 1$.

We determined τ_1 and τ_2 from the simulations of the base case and the critical case. After a regression for these two time constants, we were able to compute τ_1 and τ_2 for any given hospital within the range of our experimental design.

Next, we needed to determine the value for α. We performed several simulations with different post-EQ patient arrival rates for selected hospital combinations. We fitted these transient results according to equation (4) and estimated the α value. Again, the logarithmic scaled α value was found proportional to patient arrival rate. An example is shown in Fig. 7. The relationship is:

$$Log\ (\alpha) = c + d \cdot (\lambda)$$

where c and d are constants for a particular hospital, λ = patient arrival rate. For the base case, α = 0, λ = base case patient arrival rate. For the critical case, α = 1 and λ = critical case patient arrival rate.
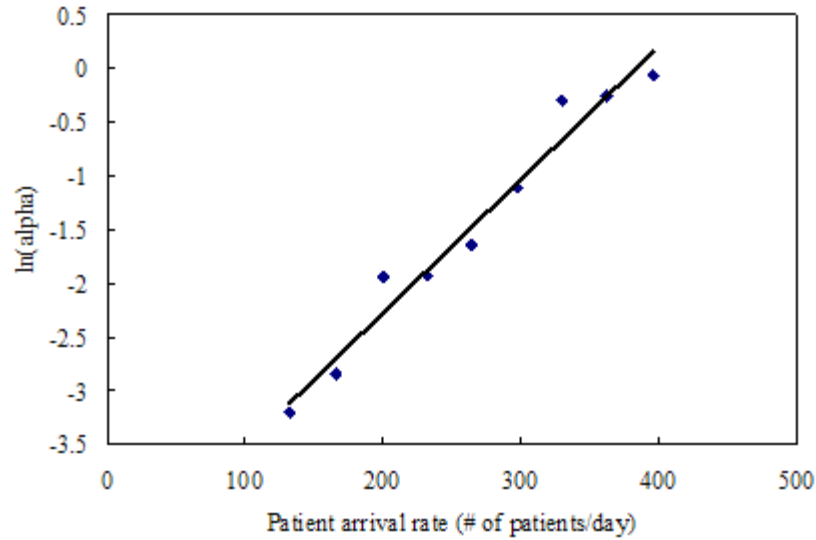
Fig. 7. Relationship between α and patient arrival rate

Therefore, we can determine c and d for any hospital. Notice that when α = 0, $Log(\alpha)$ does not exist. From actual data, we can use $Log\ (0) = -3.2$ as an approximation.

6.4.5.4 Temporal waiting times and verification

The dynamic nature of the arrivals is captured by continuously calculating the average arrival rate within a 30-minute window. This rolling time window approach is able to capture the trend while smoothing the fluctuations in arrival rates. The waiting time is estimated continuously with the updated arrival rate.
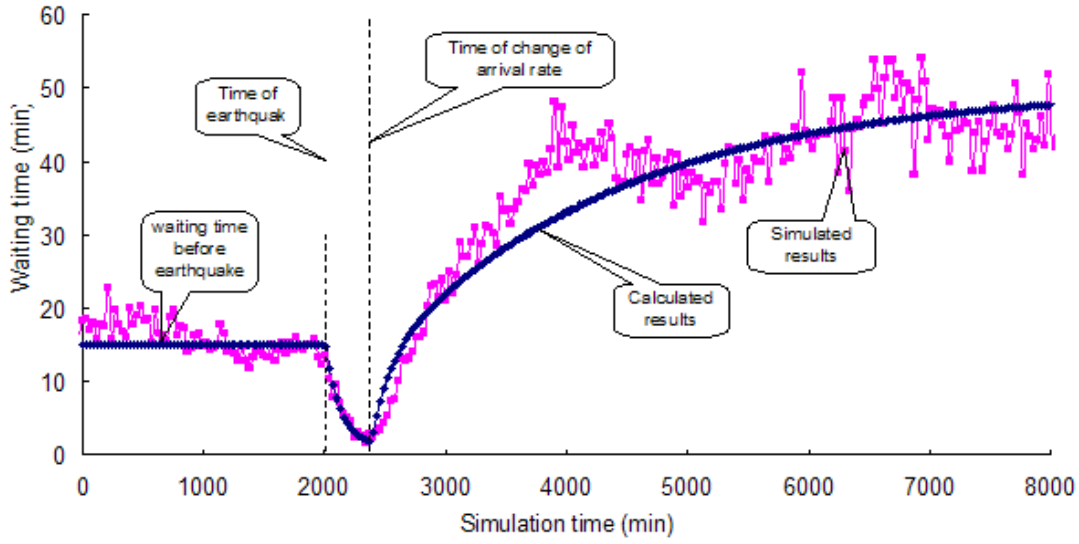
Fig. 8. Temporal waiting time corresponding to dynamic patient arrival rate

To verify the capacity estimation methodology, we simulated a dynamic patient arrival rate. The response is shown in Fig. 8. From time 0 to 2,000, patient waiting time at the hospital stabilizes at 16 minutes. The earthquake strikes at minute 2,000, which results in an increase in patient volume from 132 to 166 per day. Yet, because ER and lab speed up their processing, there is a decrease of waiting time immediately after the earthquake. Then at time 2,400, the patient arrival rate is again changed from 166 to 362 per day. The waiting time increases from that point onwards.

6.4.5.5 Capacity estimation

Since our ultimate objective is to estimate hospital capacity in terms of the number of patients that the hospital can accept with a acceptable waiting time not exceeding the survivability, it is necessary to convert the waiting times into capacity estimates.

Assuming the maximum permitted waiting time is T_m, from equation (2), in steady state, this waiting time corresponds to a maximum patient arrival rate (λ_m) given by

$$\lambda_m = (\ln(T_m) - a)/b$$

Assuming the current waiting time (transient waiting time from double exponential curve) to be a steady-state waiting time for a certain patient arrival rate, this arrival rate (λ_s) can be calculated as

$$\lambda_s = \ln(T_r(t) - a)/b$$

where $T_r(t)$ = current waiting time. Then the available capacity is equal to the difference between the maximum capacity and the used capacity

$$C = (\lambda_m - \lambda_s).\Delta t$$

where Δt = length of time, C = available capacity. If the length of time is one hour, then the available hourly capacity (C_h) is

$$C_h = \lambda_m - \lambda_s$$

6.5 Dispatch & Routing Modeling

In this section we review the Dispatcher-Router simulation model and its strategies. Included are both testable requirements and design details. We briefly present in chronological order about some of the solution methodologies developed. Some of them did not make through to the implementation stage as more quick and efficient strategies were developed with time that clearly had some advantage over others in case of a disaster environment. The aim was to develop a robust methodology for dispatching and routing of emergency vehicles (EVs) in a post-disaster environment with the support of data fusion for decision-making. In this work we considered an earthquake scenario with a large number of casualties needing medical attention. In the immediate aftermath of an earthquake, Emergency Response Centers *(ERCs)* might have to deal with collapsed buildings, fires, and hazardous material spills. Management of emergency service resources in such an environment requires efficient dispatch and routing strategies that provide rapid response for casualty pick up and delivery. The goal is to service the maximum number of the highest priority casualties with minimum service times.

6.5.1 Context: the DIRE Simulation Environment

The Center for Multisource Information Fusion (CMIF) has implemented a test bed to address Information Fusion in support of crisis-center decision-makers dealing with post-event situations for both earthquakes (a natural disaster) and chemical attacks (a man-made disaster). The thrust of this simulation is to provide realistic data that simulates the chaotic flow of both information and misinformation in the immediate aftermath of a disaster and to process this data, emphasizing level-2 and level-3 data fusion techniques. The result is an improved situation awareness that can be directed back into the simulation at (simulated) decision points in order to improve the management of the disaster according to certain measures of performance and effectiveness.

A scenario as complex as this one, comprised of many, disparate information sources, of varying reliability and timeliness, in a large urban landscape, dictated that its simulation be correspondingly complex. To that end it was decided to implement it as a number of independent simulation models, focusing on the limited perspective of the reporting and treatment of human casualties (see figure 1.) The requirement for implementing several independent but cooperating models necessitated the use of a framework under which the various models can cooperate, each adhering to its own time-line and yet not encounter causality problems such as would be generated if each model ran to completion without some synchronization with the models on which it depends. Such a framework is the High-Level Architecture / Run-Time Infrastructure or HLA/RTI, developed under the leadership of the Defense Modeling and Simulation Office of the United States government. The High Level Architecture (HLA) is general purpose architecture for simulation reuse and interoperability. The HLA was developed to support reuse and interoperability across the large numbers of different types of simulations developed and maintained by the United States Department of Defense. While the HLA is an architecture (not software) the use of runtime infrastructure (RTI) software is required to support operations of a multiple-model execution. The RTI software provides a set of services used by independent models to coordinate their operations and data exchange during a runtime execution. The entire agglomeration of models is called a "federation" and the individual models are called "federates" in this context. The sole communication among federates is a series of time-stamped messages or

"interactions" which carry all inputs and outputs to and from each federate. There are almost 60 types of interactions, each of which has a fixed source-federate, destination-federate and format.

A federate has three points of contact with the remainder of the federation; (a) the message-receipt function, (b) the message-sending function, and (c) an "update" point which is called periodically and during which call, the state of the federate is updated to the next time increment value. Typically a federate will set a time interval value which dictates how often the federate will be activated at the update function.
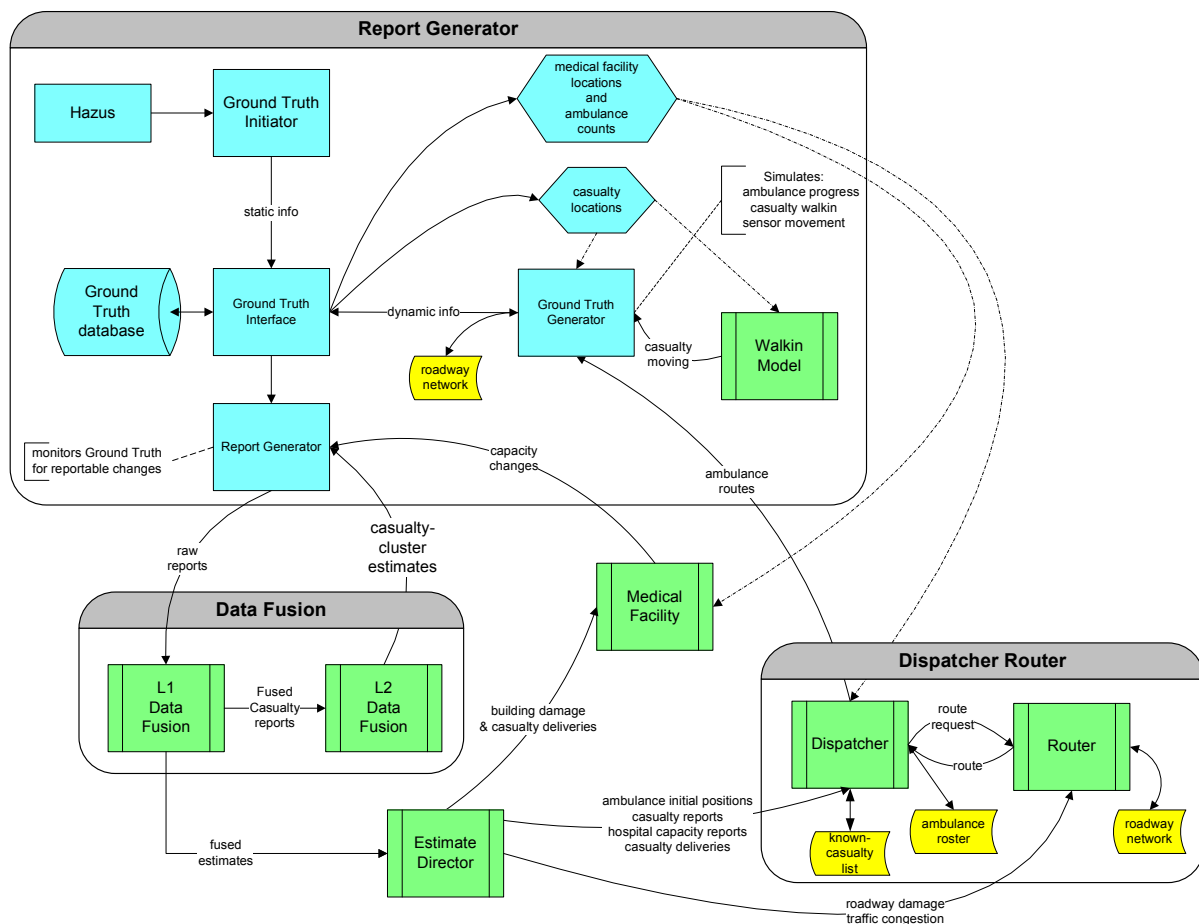
Figure 6.5.1. Interfederate Interactions

Between updates messages or interactions will be received and may be either processed immediately or saved till the next update, the choice depending on the complexity of the processing and the interdependence of the messages. At the update activation, the received

messages may be processed, the state of the federate (simulation model) advanced to the next time value and any outgoing messages generated and emitted.

6.5.2 D&R in DIRE

The Dispatcher-Router is a simulation of the two functions of dispatching ambulances ─ picking up casualties and the subsidiary one of calculating the best route (to either a casualty or a medical treatment center). The dispatcher is also a decision point in the simulation where an improved estimate of casualty location and severity, derived from the information fusion module(s) is injected back into the simulation. Thus the simulation can be run either with or without the aid of fusion, providing one rough measure of the effect of the availability of fused estimates.

The Dispatcher and the Router are two parts of the same federate; however if the necessity arose, the Router could be easily packaged as a separate federate for use by facilities other than a single ambulance dispatcher. The only function of the Router is to provide the quickest route from a source location to a destination. This calculation must take into account the effects of the disaster (such as damaged transportation infrastructure or geographic areas which must be avoided due to chemical or biological hazard).

The Dispatcher receives both reports of casualties (fused by a level-1 fusion facility), and later, reports of casualty-clusters (developed by a level-2 fusion facility) where the probability of finding severely injured casualties is high. The (simulated) dispatcher receives reports of ambulances waiting for dispatch, moving toward either a casualty or a hospital and even ambulances stopped at a previously unknown impassible obstacle (such as a collapsed bridge or tunnel) and awaiting either rerouting to their destination or a possible decision on dispatch to a new location.
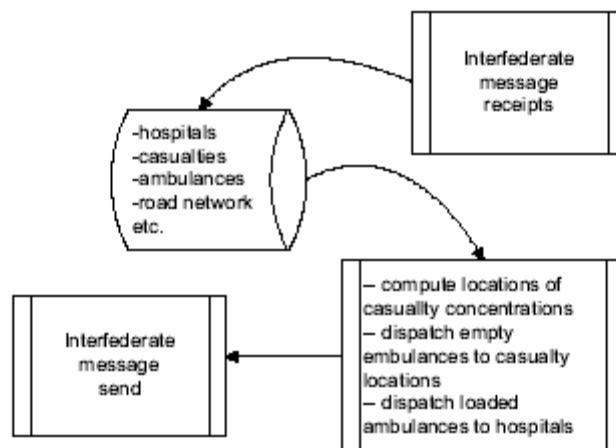
Figure 6.5.2. High-Level Data Flow

Figure 6.5.2 depicts the high-level information flow. Message receipt brings updates to the database from other simulation federates (e.g., casualty reports, casualty cluster reports, ambulance becoming free etc.). During the periodic updates, the dispatching strategy is implemented for all ambulances that currently have reached their destination (casualty pickup or hospital drop-off) and hence labeled *'idle'*. This is the trigger for ambulance routes to be generated and emitted as interactions to be sent to the model that generates the simulated ambulance movement. The message receipts are asynchronous and can come at any time but the messages that are to be sent can be generated only during the update processing.

6.5.3. D&R Design Assumptions and Interactions

6.5.3.1 Assumptions:

(1) The area chosen for study is the Los Angeles basin with the earthquake simulating rather closely the Northridge event of 1994.

(2) The Ground Truth, from which much of the simulated phenomenology is derived, has been generated by the HAZUS[1] software and includes both human casualties and building damage.

(3) The simulation is not intended to represent activity beyond 24 hours post-event (typically much less time).

(4) The simulation is a time-based model that is updated on a regular basis.

(5) Receipt of an *Ambulance Idle* message will indicate that that ambulance has no destination and is a candidate for dispatch.

(6) Receipt of an *Ambulance Idle* message with a casualty count of 0 will indicate a dispatch of that ambulance to a location that has a high probability of containing at least 3 casualties.

(7) Receipt of an *Ambulance Idle* message with a casualty count of 1 or 2 will indicate a dispatch of that ambulance to a location that contains casualties. The choice of destination will depend on the number and severity of casualties already on-board.

(8) Receipt of an Ambulance Idle message with a casualty count of 3 will indicate a dispatch of that ambulance to a hospital that has a high probability of having available capacity to treat the casualties.

(9) If on the occasion of a dispatch to a hospital, even if no hospital has any available capacity, the hospital with the best result from the attractiveness computation will still receive the dispatch.

(10) Dispatcher Router will receive reports of casualty clusters. Such reports will present a cluster as a list of cells, each containing a count of reported casualties of both severity 2 and severity 3.

---

[1] For more information on HAZUS please see http://www.fema.gov/hazus.

(11) There is no persistence in the reporting of clusters. Thus a cluster is considered to exist only from the time that it is reported until the next time the clusters are reported. Subsequent cluster reports will be considered to refer to independently-defined clusters. It is assumed that whenever higher level fusion reports clusters, it considers all the present and past information and gives estimates about the most recent state of the scenario.

(12) Casualties will be picked up by an ambulance at a dispatch location from the casualties closest to the destination node without regard for cell membership in case of a dispatch to a cluster (which the simulated ambulance driver knows nothing about).

(13) In case of dispatch to a casualty, an ambulance will pick up a casualty regardless of its ID from the node closest to the destination node specified to it.

(14) Ambulances can travel a link in any direction.

(15) The maximum capacity of each ambulance is 3.

(16) The ambulance will always try to pickup the maximum number of casualties.

(17) Type 4 casualties are mortally injured and hence not considered for pickup and delivery for obvious reasons of facilitating the service of those casualties that can still be saved.

(18) Type 1 casualties are the ones that do not need immediate medical attention and hence can be ignored.

(19) Data about road damages, traffic conditions and congestion on the roads is already available (from data fusion center). This data is summed up as a *delay factor* for each link.

(20) The capacities of the disaster area hospitals are reported as the numbers of injuries that can be treated in a given time window.

(21) There exists a model of disaster area hospitals that can provide such capacity estimates. Interested readers please refer to the paper by Yi et al [6.5-3].

(22) The condition of a patient may deteriorate while he/she is waiting in queue and hence such casualties have to be upgraded to higher casualty types. Hence a casualty can be reported more than once.

(23) The Dispatcher-Router federate is an independently-executing part of the larger simulation and has the usual three contact points (discussed above in section 2) with the other federates.

(24) The goal of this federate is to develop an awareness of the locations of (simulated) human casualties and to dispatch ambulances to those casualties (and thereafter to a hospital or a treatment center) in such a manner as to support the effort of the Data Fusion federates to improve the choosing and transportation of casualties over that of an unimproved, manual system.

6.5.3.2 Interactions

Here RG is Report Generator (that generates the reports), ED is Estimate Director (that fuses these reports and gives an estimate of the entity) and DP is Dispatcher/Router. *Input:*

(1) RGtoDP01 -Hospital locations

This message is received as initialization information and is used to build a list of all hospitals that are available for dispatch.

(2) EDtoDP01 -Casualty Observation

This message carries information about a single casualty. It is used to build a catalog of casualties to be used as dispatch targets in the event that no casualty clusters are known. The road node nearest to each hospital is not contained in the message and must be computed by Dispatcher Router.

(3) EDtoDP02 -Casualty Pickup

This message indicates that a casualty has been picked up by an ambulance. The message will be ignored and probably dropped from this list.

(4) EDtoDP03 -Casualty Delivery

This message indicates that a casualty has been dropped off at a hospital. The message will be ignored and probably dropped from this list.

(5) EDtoDP04 -Roadway Damage

This message contains reported information concerning physical damage to road links. It represents debris on the road or actual damage to a bridge or a tunnel. This information will be used to compute the average speed with which vehicles can traverse links.

(6) EDtoDP05 -Hospital Capacity

This message reports a capacity for treatment at a specific hospital, for each of three casualty severities, 1, 2 and 3.

(7) EDtoDP06 -Travel Delay

This message carries a reported delay for a specific road link. The delay will change over time and is due to traffic congestion, vehicle accident, large numbers of pedestrians etc.

(8) EDtoDP07 -Treatment Delay

This message reports an anticipated treatment delay at a specific hospital, for each of three casualty severities, 1, 2 and 3.

(9) EDtoDP08 -Ambulance Idle

This is a report from an ambulance that indicates that the ambulance has no destination and is available for dispatch. Also reported is the number of severity 2 and severity 3 casualties onboard the vehicle (if any). Different dispatching schemas are used depending on the numbers and severities of onboard casualties.

(10) EDtoDP09 -Ambulance Stuck

This report is similar to EDtoDP08 but it further represents the situation where an ambulance has followed its dispatched route, only to find that it is blocked byroad link damage that was

unknown to the dispatcher at the time of dispatch. The action to be taken will be a re-dispatch that will route the vehicle around the damaged link.

(11) EDtoDP10 -Cluster Identification

This is the report from high-level fusion processes of the location where a cluster of casualties has been detected. This cluster and its component cells will be used as dispatch targets because the probability of finding enough casualties (3) to fill an ambulance is high.

Output:

DPtoRG01 -Ambulance Route

This message contains the route from a source (ambulance present location)to a destination (casualty location or hospital).It consists of a series of road nodes interleaved with road links such that there is no 'untraversable' gap in the road map.

6.5.3.3 Casualty Cluster Generation

A cluster report will contain list of cells with an accompanying indication of Boundary ||

Non-boundary. These may be interpreted geometrically as referring to members of one or more two-dimensional groups of casualties where:

- Boundary delimits the cluster extent such that the remainder of the cluster lies on only one side of the boundary.

- Non-boundary refers to cells that "belong" to the group but are not boundary cells.

There will be cells marked "Boundary" in every cluster report. A boundary cell:

- Has at least one contiguous boundary cell.

- Has at least one contiguous cell that is not an element of the cluster.

Each reported cell will contain at least one casualty. Reports will be generated on each cluster/sub-cluster periodically. These reports will contain no historical information about the cluster but will reflect the casualty population as of the time of reporting.

Casualty clusters are reported periodically but there is no necessary identification between clusters of one report group and those of the next group. It is only guaranteed that the period between reports of one group will be shorter than the period between groups. After a cluster is reported ambulances will be dispatched as they become idle and as the cluster's attractiveness index dictates. This process will continue throughout the period until the next report group is delivered. At that time, the previous cluster definitions will be discarded and the new definitions will be used.

During this "inter-group" period, ambulances dispatched to a cluster cell will result in the maintenance of a local inventory of casualties "to be picked up". These will deter from the attractiveness of those cells that have been dispatched to so that ambulances are always dispatched, based on the latest information.

6.5.4 Router

6.5.4.1 Initial Router Design

The objective is to have the pickup and delivery of the casualties in the shortest possible time and to the appropriate hospitals to optimize the overall survivability rate. A simple (i.e. static) routing algorithm cannot be applied, since an earthquake can disrupt the road network due to damaged road segments, damaged buildings that block roads, damaged bridges and tunnels, etc. In such a case we need a set of routes that share the minimum number of links between them and are within a set percentage of the optimal shortest distance between the origin and destination (O-D). This will ensure that even if some of the links in the route for an O-D pair are damaged, we still have some useable (i.e. undamaged) backup routes available. Furthermore, we do not want to simultaneously traverse the same links as this situation increases the risk of damaged ambulances if this shared link collapses due to an aftershock. Hence we strive simultaneously for spatial dissimilarity between the alternative routes for each individual ambulance, and for temporal dissimilarity between routes for each pair of ambulances. The basic premise of our

work is that by considering temporal and spatial dissimilarity in the routing procedure we can ensure robustness.

The selection process of a path involves three stages: Generation of a large number of candidate paths for each ambulance, selection of a small set of paths for each ambulance, and selection of a single path for each ambulance, while considering all ambulance routes simultaneously.

To generate a large number of spatially dissimilar candidate paths we use the following methods:

k-Shortest Path Method

Yen [6.5-4] presented an algorithm to find k loopless paths that have the shortest lengths from one node to another node in a network. But, the paths tend to share a large number of links, implying that a very large value of k has to be used to get spatially dissimilar alternatives.

Iterative Penalty Method (IPM)

The IPM, due to Johnson et al. [6.5-5], is based on a repetitive application of an appropriate shortest path algorithm and after each application imposing a penalty on all the links in the resulting shortest path that use the same links as the previous one. Hence dissimilar paths are generated, since the sharing of links is discouraged.

Gateway Shortest Paths (GSPs)

Proposed by Lombard and Church [6.5-6] this method tries to find the shortest path by forcing the paths to go through a series of specific nodes called "gateways" and thus generate a set of spatially distinct paths by constraining them to go through different nodes. But the paths might contain loops and also might be similar.

Minimax Method

Proposed by Kuby et al. [6.5-7], this method aims to generate a set of dissimilar paths by selecting a subset of large set of paths. First k- shortest paths are generated and then a Dissimilar Subset (DS) is constructed iteratively. A dissimilarity index is defined as

$$M(P_j, P_i) = \frac{\{(1 + \beta)ds(P_j, P_i) + dn(P_j, P_i)\}}{d(P_1)}$$

where, $d(P_1)$ is the length of the first shortest path, $ds(P_j, P_i)$ is the length shared by P_i and P_j, $dn(P_j, P_i)$ is the length not shared by P_i and P_j, and β is a parameter which is a constant (generally taken = 1). We try to minimize this index. For the subsequent paths we try to minimize the maximum of the indices between the candidates and the previous paths. The formulation can be written as:

$$\underset{j \notin DS}{Min} \left[\underset{i \in DS}{Max} \{M(P_i, P_j)\} \right]$$

Generally the Iterative Penalty Method and k-shortest Path method give the best results for generating a candidate set of paths. To generate a small set of paths, we seek a subset of the routes generated at the previous stage. We want this smaller set of routes to be both spatially and temporally dissimilar. This can be posed as a 'p-dispersion problem'. The classical form of p-dispersion problem is to select p out of m given candidate points, such that the minimum distance between pairs of selected points is maximized. The objective is to have a dispersed set of points in space. Erkut [6.5-8] described this problem in detail. He gave two IP formulations, two Branch and Bound methods, and a two stage heuristic procedure to solve the problem. Erkut, Iksal and Yenicerioglu [6.5-9] compared 10 different heuristics available for solving the p-dispersion problem. The p-dispersion problem has been used in various contexts like military installations to avoid enemy attack, location of fast food franchise in an urban area, etc.

For our problem, we have used the two-stage heuristic to solve the p-dispersion problem given in [6.5-8]. In the classical p-dispersion problem, p out of m given points (1< p< m) are selected in some space, where the objective is to maximize the minimum distance between any two of the selected points. If M is the set of candidate points ($|M|= m$), and $P \subseteq M$ ($|P| = p$) and Wij is the distance between the candidates i and j, the p-dispersion problem can be presented as:

$$\underset{P \subseteq M}{Max} \left[\underset{\substack{i,j = 1,\dots,p \\ i \neq j}}{Min} \{W_{ij}\} \right]$$

In our case, *Wij* is the dissimilarity between two paths. Hence, given m paths, p paths are chosen in such a way so as to maximize the minimum dissimilarity between any two paths. The basic idea of using the two-phase heuristic is to construct an initial solution in a semi-greedy fashion and then to perform a local search to improve the initial solution [6.5-8].

The p-dispersion heuristic described above is used to get a set of paths for a single O-D pair. Since in our problem we are having n different O-D pairs, we need to apply the p-dispersion heuristic n times. After we get a small set of paths for each O-D pair then we try to address the issue of temporal dissimilarity by defining a similarity index between two O-D pairs as

$$S(P_i, P_j) = \frac{\{L(P_i \cap P_j)/L(P_i) + (P_j \cap P_i)/L(P_j)\}}{2}$$

Hence Dissimilarity is, *Wij = 1-S (Pi, Pj)*. This is the dissimilarity index considering just the spatial dissimilarity. To account for temporal dissimilarity we modify this index as, $W_{ij} = 1 - S^{\theta t}$. θ is a model parameter (needs to be calibrated), and t is the difference between the two start times of the ambulances for which we are calculating dissimilar routes. Since we know the start time of the ambulance from its current location and the expected travel time to the destination, we know the expected arrival time also. Therefore, care is taken so that too many ambulances do not reach the same destination at the same time. This is done by varying θ by say 30 minutes, 60 minutes or 90 minutes and then seeing which one is giving the best dissimilarity index. Also, sometimes, we might not want to consider this temporal dissimilarity in the route generation process. This is the case where we have to send a number of ambulances to the same spot given that a cluster of patients have been located and they need to be taken to a hospital.

6.5.4.2 Final Router Design

The coding of the routing methodology in C++ is described in detail in [6.5-3] and prepared it as a stand alone federate which could be called upon by any other federates to come up with routes between any origin and destination pair. Later it was integrated with the dispatching federate for all practical purposes in the whole simulation.

6.5.5 Dispatcher

The dispatcher logic is described in detail elsewhere [6.5-3] and is summarized here in pseudocode.

```
For Each Update
     For Each Ambulance
          if idle
               switch (onboard casualty count)
               cas == 3                    // ambulance is full
                    Dispatch-to-most-attractive-hospital
                    generate route message (DPtoRG01)
                    update state-variables
                    break                   // go to next ambulance
               cas == 2                    // room for 1 more
               cas == 1                    // room for 2 more
                    Step size = 800 (metres)
                    if( On-Board ==2)
                         Multiplier = 4
                    if( On-Board ==1)
                         Multiplier =  6 or 8 (Depending on severity)
                         Dispatch-to-neighborhood
                         if dispatch successful
                               generate route message (DPtoRG01)
                               update state-variables
                               break      // go to next ambulance
                         Dispatch-to-most-attractive-hospital
                         generate route message (DPtoRG01)
                         update state-variables
                         break                // go to next ambulance
               cas == 0                     // no casualties onboard
                    if clusters exist
                         For Each cluster
                               calculate attractiveness
                         pick most attractive cluster
                         if amb loc is inside a cell of the cluster
                               Dispatch-to-neighborhood
                               if dispatch successful
                                     generate route message (DPtoRG01)
                                     update state-variables
                                     break // go to next ambulance
                         // no neighboring casualty is attractive
                         //    or  amb is outside all clusters
                         Dispatch-to-cluster-boundary-cell
                         if dispatch successful
                               generate route message (DPtoRG01)
                               update state-variables
                               break      // go to next ambulance
                         else               // boundary cells unusable
                               mark cluster "unusable"
                               break      // repeat this ambulance
                    else                    // no clusters exist
                         Multiplier = 40
                         Dispatch-to-neighborhood
```

```
                      end of switch
              else                // this ambulance is moving.
// Dispatch-to-neighborhood
        for i = 1 to Multiplier
              neighborhood = i * Step size
              For Each casualty in neighborhood
                      calculate attractiveness
              pick max attractiveness
              if max > some minimum attractiveness
                      calculate fastest route
                      report dispatch successful
              else
                      report dispatch unsuccessful
// Dispatch-to-cluster-boundary-cell
        find closest cluster
        For Each boundary cell in this cluster
              calculate attractiveness
        pick max attractiveness
        if max > some minimum attractiveness
              calculate fastest route
              report dispatch successful
        else
              report dispatch unsuccessful
// Dispatch-to-most-attractive-hospital
        For Each hospital
              calculate distance from ambulance loc
              calculate attractiveness
        pick most attractive hospital
        calculate fastest route
        report dispatch successful
```

The code was tested (stand alone) to check for bugs and other necessary fixes. It was then integrated with Router code where in the Dispatcher uses a function call to ask Router to generate routes.

6.5.6 Effect of Shelters on Transportation System

Here we tried to model the effect of displaced population in case of a disaster situation on the traffic system situation. Traffic system situation is defined by the reduced road capacity and the consequent increase in travel times. This capacity deduction can be due to the debris on road, due to damage of the road segment itself or due to displaced population. Our aim is to deduce the effects of displaced population over road capacity deduction. The inaccessible hospitals and clusters are a direct result of reduced road capacities. We assume that we have been given an area of study wherein the effects of population displaced from their homes on the traffic system situation has to be evaluated. As a case study we will work on the Northridge region in LA. We assume in our analysis that a disaster like an earthquake has struck this region. Data regarding each census tract like –population, number of casualties and their severities, and the damage is

AFOSR F49620-01-1-0371

provided by HAZUS. For each census tract the number of displaced population is directly proportional to the number of casualties and the proportionality constant is given by HAZUS. For each census tract we can assume that only a small fraction of the population is injured and a large fraction is the potential displaced population that will be displaced from their homes to other places. It is due to this displaced population that the roads get congested contributing significantly to the problems faced by the rescue workers and ambulances to serve the casualties in the affected area. The displaced population is the one that has been displaced form their homes and are trying to get somewhere, hence in the process congesting the roads since the magnitude of displaced population can be enormous (it is much more than the number of casualties). We have data coming in form Level 2 Fusion about the cluster of casualties. We assume that L_2 Fusion provides us with *casualty clusters* that are a set of contiguous cells and each cell consists of a certain minimum number of casualties. In order to use this cluster information, we can think of the census tracts to be made up of such cells. Some of the cells of the census tract will be part of the clusters and some of them will lie outside the cluster.

We assume for our analysis that a member of the population of a cell is likely to displace from its parent cell depending on whether the cell is a part of the cluster or it is outside a cluster. When the cell is outside a cluster, this likelihood is inversely proportional to some power of the distance (say square of distance) between its current location and the closest cluster boundary and is given by:

$$\Pr(displacement) = \frac{k_1}{d^a}$$

where k_1 is the constant of proportionality. When the cell is part of a cluster, this likelihood is directly proportional to the fraction of casualties in the cell and is given by:

$$\Pr(displacement) = k_2 . Fraction\ of\ people\ injured\ ;$$

where k_2 is the constant of proportionality.
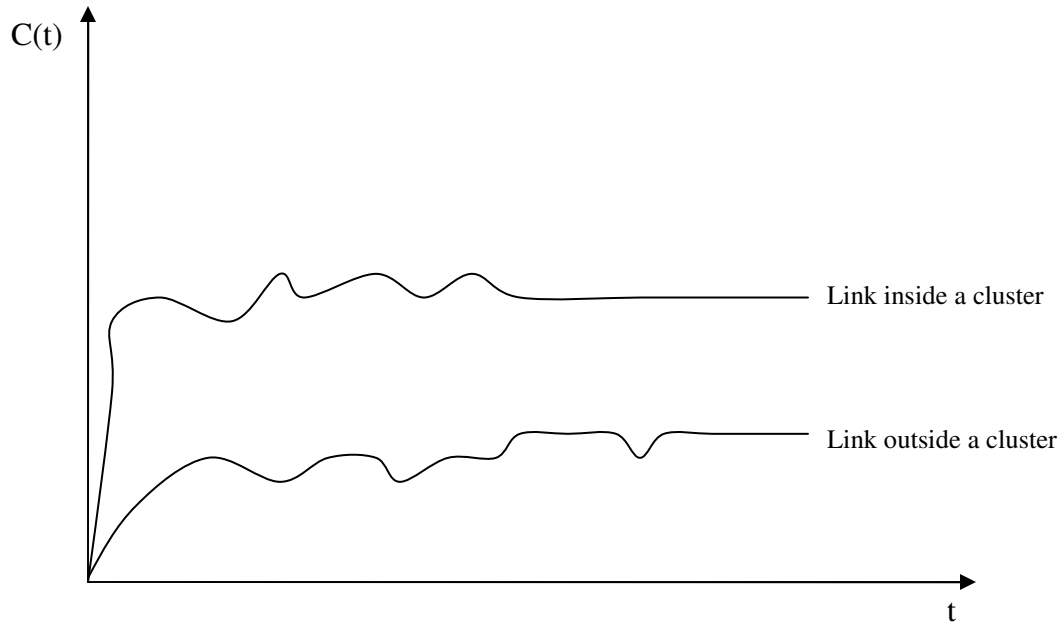
To explain the behavior of the displaced population, how they will move and which places they will try to go in order to reach safer location, a Gravity Model was used. The gravity model offers a good application of the spatial interaction method. It is named that way because it uses a similar formulation than Newton's gravity model, which implies that the attraction between two objects is proportional to their mass and inversely proportional to their respective distance. Consequently, the general formulation of spatial interactions can be adapted to reflect this basic assumption to form the elementary formulation of the gravity model:

$$T_{ij} = k \frac{P_i P_j}{d_{ij}}$$

- P_i and P_j : Importance of the location of origin and the location of destination.

- In our case we can assume that:

    o P_i = Number of casualties in the cluster, and

    o P_j = Number of beds available in the hospital.

- d_{ij} : Distance between the location of casualty and the location of hospital.

- k is proportionality constant. Related to the rate of the event. For instance, if the same system of spatial interactions is considered, the value of k will be higher if interactions were considered for a year comparatively to the value of k for one week.

Thus, spatial interactions between locations i and j are proportional to their respective importance divided by their distance.

There is a simple and much more flexible formulation of the gravity model:

$$T_{ij} = k \frac{P_i^{\lambda} P_i^{\alpha}}{d_{ij}^{\beta}}$$

1. β (beta) : Transport friction. Related to the efficiency of the transport system between two locations. Rarely linear in space as the further the movement the greater the friction of space. For instance, a highway between two locations will have a weaker beta index than a road. This is useful and hence similar parameter can be used in our case.

2. λ (lambda) : Potential to generate movements (emissiveness). For movements of people, lambda is often related to an overall level of welfare. For instance, it is logical to infer that for retailing movements an equal population, a location having higher income levels will generate more movements. In our case it can be assumed that the cluster having higher number of casualties will generate more movements.

3. α (alpha) : Potential to attract movements (attractiveness). Related to the nature of economic activities at the destination. For instance, with an equal population, a center having important commercial activities will attract more movements. In our case it can be assumed that the hospital having higher number of beds will attract more casualties.

A part of the difficulties related to the usage of spatial interaction models, notably the gravity model, is related to their calibration. Calibration consists in finding the value of parameters (constant and exponents) to insure that the estimated results are similar to the observed flows. If it is not the case, the model is almost useless. It is impossible to know if the process of calibration is accurate without comparing estimated results with empirical evidence.

In the two formulations of the gravity model that has been presented, the simple formulation offers a good flexibility for calibration since four parameters can be modified. Altering the value of beta, alpha and lambda will influence the estimated spatial interactions. Furthermore, the value of the parameters can change in time due to factors such as technological innovations and economic development. For instance, improvements in transport efficiency generally have the consequence of reducing the value of the beta exponent (friction of space). Economic development is likely to influence the values of alpha and lambda.

Often, a value of 1 is given to the parameters, and then they are progressively altered until the estimated results are similar to observed results. Calibration can also be considered for different O/D matrices according to age, income, gender, type of merchandise and modal choice. A great part of the scientific research in transport and regional planning aims to find accurate parameters for spatial interaction equations. This is generally a costly and time consuming process, but a very useful one. Once a spatial interaction model has been validated for a city or a region, it can then be used for simulation and prediction purposes.
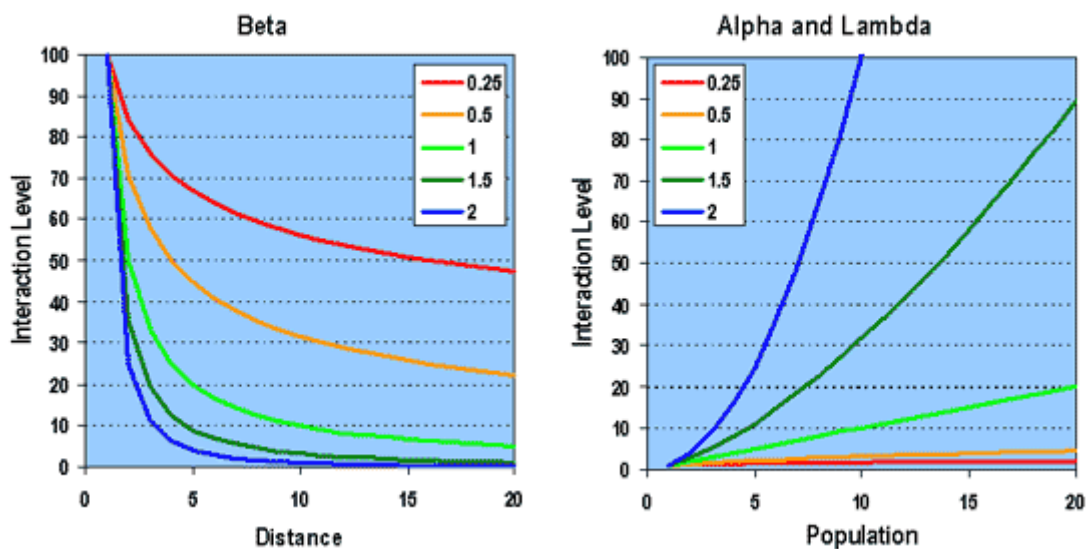
Figure 6.5.3. Effects of beta, alpha and lambda on Spatial Interactions

Variations of the beta, alpha and lambda exponents have different impacts on the level of spatial interactions. For instance, the relationship between distance and spatial interactions will change according to the beta exponent. If the value of beta is high (higher than 0.5), the friction of

distance will be much more important (steep decline of spatial interactions) than with a low value of beta (e.g. 0.25). A beta of 0 means that distance has no effects and that interactions remain the same whatever the concerned distance. Alpha and lambda exponents have the same effect on the interaction level. For a value of 1, there is a linear relationship between population (or any attribute of weight) and the level of interactions. Any value higher than 1 implies an exponential growth of the interaction level as population grows.

Another way of using a gravity model is to estimates the distance customers (casualties) will be willing to travel to go to a hospital after comparing quality, capacity and other factors. A rule of thumb is referred to as *Reilly's Law of Retail Gravitation*. The law assumes that people want to shop in larger towns, but their desire declines the farther and the longer the time they must travel to get to those places. Thus, larger towns draw customers from a larger trade area than smaller towns. The following formula estimates the maximum distance customers will travel to shop in a smaller town.

$$Maximum\ distance\ to\ smaller\ town\,(Y) = \frac{Road\ distance\ between\ towns\,(X)\,and\,(Y)}{1 + \sqrt{\dfrac{Importance\ of\ larger\ town\,(X)}{Importance\ of\ smaller\ town\,(Y)}}}$$

Similar analogy can be used in our case. Smaller town can be assumed to be a hospital and the larger town can be assumed to be a cluster. Then the maximum distance to a smaller town will depict the maximum distance a casualty from a cluster will be willing to travel to go to a particular hospital.

6.6 Visualization Modeling

The time immediately following a natural or man-made disaster can be a chaotic experience to any individual or community. This is evident with regards to the natural and man-made disasters, which have occurred, in recent years. A prior study has shown that in an earthquake situation, the information collected and dispersed in the first 72 hours is the most crucial, since most people still severely injured after this time are not likely to survive [6.6-1]. When a disaster spreads over an area, and causes thousands of casualties in a short time, it is nearly impossible to manage the disaster by human observation alone because of the massive amount of incoming information. In

fact, for large-scale disaster management, the first and most imperative step is the awareness of the situation in order to optimize the allocation of available resources. Therefore, the situation awareness is an essential role of a disaster monitoring or visualization system. With the paradigm of conventional geo-referenced display, this is difficult to achieve because its implementation is limited to the positioning of the corresponding graphic images. This usually results in thousands of scattered and cluttered icons on the display. The present research provides a monitoring environment through efficient data management and a user-friendly graphics interface that deals with the massive influx of data from the data fusion process. Furthermore, our technology adds time-aggregated data management that contributes to the visualization of more abstract and comprehensible graphics. This approach encourages tactical thinking and strategic control for a severely attacked area [6.6-2].

6.6.1 Previous Visualization Work in the Emergency Setting

Recently, researchers have been involved in data fusion for dealing with complex natural phenomena or algorithmic problems. A few visualization projects exist involving information fusion, such as a weather visualization application for emergency planning [6.6-3], NASA's wind tunnel simulation [6.6-4], and seismic activity visualization by the University of California at Irvine [6.6-5]. A research team from the University at Buffalo has also been working to create a battlefield visualization scenario using fused data [6.6-6].

Without fused data, there have been studies on emergency response, such as decision making aids by interaction through voice/gesture recognition completed by Pennsylvania State University [6.6-7], a framework for incorporating the many emergency response models for a simulation by Jain and McLean [6.6-8], and satellite/airborne image or video processing for the Kocaeli earthquake in Turkey by Ozisik and Kerle [6.6-1]. The works mentioned above implemented data processing and mapping in a two or three dimensional space, but do not provide abstract information from which a user could comprehend a situation. The work of Kim and Kesavadas considered effective icon/symbol generation regarding the viewer's visual recognition, which has been a cognitive issue in the military community since the advent of the digital display in the 1960's [6.6-9]. They have suggested *Automated Dynamic Symbology* by parameterization of graphic components connected to fused data. Their methodology was

considered as an appropriate feature for visualization of strategic information and remote-networked implementation, and served as a basis in the current research.

6.6.2   Issues and Our Approach

The present work is an achievement of a large scale fusion-based post-disaster simulation project. For simulation test, it specifically takes the post-earthquake situation data which occurred in Northridge, California on January 19, 1994. This was considered to be one of the worst earthquakes in the Los Angeles area in recent memory [6.6-10]. Our simulation model relies on the output from HAZUS [6.6-11] developed by the Federal Emergency Management Agency (FEMA). The fused data is currently being produced by a multidisciplinary group at our institution [6.6-12], [6.6-13]. The fusion output includes data of low-level fusion (identification) which covers roadway damage, casualties, hospitals, and ambulance routing and police information. As mentioned earlier, this approach usually causes information overflow to a viewer, such as different types of icons cluttering and overlapping on top of each other. Therefore, our system includes high-level fusion data (situation awareness and threat assessment), such as casualty clusters and its trends and prediction.

A challenge for the visualization of a post-disaster simulation is to deal with the substantial and complex data interface so a user can manipulate and retrieve desired information. Unlike the previous works, our application provides a visual display of emergency response data at run-time and through a networked simulation environment. The advantage is that it displays information as it is received and without delay. Commercially available Geographic Information System (GIS) software has been used for the data construction of urban terrain and traffic network. However, they are only capable of low-level fusion output, which is the functionality of location and identification. They do not have the functionalities for high-level fusion that demands many complicated tasks, such as large data-set manipulation, dealing with time aggregated data, and the capability of putting depth and height cues to the display.

To deal with such enormous and complicated data, we have developed the common class interface for sharing and synchronization of 2D and 3D graphics. It allows a user to see what resources are available, and where casualties are located in 2D, as well as 3D, which give a better understanding of the spatial relation amongst the resource objects. Each visual mode adopts

different application programming interfaces (API) and rendering environment, Windows MFC and OpenGL for example. Our work achieved implementation of both in one application combined in sync, which is described in the following sections.

6.6.3   Run-time Federation Interface

In this project, information communication is implemented with the HLA/RTI composed of several federates. A federate simply being one piece of the RTI which carries out a specific task, such as information regarding walk-in casualty or medical facility. The RTI allows for common variables to be changed by one federate and then updated in another based on the concept of "publishing" and "subscribing" to variables [6.6-14]. The following section explains the interface of the post-disaster simulation and describes how the data is passed to the visualization system.

The importance of HLA/RTI in the current simulation is that all simulation data is being generated at run-time. It should also be noted that for each federate it is crucial to know the current state of the situation at all the times. This is achieved through the report flow, or exchange interface, between federates (Fig. 6.6-1).
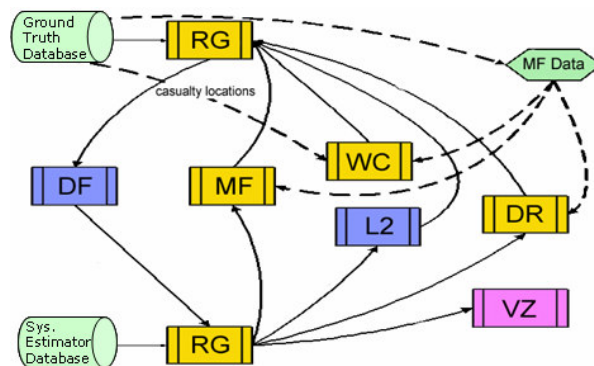
Figure 6.6-1.   Post-earthquake simulation data flow in the HLA/RTI network.

The Report Generator federate (RG) generates all reports for the simulation based on the output from the HAZUS earthquake model [6.6-10]. Data Fusion (DF, also called Level-1 fusion), then decides which reports are not repeated and fuses them into one report. This information provides the core to the simulation which the rest of federates use to carry out their own tasks. Level-2

fusion (L2) determines the time-stamped formation of casualty clusters from the casualties reported by RG. The figure shows the interaction between other federates, such as Walk-in Casualty (WC), Medical Facility (MF), Dispatcher and Router (DR), and Visualization (VZ).

Unlike the data relay in other federates, the visualization federate currently only takes information from the rest of the federation via Report Generator/Estimate Director. To create a highly abstract and robust runtime performance, we adopted the C# (C sharp) programming language on the Microsoft .NET platform [6.6-15]. Because the HLA/RTI is designed to support only C++ objects, it was critical to come up with a technique in order to integrate the two different programming codes. A solution was found for bridging the two executions not on the programming level, but the OS level. A system was devised for a directory to be setup where the C++ process stores all the report messages in ASCII format which can then be read by the C# process. Running in a call back loop, the directory watcher notifies the parser immediately after it gets the reports from the estimate director federate. The stored ASCII text is then parsed and stored in a shared memory for synchronization of the 2D and 3D graphics (Fig. 6.6-2).
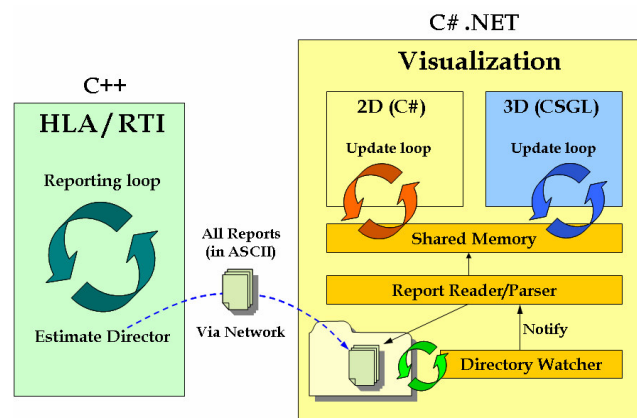
Figure 6.6-2   Data interface of the visualization federate between the HLA/RTI architecture.

6.6.4 Modeling of Visualization framework

The following section addresses how we achieved efficient data manipulation of a large data set and integration of two different kinds of graphics (2D and 3D) to provide sufficient options to a viewer.

6.6.4.1 Data Layers: The Visualization Pipeline

Layering is a useful way to organize massive GIS data. The United States Geological Survey (USGS) [6.6-16] offers an accurate depiction of the Northridge area which was used for our vector map. In addition to map viewing capabilities, a monitoring capability has also been developed. All the GIS data is layered at the bottom of the pipeline and rendered first. Built on top of this information is the federate data from the HLA/RTI. Placed on top of the federate information is the graphical user interface (GUI) developed for the simulation (Fig. 6.6-3). Data layering and implementation not only provided an easy-to-debug environment to programmers, but also produced better rendering performance.
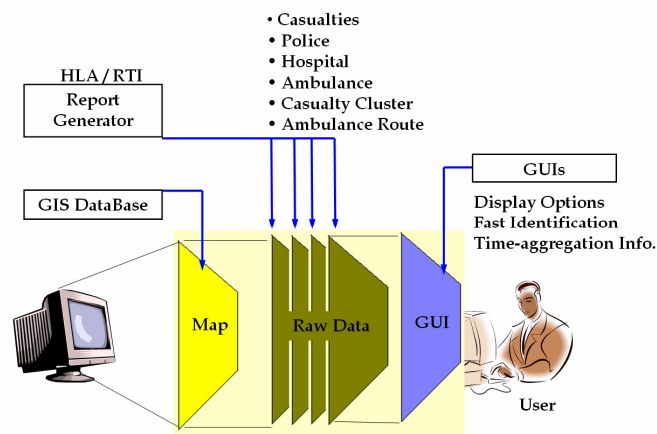
Figure 6.6-3   Data layers of the software architecture

Map Layer: Generation of Fast Vector Map

Two C# namespaces were created to handle the generation of the GIS map database for the display (Fig. 6.6-4). First, the *Geometry* namespace deals with the definition of all geometrical entities in a hierarchical structure. Another namespace, *GIS,* stores structured information of GIS objects for display. It uses a primitive class definition from the *Geometry* namespace with real GIS data. While all other GIS data was used from *.dxf files, a detailed road network is generated from a Tele-Atlas database file [6.6-17] which is helpful in determining ambulance routes and retrieving street information. Each of the links is represented by a 256 character line containing information like Link ID, Length, Street Name etc., which is further divided in a database of start and end points. Each end point is known as a node, which is stored with a

unique ID and Universal Transverse Mercator (UTM) geographic coordinate in the database. A *road link* class then stores all the information in runtime memory. Finally this map is stored in the *Common Container* class structure for further process.
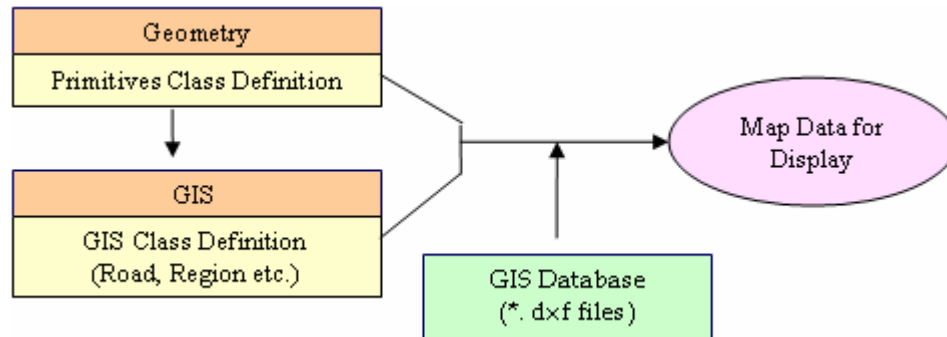
Figure 6.6-4.   Pipelines to Map Data Generation.

Raw Data Layer: Data Extraction from Incoming Reports

The Disaster-RTI defines all types of federate class definitions including casualties, police, ambulance, medical facility, roadway damage, ambulance route etc., in terms of their properties and functions (Fig. 6.6-5). Dynamic arrays have been built on top of this information to store their objects and reports in a hierarchical order. A separate class called *Visualization* is then defined for automatic generation of different color data which is used for each zip-code region in the Northridge area. This class is also used for identifying proper symbols for each federates' database which is stored in a symbol database.

The directory watcher class and its functions are defined in the *WinGUI* namespace which keeps a track of new reports in the directory folder. For every report from the RTI, raw data is generated for display by combining these files and having the data stored in the *Common Container* class for further processing.

GUI Layer: User Interface for Data Manipulation
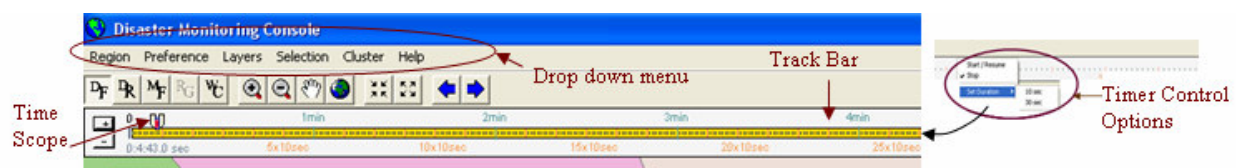
Figure 6.6-6**.** Linear clock and track bar for the manipulation of time-aggregated data.

What is controlled by the GUI is directly related to what is being captured in the federate layer underneath. The GUI includes a menu and tool bar control system with the capabilities of mouse and keyboard interactions. Since the simulation has been designed to run over a period of time, it is important to store accumulated federate data for further usage. The track bar shown in the GUI pane gives the user the ability to go back in time (Fig. 6.6-6). This way a person can see what progress is being made in specific areas over a period of time. It also has an option for time scope expansion, which allows for a longer duration of time to be rendered at once [6.6-18].

6.6.4.2 Integrated Simulation Architecture
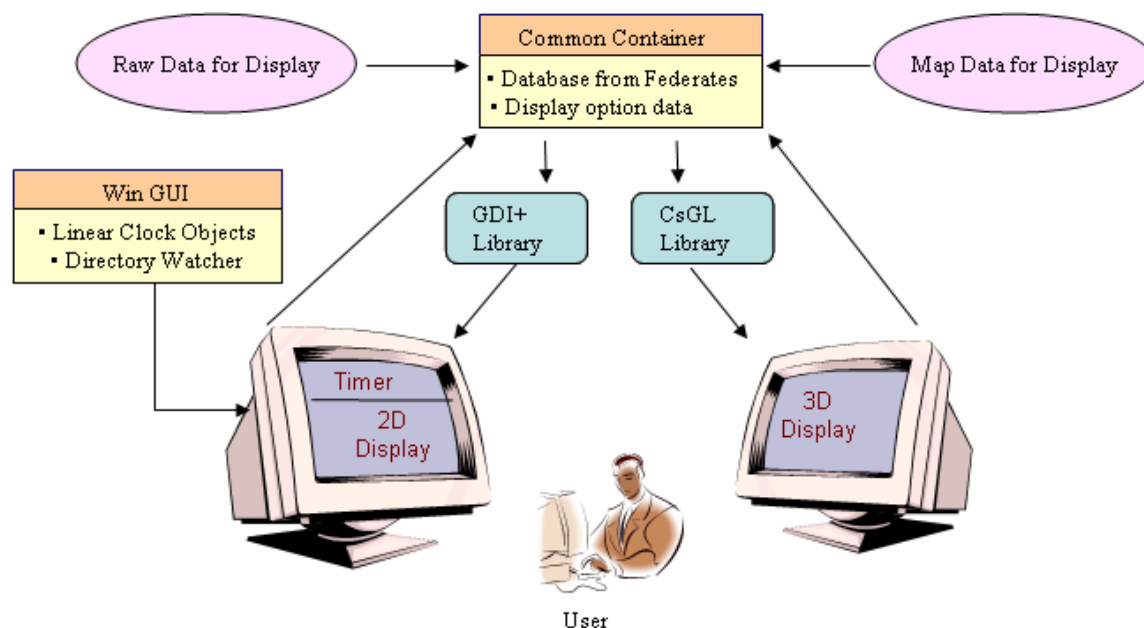
Figure 6.6-7   Integrated simulation architecture for multiple and different display of 2D and 3D.

We took advantage of multiple windows of two different graphics modes: two dimensional Windows MFC-based graphics, and three dimensional OpenGL-based graphics. For the synchronization in these multiple windows, display variables in either the 2D or 3D window need to be updated in the other window. Since it is impossible to swap references on two files due to restriction on circular dependency in C#, we developed a common class to access the shared memory and update the value (Fig. 6.6-7). All the corresponding objects for the federate report database (raw data) and map data are stored in the *Common Container* which can be

passed to the 2D and 3D windows for display as needed. Using the Graphical Device Interface Plus (GDI+) library for creating graphics in C#, all 2D graphics could be created in an effective manner [6.6-19]. The 3D display objects and navigation controls were then defined in the *3D display* class using the CsGL graphics library, which is simply an OpenGL wrapper for the C# programming language.

6.6.5 Run-Time Fusion Data Visualization

The following subsections report on the user interface and visual displays for low level fusion functionality (position and identification) and high level fusion functionality (situation awareness).

6.6.5.1 Multiple Display: 2D Fast Vector Map & 3D Visualization

The present research includes graphics controls and data manipulation found in common geo-referencing system. The zip code areas, road data and even the colors used, can be altered as needed by the user. This allows for the simulation to be viewed in a way that is least visually distracting so more attention can be focused on the resources (Fig. 6.6-8).

Figure 6.6-8   Manipulation of map environment: color (left), grayscale (middle), and single color.

The image of Fig. 6.6-9 (left) shows raw data from the report generator, such as casualties, police and ambulances. The image in the middle shows the corresponding view in 3D space. Even though 3D simulation has some disadvantages, such as unfamiliar interface and viewpoint control, the height and depth cues are an invaluable source of knowledge to a user [6.6-20]. This is especially true in a natural or man-made post-disaster simulation for an urban area. Height and

depth cues are crucial for buildings or other volumetric spaces. As mentioned, this 3D view works in sync with the 2D graphics, so that viewers can easily switch their attention at their own preference. In the future the 3D view could be integrated with 3D building, structures and landmarks and to provide more comprehensive urban casualty visualization.
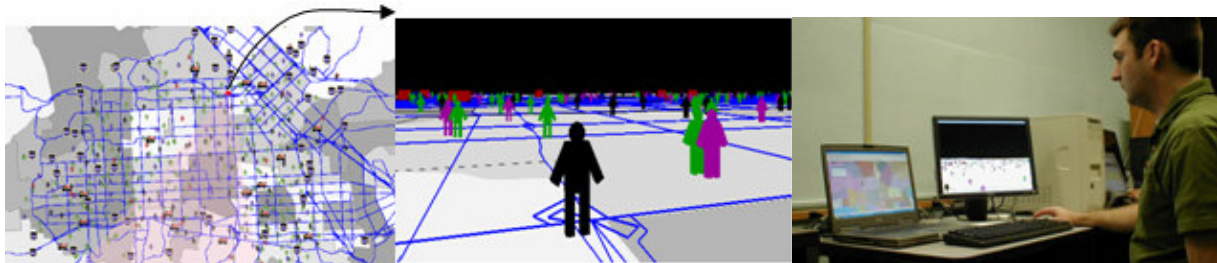
Figure 6.6-9   2D view of Northridge Area (left), corresponding 3D view (middle), and actual implementation of multiple display visualization system.

6.6.5.2 Display Control

The requirement of fast identification is essential for visualization of low level fusion. In order to be able to view only federate data which is desirable, a menu bar system was developed to switch certain data layers 'on' and 'off'. As mentioned, this avoids the problem of cluttering that occurs if too many reports are coming into the visualization federate at one time. A context menu has been included which can be accessed by right clicking in the display window and performs the same function as the menu bar (Fig. 6.6-10, left). In the 3D view, an independent viewing position was created to allow for an outside viewer to navigate throughout the 3D environment to give a better sense of realism. We took advantage of the ability to change the scale and size of the icons in 3D, so based on the size of the data, a user can get a better understanding of the disaster scenario.

Figure 6.6-10   Fast data manipulation and identification: resource context menu (left), and pop-up window for object identification and the visualization of its trend (right).

6.6.5.3 Quick Identification

The information specific to each casualty is stored over time and is available to the user by simply hovering over a casualty icon on the screen. A pop-up window then appears which shows the pertinent information (Figure 6.6-10, right). Since it would be impossible to track each casualty if thousands were present the information regarding casualty clusters will be more useful for time aggregation analysis (section 4.4). Therefore, specific areas can be monitored during the simulation.

All ambulance route reports in the simulation come with road links Id's, which define the routes for the ambulances. It is possible to store all road link Id's in a report database and use that Id with a search algorithm to retrieve all related information by accessing a link database. However, this will increase runtime computation, which further lowers the performance. An easy and compatible solution to the problem is to store the object index in a container of road links objects. In this way, a search algorithm only needs to be run once when a report is received. The data can then be retrieved very quickly by just accessing the container by object index. We have used the binary search algorithm to find index of road link object in a large container. This algorithm needs $O(\log N)$ iterations in worst case to find an object in a container, which is acceptable for database of about 30,000 objects. To make efficient use of runtime computer memory it is desired that all the information regarding road link data be stored only once so that it can be retrieved as needed by accessing its unique Id (Fig. 6.6-11). Each file from the RTI contains multiple routes at a time which show a separate route for each ambulance. A hierarchical dynamic array structure has been created to store a report time and multiple route information at that time for each report. Each route can then be accessed by road link index in that array database.
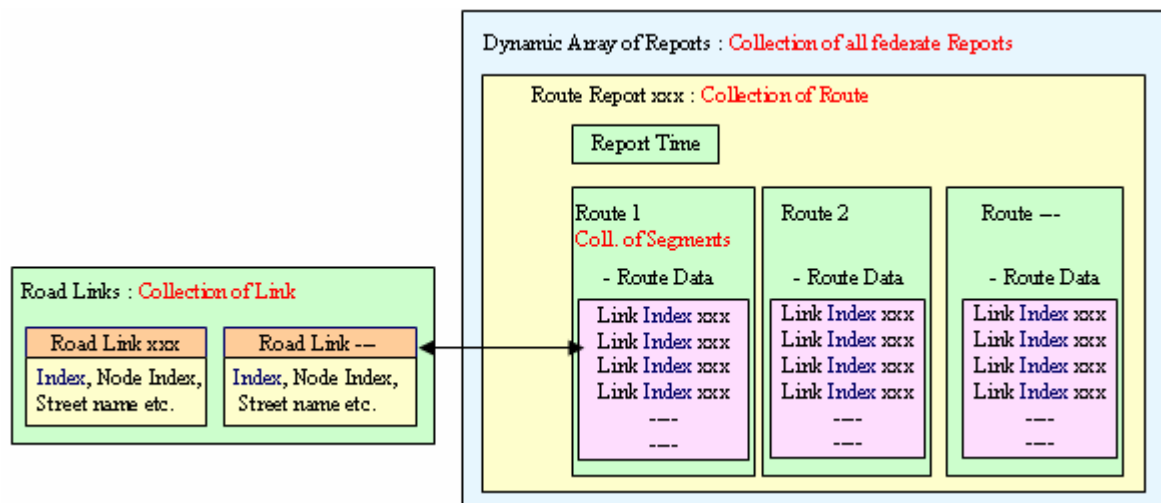
Figure 6.6-11 Database design for information storage.

As shown in Fig. 6.6-12, ambulance paths can then be highlighted on the screen by merely hovering over them with the mouse. This allows a user to get a clearer understanding of the path and the specific road it may be located on. In order to optimize performance for graphical output, only the current path with the mouse tip is refreshed during mouse hovering. Therefore, the entire display does not have to be refreshed which can distract the user. To catch this mouse event, a small rectangular region is defined around each road link by considering its spatial orientation in the 2D window. As soon as any region catches the mouse tip the searching algorithm will exit with the current route index from the main report index. Based on the current ambulance route index a graphical region is created by joining small oriented rectangular regions of road links which will make the ambulance route. Finally, the system invalidation function is called to refresh only that region in the graphical output window. A backup route index is also stored in run-time memory which is helpful in refreshing the same route region again when the mouse tip moves out of that region.
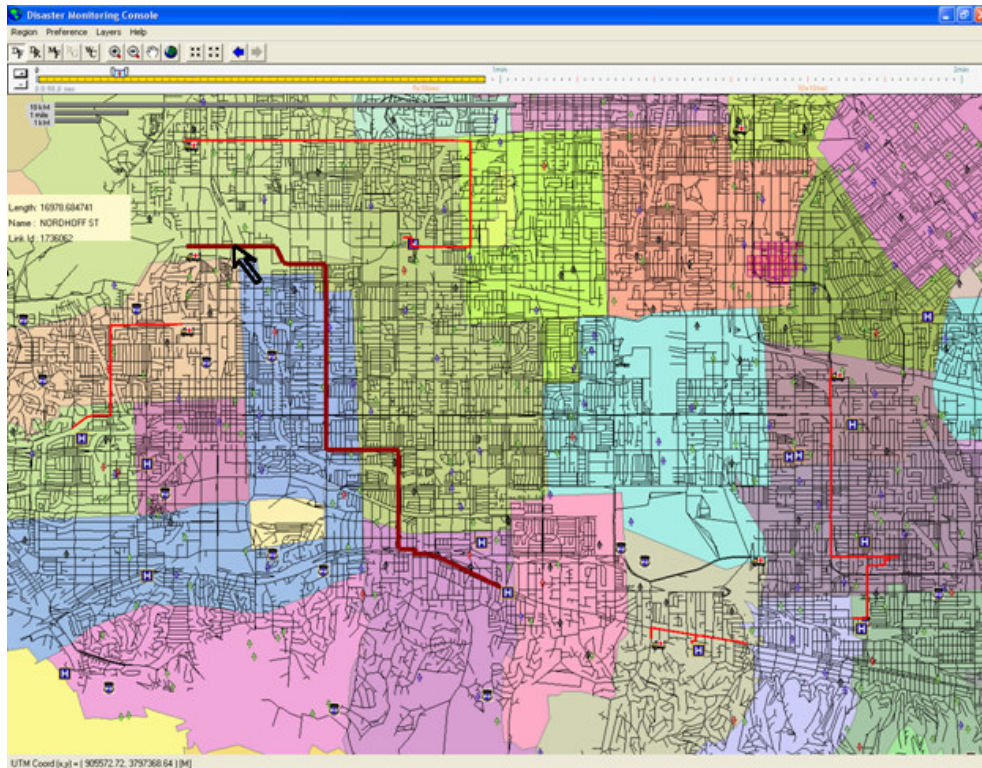
Figure 6.6-12   Ambulance route display in an urban area of Northridge region.

6.6.5.4 Clusters: Regional Information

A cluster can be defined as the extraction of large amounts of data and displaying it as a group in a vast database system. In a post-disaster simulation the need for dealing with large sets of data and comprehending abstract information is a key to understanding situation awareness. Work on clustering with fused data can be found in applications for a battlefield [6.6-21] and the visualization of large datasets [6.6-20]. In our simulation, the casualty cluster information is produced by the Level – 2 fusion federate [6.6-22].

Formulation of Cluster and Boundary Formation

As an approach to cluster visualization, the entire Northridge area was divided into cells with horizontal and vertical grid data. Fig. 6.6-13 (1) shows a cluster with a group of cells which defines the cluster. The outlier information is helpful in the identification of the area the cluster covers, as well as, interpolating intermediate shape information while morphing. Cell information is also effective in the quick identification of areas with the highest emergency

within a cluster. In the figure, the opacity of a cell represents the number of casualties present and allows for quick identification of the most troubled spots. It is also possible to see clusters when only considering one level of severity by using the context menu. In addition to the cluster, a boundary can help a viewer recognize the shape of the cluster. With only boundary information, the trend of the regional shape can easily be comprehended (Fig. 6.6-13, right).
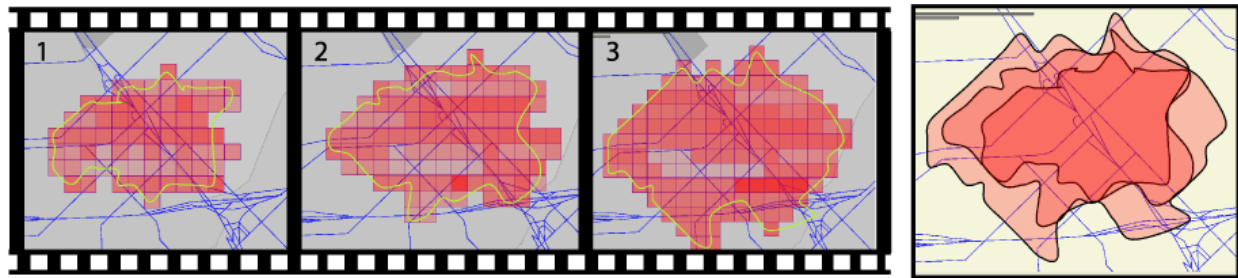
Figure 6.6-13   Situation awareness by cluster morphing. The casualty clusters between two discreet time steps (1 & 3) are generated by interpolation (2). The expanding trend of such cluster boundary is shown for better understanding of the situation over time (right).

Dynamic Visualization by Morphing

It is difficult to capture the trend of a situation from one specific time to another. In most cases, the user has to depend on his/her memory to relate the states. In order to provide a way of seeing how the clusters change with respect to time, we incorporated morphing into the simulation. All the visualization elements, such as position, color and shape were stored at each discrete time step. Corresponding discrete states were then interpolated to create the morphing. However, the associated cluster ID may change from one point to the next which makes it difficult to morph the clusters directly. Therefore, morphing was carried out on a cellular level. This way individual cells, which comprise the clusters, may appear and disappear from subsequent states to help inform the user. Using the morphing interface, a user can keep track of a cluster and cells, comprehend the trend, and have effective awareness of the situation (Fig. 6.6-13).

6.7 Secondary incident modeling

High level data fusion is the process of filtering multiple data streams through a reasoning process informed by domain knowledge in order to construct a deep understanding of a given

situation. This requires estimating and then mapping its current configurational properties and predicting their future implications. The main task of the reasoning process is to parse the set of all feasible hypotheses to discover those most consistent with the data and with prior knowledge. These hypotheses take the form of assertions about the current situational state, such as the identities and attributes of significant objects and relationships, together with assertions about the uncertain future evolution of these objects and relationships, the likelihoods of these scenarios and their relevance to the decision-makers.

A fundamental issue in the design of a high level data fusion system is the range of hypotheses it is willing to entertain. If the hypothesis set is tightly constrained *a priori*, the computational complexity of the search for hypotheses matching a given data set is reduced, as are the system hardware requirements, network bandwidth and data fusion product report latency. On the other hand, a decision support system cannot discover what it is unwilling to entertain. So if unexpected events occur, those which were deemed too unlikely to be considered at design time, they will not be discovered and properly identified.

The most important class of such unlikely but potentially significant events in the context of data fusion for emergency response are secondary incidents [6.7-1]. Secondary incidents have three defining attributes:

1. They are caused or facilitated by the primary disaster event

2. They create an additional hazard to life and/or property

3. They have implications for future decision-making

For instance, a gas line rupture secondary to an earthquake, a fire secondary to a plane crash, or the Khamisiyah release during Desert Storm [6.7-2].

6.7.1 Secondary events in emergency response

The primary disaster event will engender its own chain of likely events requiring emergency response. For these a tightly bounded universe of discourse of plans, hypotheses, rules, events and entities can be enumerated in advance and managed as the scenario unfolds. The danger in

this strategy is that events narrowly interpreted will be misinterpreted, with the real nature of the situation hidden until the consequences are unacceptable.

The secondary incident challenge is considered in this project in two settings. With respect to the general design principles of data fusion systems for emergency response which are "secondary incident aware," methodological considerations and the choice of a compatible reasoning system are discussed in Section 6.9.

In order to exercise the selected approach, a secondary incident generator was built into the earthquake test bed DIRE. That generator is the subject of this section. The scenario simulated is one of a secondary hazmat incident caused by the rupture of a stationary chemical storage tank, or the crash of a hazmat transporter and the rupture of its tank. It is assumed that the subsequent hazmat plume is relatively odorless and colorless, but potentially lethal.

The challenge in this scenario is to correctly identify the occurrence of a secondary incident subsequent to the primary earthquake event, and to estimate its most important attributes such as location and time of release, quickly enough to save lives. The problem is that many individual casualties, and clusters of casualties, which are consequent to the the primary earthquake event will be being reported at the same time. It is not clear how to discriminate the secondary casualties from the primary in order to discover the occurrence of the secondary incident, how to reason about this event in the confusing context of the primary event.

The use of data warehousing techniques [6.7-3] or data mining [6.7-4] in this process can be of considerable value. In either case, a principal question to be answered is whether the reasoning process is based on representations of uncertainty that are symbolic or numeric. Non-monotonic logics and rule-based architectures are typical choices on the symbolic side, and Bayes Nets, connectionist approaches or genetic algorithms on the numeric. Here we choose to employ a hybrid of the two approaches: a belief-based argumentation scheme. This is an abductive reasoning approach employing rules to define evidentiary elements (arguments), and quantitative beliefs to assess the strength of these arguments. For a given hypothesis all arguments corroborating or refuting that hypothesis are marshalled, and using the Dempster-Shafer rule of combination, combined to determine belief in that hypothesis. This approach is detailed in [6.7-

5] and is discussed in Section 6.9 of this report. In the remainder of this section, the instantiation of this scenario in the secondary event generator in DIRE will be discussed.

6.8 Secondary incident modeling

The model for fluid dynamical atmospheric dispersion of toxic material we chose is the basic Gaussian model [6.7-6]. Given mass M of the toxic material entering the atmosphere at $(x,y,z)=(0,0,0)$ at time $t=0$, and given a constant wind with components v_x and v_y, the concentration of the material at (x,y,z,t) is approximated as

$$C(x, y, z,t) = \frac{M}{8(\pi t)^{3/2}(E_x E_y E_z)^{1/2}} \exp\{\frac{(x-v_x t)^2}{4E_x t} - \frac{(y-v_y t)^2}{4E_y t} - \frac{z^2}{4E_z t}\} \quad [1]$$

The E's are dispersion coefficients which are determined empirically. This model predicts that the mean (point of highest concentration) of the plume propogates from the release point (0,0,0) with the wind, and the variance (spread) of the plume in all directions increases linearly with time.

There are obvious shortcomings of this model. There is no ground plane at $z=0$ to prevent diffusion below the ground. The rate at which the plume spreads out in this model does not depend on the wind velocity (v_x, v_y), ie. turbulent mixing is not taken into account. Removal of material by condensation, chemical or photoconversion is not considered. Atmospheric layering, for instance inversions or stratification, is not modelled. Nor is the topology of the ground over which the plume propogates, a factor of particular concern in urban settings. But the goal of this generator is not to produce a highly resolved and accurate model, rather to generate a plume generally consistent with first-order fluid-dynamical and meteorological laws.

Since we are only interested in the concentration at ground level, the form of this model employed is the 2-d version of the above,

$$C(x, y,t) = \frac{M}{8(\pi t)^{3/2}(E_x E_y E_z)^{1/2}} \exp\{\frac{(x-v_x t)^2}{4E_x t} - \frac{(y-v_y t)^2}{4E_y t}\} \quad [2]$$

- 152 -

where we have set z=0, and overloaded the definintion of the concentration function C.

Preparing to discretize the problem, define the concentration C at (x,y,t) due to a mass M released at (x_o, y_o, t_o), assuming that no additional material has been or will be released.

$$C(x, y, t; x_o, y_o, t_o) = \frac{M}{8\pi(t - to)^{3/2}(E_x E_y E_z)^{1/2}} \exp\{\frac{((x - x_o) - v_x t)^2}{4E_x(t - t_o)} - \frac{((y - y_o) - v_y t)^2}{4E_y(t - t_o)}\} \quad [3]$$

Again we are overloading the C-function definition. Now let t-to = Δt, x-xo = $i\Delta x$, y-yo=$j\Delta y$. Then the concentration in a grid cell one time step Δt later due to the dispersion of a unit concentration worth of material from the grid cell (i,j) cells away is

$$h(i, j) = \frac{\Delta x \Delta y \Delta z}{8\pi\Delta t^{3/2}(E_x E_y E_z)^{1/2}} \exp\{-\frac{(i\Delta x - v_x \Delta t)^2}{4E_x(\Delta t)} - \frac{(j\Delta y - v_y \Delta t)^2}{4E_y \Delta t}\} \quad [4]$$

Here we have used the fact that *M* is the product of the volume of the grid cell and its concentration.

Next suppose at time *k* the current concentration map is *C(i,j,k)* given by [2] with discretizations $t=k\Delta t$, $x=i\Delta x$, $y=j\Delta y$. Then the concentration at grid cell (m,n) at time k+1 can be found by convolving the concentration map at time *k* by the impulse response [4],

$$C(m, n, k + 1) = \sum_i \sum_j h(m - i, n - j)C(i, j, k) \quad [5]$$

The sum in [5] is taken over all grid cells (*i,j*), and [5] is evaluated for each grid cell (*m,n*). In computing [5], the impulse response *h(i,j)* is precomputed and stored, it is the same for each *k*. Note that the wind velocity (v_x, v_y) can change arbitarily over time. The assumption is that over a single timestep Δt it is constant over the entire grid and over the entire time step, but not that it is necessarily constant time step to time step.

Concerning the model limitations, the lack of a ground plane and the removal of material simply scale the concentrations, and since we are going to set injury thresholds we can accomodate that easily. The lack of turbulent mixing does lead to oversimplified results, but if we limit ourselves to low wind speeds this is not a serious problem. Test runs show that in the case of a continuous point release, a plausible "teardrop" shaped plume is formed which is aligned with the prevailing wind direction, and this plume spreads and dissipates with time.

6.7.2. Creating secondary incident casualties

The primary earthquake event casualties are laid down in DIRE as the simulation run begins, based on HAZUS casualty estimates. A secondary Hazmat event occurs at a random time and location within the first 4 hours following the primary earthquake event. The basic premise of adding secondary incident casualties is that it is done on a dose-response basis. The concentration profile is integrated over time to create a spatially distributed dosage. A fraction of the total population which will sustain related injury within a given census tract is then determined based on the dose and the response to that toxic material. That fraction is multiplied by the population of that census tract and that number of new casualties is laid down randomely within the census tract.

Specifically, the entire Northridge area is divided into grid squares that cover the 121 square miles of the disaster area. The gridpoints are about 100 meters from one another. The concentration at each gridpoint is updated according to the precomputed text files in the plume folder determined by the dispersion model [5]. Casualties are added only around each gridpoint being created.

If the scaled average concentration of the current grid cell since the plume began is greater than a threshold value specified in the .ini file, then there are new casualties added. The value of pernew is the percentage of non-casualties within the region around the gridpoint that will become casualties. This percnew is calculated using a logistic transform. The basic premise of the model is that the percentage of non-casualties that become casualties increases as the average concentration increases above the threshold set in the .ini file. The average concentration was used as the basis for adding casualties so as to smooth out the additions over time. For example if a gridpoint suddenly had a higher concentration, then the number of casualties would gradually

increase over time, not necessarily all at once. One other parameter in calculating the percnew is newcasmult. This is a multiplier in the .ini file that determines a base rate of adding casualties. The next lines for searchrect and newpt are determined for the placement of new casualties. I then calualte the total number of new casualties to add. This is done with the line around 2628

Numnew is then calculated by taking the population of the 100 meter by 100 meter square around each gridpoint. This is the population of the census tract divided by the census tract area times the area of the square (.01). The population that is already casualties (casrecs.count) is then subtracted. This number is multiplied by the percentage that become new casualties, producing the number of new casualties. The next loop adds the casualties into the casualty layer of the database. The last main part of the update plume routine, updates all of the casualties around the current grid point with the new concentration.

6.7.3 Example plume illustrating extended source release and wind shift

The following sequence of images were acquired from the visualization federate during a DIRE run in which a toxic release occurred near the center top of the field of view and the wind initially blew at 5 Km/h from the north. The human figure icons represent reported primary event casualties of various severities.

t=10min
Steady release continues
Plume growing downwind

t=15min
Steady release continues
Plume growing downwind
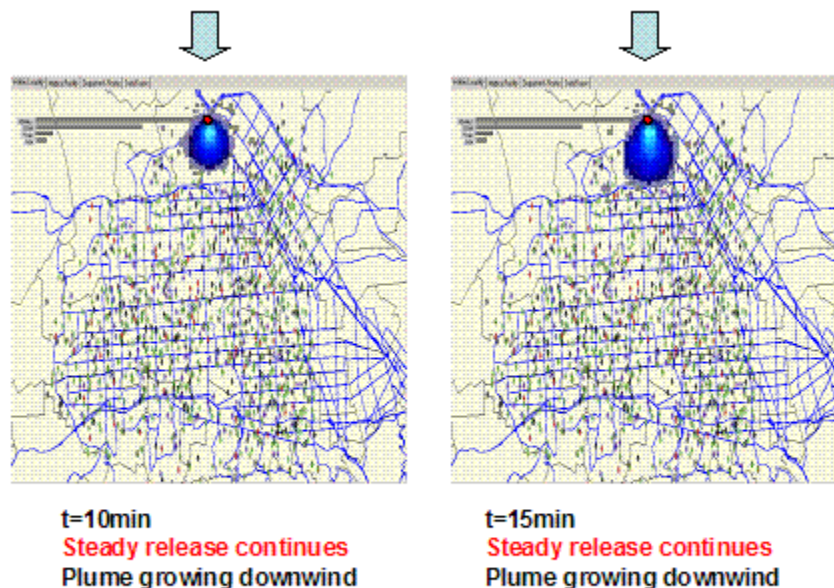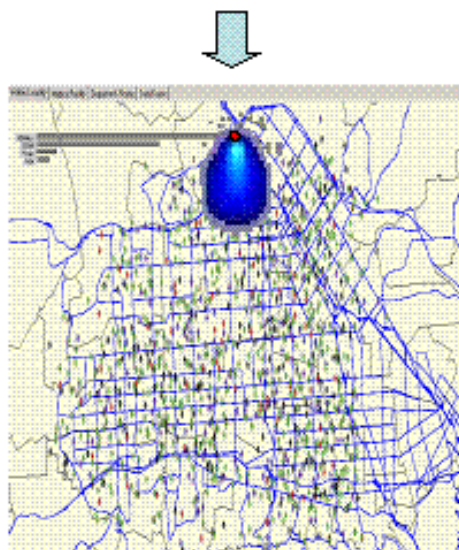
t=20min
Steady release continues
Plume growing downwind
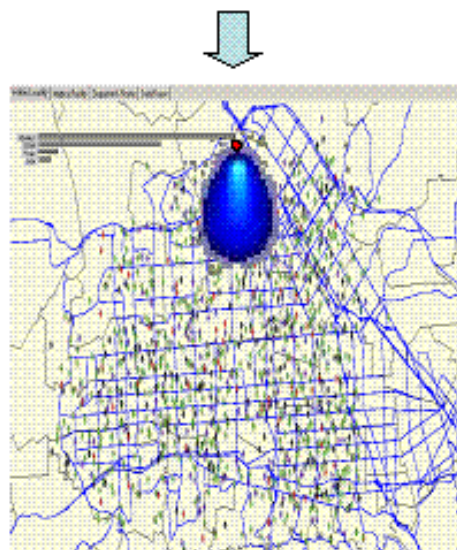
t=25min
Steady release continues
Plume growing downwind
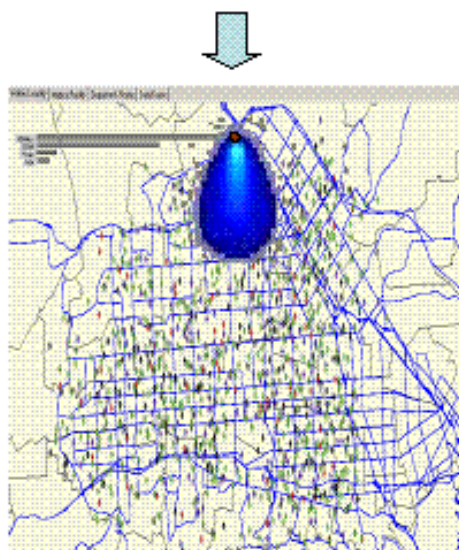
t=30min
Release attenuating
Plume growing downwind
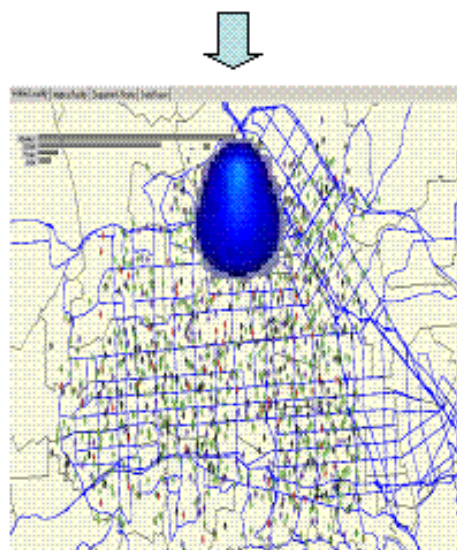
t=35min
No additional release
Plume detaching
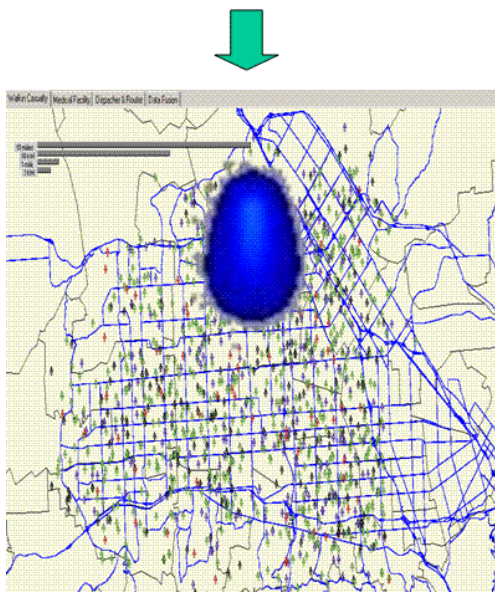
t=40min
No additional release
Plume moving downwind
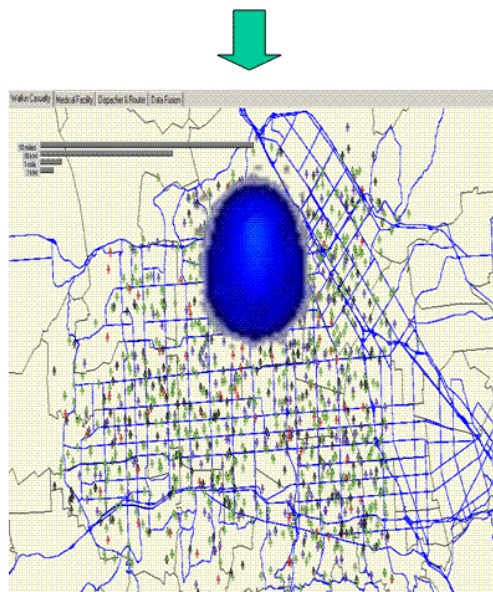
t=45min
No additional release
Plume moving downwind
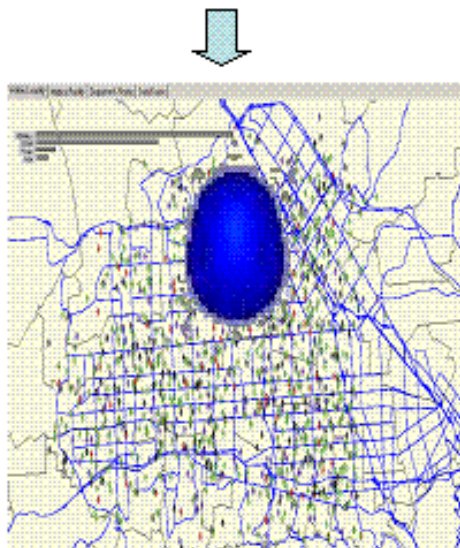
t=50min
No additional release
Plume diffusing, dispersing

t=55min
No additional release
Plume diffusing, dispersing

AFOSR F49620-01-1-0371

t=60min
No additional release
Prevailing wind shift

t=65min
No additional release
Plume moving downwind
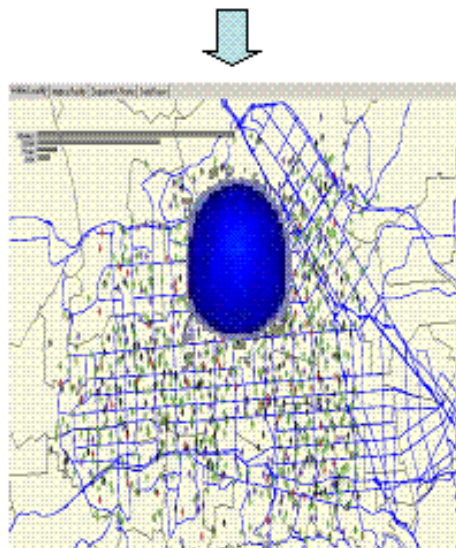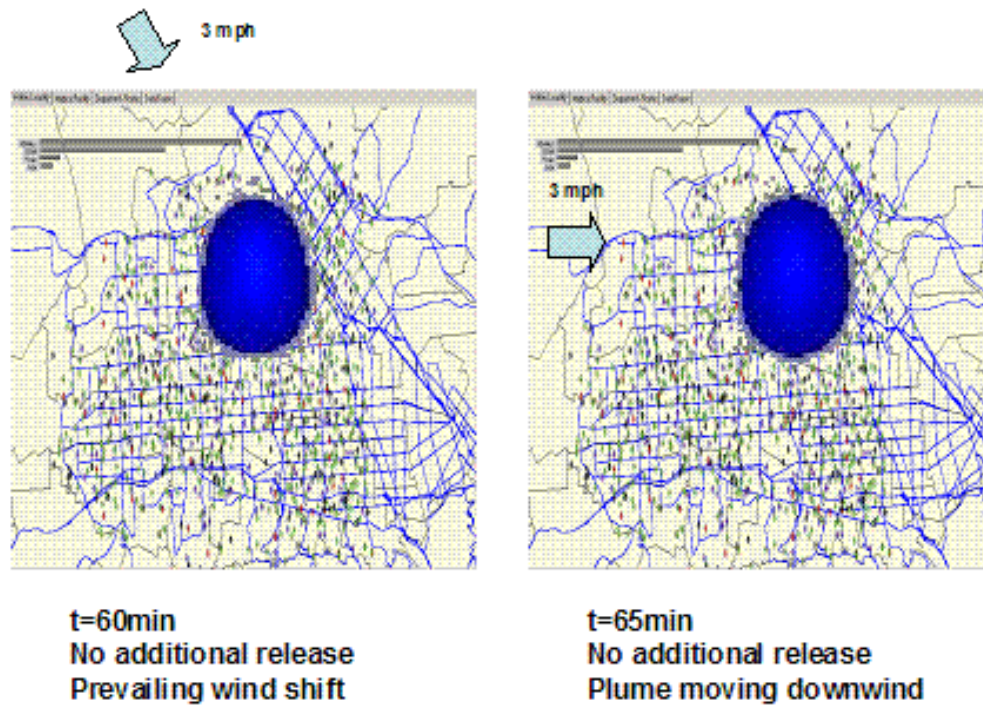
6.8 Distributed L0/L1 Fusion

In this section the detailed design of the Disaster Assessment Level 0/1 Distributed Fusion processing will be presented. The inputs are the Main Reports data stream which contains reports from each jurisdiction communications center and a mutual aid jurisdiction (MAJ) in a different order due to communications and processing delays. The outputs are the consistent tactical pictures (CTP) from each jurisdiction each containing the 5 data base files, one for each Level 1 entity type. The performance metrics are the accuracy, timeliness, and consistency of these jurisdiction CTPs.

Section 6.8.1 defines the distributed fusion node network for 2 jurisdiction communications centers and the intelligence preparation for each source to generate the attribute class confidence vectors given an attribute declaration. Section 6.8.2 defines the fusion node processing for each of the 5 fusion node types for each jurisdiction center in the distributed fusion network.

6.8.1 Distributed Fusion Node Network Design

The distributed fusion node network design is shown in Figure 6.8-1. For each communications center (CC) this is a Level 1 sequential fusion network that fuses call for service and first responder data with the "global" CTP as it is received. The dispatchers at each communications center restrict their view of the CTP to within their own jurisdiction boundaries as desired. The Figure shows the separate police location data input fusion nodes that will be implemented with replacement in since police reporting rates are not sufficient for filtering. The same will be done for ambulance location inputs, though not shown. The other police and ambulance report types pass through one type at a time into the CC sequential fusion node process as shown.

In both communications centers (i.e., Jurisdiction ID = 1 or 2) and the mutual aid jurisdiction (MAJ) (i.e., Jurisdiction ID = 3) will be integrated from the beginning of the scenario(s). The communications center reporting assets will be the police, ambulance, civilian, and hospital sources described above. The only MAJ reporting assets will be its shared officers and ambulances. will fuse data from one jurisdiction that is reported to another jurisdiction (e.g., a civilian on the boundary reporting on an incident across the street boundary) to remove duplicates and update with improved information.

All centers will share and fuse all of the police, civilian, ambulance, and hospital simulated report data. This architecture presumes that the bandwidth of the communications between Communications Centers and computational power is sufficient to enable all calls for service report sharing.

Figure 6.8-1: Selected Distributed Fusion Node Network

The pre-scenario processing of confusion matrices and a priori class vectors will be performed as described next. The approximation to the a posteriori severity class vectors from the confusion matrix will be made as follows:

$$P(C|S)= P(D|C) \ P(C)/ \ \Sigma_K \ \{ \ P(D_K \ |C) \ P(C) \ \}$$

for each class C where P(D|C) are the elements in the confusion matrix column for the given declaration. Before making this calculation the elements of the column vector are perturbed by the "confusion perturbation factor" listed in the DAUpdate.ini file. The factor is multiplied by the confusion matrix element to determine the bounds around the column elevation over which a uniform draw is made. The baseline perturbation is ±10% of each of the values (i.e., baseline confusion matrix perturbation factor (CMAPertFactor) = .1) which is then normalized to sum to one. These calculations are done once when each confusion matrix is loaded for a scenario. The a priori vector, P(C), will also be perturbed in the same way (i.e., baseline a priori perturbation factor (APPertFactor) = .1) once when each a priori matrix is loaded. If a class 0 is received, then the a priori vector is used.

For example, with the police casualty severity confusion matrix in Figure 6.8-2 and a police report of severity 3, the a posteriori class severity class vector for the a priori casualty severity class vector = [.3, .3, .2, .2] is as follows:

$$P(C=1|S) = P(D=3|C=1)/ \Sigma_K\{ P(D =K |C=1) P(C=1)\} = .03 * .3/ [.03*.3 + .07*.3 + .8*.2 + .38*.2] = .009/[.009+.021+.16+.076] = .009/.266 = .034$$

So, P(C|S) = (.034, .079, .601, .286). Since these values are very rough to begin with, this rough approximation should be sufficient. Note for comparison that if the a priori was uniform a sample perturbation would be P(C|S) = (.02, .06, .62, .3).

In summary, all the confusion matrices will be preprocessed using the scenario a priori class vector to generate these a posteriori severity class vectors. Perturbations will be added to these a priori and a posteriori vectors before each scenario is run.

Truth Class	Declaration Class 1	Declaration Class 2	Declaration Class 3	Declaration Class 4	Declaration Unknown 5	Total Prob.
1	0.75	0.07	0.03	0	0.15	1
2	0.15	0.60	0.07	0.03	0.15	1
3	0	0.05	0.8	0.05	0.10	1
4	0	0.02	0.38	0.50	0.10	1

Figure 6.8-2: Sample Severity Confusion Matrix

6.8.2 Fusion Node Processing

The data fusion processing will follow the Data Fusion and Resource Management (DF&RM) Dual Node Network (Dual Node Network) architecture. The distributed fusion call for service

data base (CDB) will be generated at each of the 2 jurisdiction sites. Each jurisdiction fusion network will consist of 5 level 1 fusion segments corresponding to casualties, emergency vehicle location (i.e., police and ambulance), facility (i.e., hospital and emergency facilities), transportation link delays, and bridge damage. Figure 6.8-3 describes each of the fusion node component functions. The tailoring of the fusion node networks for each of these segments at each jurisdiction site and the tailoring of each of the 5 fusion node types for in each segment is described below. The three general fusion processes in each fusion node are as follows:

1. Data Preparation: Propagate the current CTP to the current time.

2. Data Association: This function is composed of hypothesis generation, hypothesis evaluation, and hypothesis selection. Hypothesis generation determines all the feasible report to CTP associations using gating schemes on location, ID/attributes, and parameters. Hypothesis evaluation computes the score for each feasible association as the product of the location score, parameter score, attribute score, and a priori score. Hypothesis selection searches through the feasible association matrix to select, in this case, the highest scoring of all report association hypotheses.

3. State Estimation: The CTP entity location, parameters, and attribute (e.g., injury severity) states are updated.

	DATA ASSOCIATION	
PRIOR DATA FUSION NODES & SOURCES	DATA PREPARATION	HYPOTHESIS GENERATION / HYPOTHESIS EVALUATION / HYPOTHESIS SELECTION	STATE ESTIMATION & PREDICTION	USER OR NEXT FUSION NODE

SOURCE SENSOR STATUS

RESOURCE MGT CONTROLS

- **DETECT AND RESOLVE DATA CONFLICTS**
- **CONVERT DATA TO COMMON TIME AND COORDINATE FRAME**
- **COMPENSATE FOR SOURCE MISALIGNMENTS**
- **OUTPUT RECONCILED SENSOR DATA**

- **GENERATE FEASIBLE & CONFIRMED ASSOCIATION HYPOTHESES**
- **SCORE HYPOTHESIZED DATA ASSOCIATIONS**
- **SELECT, DELETE, OR FEEDBACK DATA ASSOCIATIONS**
- **OUTPUT SELECTED DATA ASSOCIATIONS & CONFIDENCES**

- **ESTIMATE/PREDICT ENTITY STATES**
  - KINEMATICS, ATTRIBUTES, ID, RELATIONAL STATES
- **ESTIMATE SENSOR/SOURCE MISALIGNMENTS**
- **FEED FORWARD SOURCE/ SENSOR STATUS**
- **OUTPUT ENTITY & AGGREGATE STATE ESTIMATES**

Figure 6

6.8.2.1 Casualty Fusion Segment

The casualty fusion network will receive the police, ambulance, and civilian casualty reports as available. The data will be processed one source type and one report at a time. The casualty fusion segment is composed of a sequence of such fusion nodes as reports arrive over time as described in Section 2. The processing in each casualty fusion node is described next.

Casualty Report Data Preparation

The attribute declarations will be converted to attribute confidences using a table look-up of the vectors computed prior to running the current scenario as described at the end of Section 2. No coordinate transformations are used since the data is already in a common coordinate system and there are no misalignments. Also, no time propagation is done since casualty and other entity motion is not modeled here.

Casualty Report Data Association

During data association the casualty fusion node will associate each report from a civilian, police, or ambulance with the existing active casualty data base (CDB) and then in state estimation update the CDB. The location, CASID, severity, age, race, sex elements of each

report will be used as available to gate and associate the casualty with a CDB call for service. Data association is composed of hypothesis generation, evaluation, and selection.

Hypothesis Generation

Hypothesis generation will use simple gates to eliminate numerous CDB casualty track as not feasibly associated with the given casualty report. These gates will use all available elements of the report and CDB entries to include

1. If the report has a CASID, search the CDB for the matching non-zero CASID. If find a match then this CDB call for service is the only feasible and go to state estimation to update the CDB call for service

2. Gate out all CDB call for service marked as "picked-up", "walk-in", or "dropped-off"

3. time: gate out user selected old CDB calls for service (baseline: 10,000 seconds)

4. casualty location: distance (i.e., Euclidean) between the report and the CDB track in meters must be less than user selected number of standard deviations where the distance sigma = $[P_x + R_x]^{.5}$ using the independence of the errors and $P_x = P_y$ & $R_x = R_y$ (baseline: gate is 5 sigma where civilian the $R_x = R_y$ square root (i.e., measurement sigma) is 20 m and the responder measurement sigma is 10 m)

5. casualty severity: gate out CDB calls for service whose casualty severity confidence is below user specified threshold for reported casualty severity declaration (baseline: .01)

6. casualty age: gate out CDB calls for service whose casualty age confidence is below user specified threshold for reported casualty age (baseline: .01)

7. casualty race: gate out CDB calls for service whose casualty race confidence is below user specified threshold for reported casualty race (baseline: .01)

8. casualty sex: gate out CDB calls for service whose casualty sex confidence is below user specified threshold for reported casualty sex (baseline: .01)

The attribute gates are not applied if a 0 (i.e., no report) is received nor if an unknown type is received. The output from hypothesis generation is the set of feasibly associated CDB calls for service that pass all the above gates. Go through data base one time.

Hypothesis Evaluation

The process will apply a MAP a posteriori Bayesian approach to evaluate the association confidence of each feasible CDB call for service with the given report. For reports where CASID is available only it will be used for association. When the CASID is not available, these deterministic MAP scores for each the feasible CDB call for service are based upon casualty location, injury severity, race, sex, age and will be compared to the no association score. The no association score contains 2 terms. Namely, the probability that the report should initiate a call for service in the CDB and the probability that the highest confidence CDB call for service is not the given report and should be retained.

The MAP score for association is the expansion of the following:

$$max\ P(H|R) = max\ \{P(R|H)\ P(H)\} = max\ \{P(Y|H)\ P(A|Y,H)\ P(H)\}$$

$$= max\ [\Pi_T\ \{P(Y(S)|Y(T),H)\ P(C(S)|C(T),Y(T),Y(S),H)\ P(Z(S)|Y(S),\ Y(T),\ C(T),C(S),\ H)$$
$$\Pi_A\ \{\Sigma_K[P(K_A|Z(T),Y(T),C(T),H)\ P(K_A|Z(S),Y(S),C(S),H)\ /\ P(K_A|Y(T),Y(S),C(T),C(S),H)]\}$$
$$P(H)\}]$$

such that

1. the maximization is are over all association hypotheses H,

2. H is the set of feasible association (or non-association) hypotheses,

3. R are the source report and CDB calls for service track data, Y is the set of all kinematics, A is the set of all attributes from both,

4. the first product is over all independent track-to-truth hypotheses,

5. the second product is over all the noncommensurate independent attributes,

6. the sum is over all the possible classes of each attribute,

7. $Y(T)$ are the call for service kinematics & $Y(S)$ are the report kinematics,

8. $C(T)$ & $C(S)$ are commensurates from the call for service and report data

9. K are the elements of the disjoint casualty attribute class tree,

10. $Z(T)$ & $Z(S)$ are noncommensurate attributes (i.e., independent when conditioned on the object class K) from the call for service and report, respectively,

11. $P(H)$ is the a priori confidence in the association hypothesis.

The $P(A|Y,H)$ noncommensurate attribute term is expanded as follows for each casualty attribute (i.e., severity, age, race, sex):

$P(A|Y,H) = P(severity, age, race, sex| Y,H)= P(severity| age, race, sex, Y,H) P(age| race, sex, Y,H) P(race| sex, Y,H) P(sex| Y,H) = P(severity| Y,H) P(age| Y,H) P(race| Y,H) P(sex| Y,H)$

Thus the noncommensurate term is the product of the summation terms for each of the 4 attributes as shown.

The total scene hypothesis score is the product of the individual hypothesis scores for 5 types of report, S, to track, T, associations of kinematics, Y, & attributes, Z:

1. Association Hypotheses

$P(Y(S)|Y(T),H) P(Z(S), Z(T)|Y(S), Y(T), H) P(H) = \{|V|^{-1/2} \} exp[-1/2\{I^T V^1 I\}] \bullet \Pi_A\{\Sigma_K[P(K_A|Z(T),Y(T), H) P(K_A|Z(S),Y(S), H)/P(K_A|Y(T),Y(S), H)]\} \bullet [1-P_{FA}(S)] [1-P_{FA}(T)] P_D(S) P_D(T)$

2. Pop-up (i.e., Track Initiation) Hypotheses

$P(Y(S)|Y(T),H) P(Z(S), Z(T)|Y(S), Y(T), H) P(H) = \{E(|V|^{-1/2})\} exp[-1/2\{\mu\}] \bullet [1-P_{FA}(S)] [1- P_D(T)] P_D(S)$

3. False Alarm (FA) Hypotheses

$$P(Y(S)|Y(T),H) \, P(Z(S), Z(T)|Y(S), Y(T), H) \, P(H) = \{E(|V|^{-1/2})\} \, exp[-1/2\{\mu\}] \bullet P_{FA}(S) \, P_D(S) = 0$$

4. Propagation Hypotheses

$$P(Y(S)|Y(T),H) \, P(Z(S), Z(T)|Y(S), Y(T), H) \, P(H) = [1-P_{FA}(T)] \, [1- P_D(S)] \, P_D(T)$$

5. Track Drop Hypotheses

$$P(Y(S)|Y(T),H) \, P(Z(S), Z(T)|Y(S), Y(T), H) \, P(H) = P_{FA}(T) \, P_D(T) = 0$$

such that

1. $I = (x_T , y_T) - (x_S , y_S)$ which is the difference of the report and feasible CDB call for service locations

2. $V = P + R$ for each feasible association where

3. P is the 2x2 matrix for the feasible CDB with x and y variances along its diagonal and zero otherwise

4. R is the 2x2 matrix with the user selectable source x and y variances along its diagonal (baseline: 100 m^2 for police and ambulance reports and 400 m^2 for civilian reports

5. $|V|^{-1/2} = \{1 / [$square root of the determinant of V$]\}$ where the determinant of V is the product of the diagonal terms for diagonal matrices

6. $E(|V|^{-1/2})$ is computed using the V matrix from the highest scoring association

7. the normalization term in the pop-up hypothesis, $\mu = 1.39$ for 2 DOF that we have

8. The probability of false report (i.e., $P_{FA}(S)$) and false track (i.e., $P_{FA}(T)$) are zero for all sources due to the risk in dropping a call for service

9. The probability of reporting a casualty in the CDB (i.e., $P_D(S)$) for civilian, police, and ambulance (i.e., all) sources is a user specified parameter (baseline: = .8)

10. The probability of a gated call for service in the CDB (i.e., $P_D(T)$) being the reported casualty is also a user specified parameter (baseline: = .8)

Thus the association score for each feasible CDB call for service is #1 above. The non-association score is the product of equations #2 and #4.

A numerical example of an association score (i.e., equation #1 above) is given next. For a current police report at (100m, 100m) with the x and y standard deviations = 10m associating with a CDB call for service that was initiated last update time with a civilian report at (150m, 150m) with standard deviations in x and y of 20m, the innovations, I = (150, 150) – (100,100) = (50,50) which is a 2x1 matrix. The innovations covariance, V has 400 + 100 = 500 along the diagonal of the 2x2 matrix. The $|V|^{-1/2} = (500 * 500)^{-1/2} = 1/500 = .002$. $(I^T V^{-1})I = (50/500, 50/500) (50, 50)^T$ = (50/10) + (50/10) = 10. Thus, the probability of association term due to the location data match is as follows:

$P(Y(S: police\ location\ report\ )|Y(T: CDB\ track),H) = \{|V|^{-1/2}\}\ exp[-1/2\{I^T V^{-1} I\}] = .02\ exp(-1/2\ \{10\}) = .02 * .0067 = .00135$

Let P(C|S: police casualty severity report) = (.03, .08, .60, .29), let the only associated CDB call for service have the a posteriori severity class vector is (.1, .2, .6, .1), and let the a priori casualty severity class vector = [.3, .3, .2, .2], then the probability of association term due to the casualty data match is as follows:

$P(casualty\ severity\ match) = \Sigma_K[P(K_A|Z(T),Y(T), H)\ P(K_A|Z(S),Y(S), H)/P(K_A|Y(T),Y(S), H)] =$

$\{[.03\ x\ .1]/.3\} + \{[.08\ x.2]/.3\} + \{[.6\ x.6]/.2\} + \{[.29\ x.1]\}/.2\} = .01+.05+1.80+.15 = 2.01$

Let P(C|S: police race report) = (.03, .08, .60, .29), let the only associated CDB call for service have the a posteriori race class vector is (.1, .2, .6, .1), and let the a priori race class vector = [.3, .3, .2, .2], then the probability of association term due to the casualty data match is as follows:

P(race match) = Σ_K[P(K_A|Z(T),Y(T), H) P(K_A|Z(S),Y(S), H)/P(K_A|Y(T),Y(S), H)] =

{[.03 x .1]/.3} + {[.08 x.2]/.3} + {[.6 x.6]/.2} + {[.29 x.1]}/.2} = .01+.05+1.80+.15
= 2.01

Let P(C|S: police age report) = (0, .03, .08, .60, .29), let the only associated CDB call for service have the a posteriori race class vector is (0, .1, .2, .6, .1), and let the a priori race class vector = [.01, .29, .3, .2, .2], then the probability of association term due to the casualty data match is as follows:

P(age match) = Σ_K[P(K_A|Z(T),Y(T), H) P(K_A|Z(S),Y(S), H)/P(K_A|Y(T),Y(S), H)] =

{[.0 x .0]/.01} + {[.03 x .1]/.29} + {[.08 x.2]/.3} + {[.6 x.6]/.2} + {[.29 x.1]}/.2} =
.01+.05+1.80+.15 = 2.01

Let P(C|S: police sex report) = (.9, .1), let the only associated CDB call for service have the a posteriori race class vector is (.9, .1), and let the a priori race class vector = [.5, .5], then the probability of association term due to the casualty data match is as follows:

P(sex match) = Σ_K[P(K_A|Z(T),Y(T), H) P(K_A|Z(S),Y(S), H)/P(K_A|Y(T),Y(S), H)] =

{[.9 x.9]/.5} + {[.9 x.9]}/.5} = 1.62 + 1.62 = 3.24

The association score for all 4 casualty attributes is the product of these as follows:

Π_A{Σ_K[P(K_A|Z(T),Y(T), H) P(K_A|Z(S),Y(S), H)/P(K_A|Y(T),Y(S), H)]} =
*2.01*2.01*2.01*3.24 = 26.3*

The overall association score for the police report and CDB call for service is as follows:

P(Y(S)|Y(T),H) P(Z(S), Z(T)|Y(S), Y(T), H) P(H) = {|V|^{-1/2} } exp[-1/2{I^T V⁻¹ I }] •
Π_A{Σ_K[P(K_A|Z(T),Y(T), H) P(K_A|Z(S),Y(S), H)/P(K_A|Y(T),Y(S), H)]} • [1-P_{FA}(S)] [1-
*P_{FA}(T)] P_D(S) P_D(T) = .00135 * 26.3 *.8 *.5 = .014*

Since in this example, this is the only feasibly associated CDB call for service it is compared to the nonassociation hypothesis. The score is the product of equations 2 and 4 above. The result is as follows:

$$P(Y(S)|Y(T),H) \, P(Z(S), Z(T)|Y(S), Y(T), H) \, P(H) * P(Y(S)|Y(T),H) \, P(Z(S), Z(T)|Y(S), Y(T), H) \, P(H) = \{E(|V|^{-1/2})\} \, exp[-1/2\{\mu\}] \bullet [1-P_{FA}(S)] \, [1- P_D(T)] \, P_D(S) * [1-P_{FA}(T)] \, [1- P_D(S)] \, P_D(T) = .02 \, exp[-1/2\{1.39\}] * .5 * .8 * .2 * .5 = .02 * .5 * .04 = .004$$

For this example, the association score =.014 > .004 = the nonassociation score, thus the association will be selected in the hypothesis selection function described next.

Hypothesis Selection

In the hypothesis selection the CDB call for service with the highest association score is compared to the non-association score (i.e., the product of #2 and #4). The highest score between the 2 is selected.

Casualty State Estimation

If the non-association is selected a new entry (i.e., call for service) is initiated in the CDB with the reported information. Otherwise the report is used to update the highest scoring association call for service. The following elements of the selected CDB call for service will be updated during state estimation:

1. last associated CDB report type (if =27 has been picked-up, if 25 dropped-off, if 61 walked-in)

2. jurisdiction ID

3. casualty ID (CASID) (=0 if unknown)

4. last update time in seconds

5. casualty location (x,y) = (east, north) in meters

6. casualty location covariance (i.e., P)

7. casualty severity vector

8. casualty age vector

9. casualty race vector

10. casualty sex vector

11. number of times this call for service has been updated

12. cumulative correct association probability (for track initiation it is 1, then for each subsequent update multiply by the current association probability which is approximated as the association score divided by sum of all the feasible association scores including the non-association score.

The location will be updated with a standard Kalman filter assuming no casualty movement until a drop-off or walk-in report is received. When either of these 2 types are associated, then a replacement will be used for all state updates. The Kalman filter update here is applied as follows:

$$(x_T, y_T) \text{ (updated)} = (x_T, y_T) + K [(x_S, y_S) - (x_T, y_T)]$$

$$\text{where } K = P [P+R]^{-1}$$

$$P(\text{updated}) = [I - K] P$$

Ambulance pick-up and drop-off reports will contain a CASID. This is also true of walk-in reports. These states in the CDB will be 0 until these reports are received for each casualty. The ambulance pick-up report will be used in a Kalman update the casualty location (e.g., since its position has not changed yet), whereas the drop-off and walk-in locations will be updated with replacement.

The attribute update for these 3 report types will be by replacement. The current report time, type, and reporter ID will be added to the updated CDB call for service or to the file it points to. The attribute update for all other associated casualty reports (e.g., injury severity, age, race, sex) will be Bayesian. The attribute confidence update equation for each of the 4 attributes uses the attribute confidence vectors attached in data preparation above as follows:

P(class C| S, T, Y, H) = [P(C|S, Y, H) P(C|T, Y, H)/P(C|Y, H)]/ Σ_K [P(K|S, Y, H)
P(K|T, Y, H)/P(K|Y, H)]   if P(C|H)≠0 [= 0   if P(C|H)=0]

where

1. C is the element of the a posteriori severity class vector being updated,

2. P(C|S, Y, H) is the element of the a posteriori severity class tree from the source report

3. P(C|T, Y, H) is the element of the a posteriori severity class tree from the associated data base track entity

4. P(C|Y, H) is the a priori probability of a casualty of type C given only entity location & H, the association hypothesis. This is a user input. If the a priori has a uniform distribution, then we can ignore this term.

5. K is the index over the disjoint classes for each attribute (i.e., severity, age, race, sex) [summed over for normalization],

These confidences will improve with additional confirming reports and reduce otherwise. If a conflict is obtained in the attribute update (i.e., all confidences are 0), then flag and use the next associated report to initiate a new attribute confidence vector.

Below is an example for a given a severity class ontology with 4 elements [severity 1, severity 2, severity 3, severity 4]. Given a police casualty severity report = 3 then the a posteriori severity class vector as derived above is P(C|S)= (.03, .08, .60, .29). If the associated CTP casualty a posteriori severity class vector is (.1, .2, .6, .1) and the a priori casualty severity class vector = [.3, .3, .2, .2], then the fused casualty severity class vector update becomes the following:

P(severity 1) = {[.03 x .1]/.3}/ {.01+.05+1.80+.15} =.01/2.01 = .005

P(severity 2) = {[.08 x.2]/.3}/ {.01+.05+1.80+.15} =.05/2.01 = .025

P(severity 3) = {[.60 x.6]/.2}/ {.01+.05+1.80+.15} =1.8/2.01 = .895

P(severity 4)= {[.29 x.1]}/.2}/ {.01+.05+1.80+.15} =.15/2.01 = .075

6.8.2.2 Emergency Vehicle Fusion Segment Node Processing

This fusion network processes police and ambulance vehicle location reports one at a time as they arrive in a sequential network of fusion nodes. Each node does no data preparation since no

propagation and no attribute vector insertion is needed. Data association is by unique police or ambulance ID. The emergency vehicle or police location update is by replacement since the 5 minute update rate is not sufficient to track the police vehicles using velocity. The Emergency Vehicle Data Base (EVDB) entries to be updated during state estimation are as follows:

1. EVDB report type

2. Police Vehicle ID

3. Jurisdiction ID

4. Last updated time in seconds

5. Last updated location (x,y) = (east, north) in meters

6. ambulance idle attribute set when that report type is received (i.e., 0 or 1)

7. ambulance stuck set when that report type is received (i.e., 0 or 1)

6.8.2.3 Facility Fusion Segment Node Processing

The facility fusion node network processes police hospital/emergency facility and hospital status reports one at a time as they arrive in a sequence of nodes each treating a single observer report. Data preparation converts the facility damage declaration to a severity confidence vector using a table look-up of the vectors computed prior to running the current scenario as described at the end of Section 2.   Each report contains a unique hospital/emergency facility ID that enables unique association with the facility data base (FDB). The elements of each entry (i.e., track) in the FDB to be updated during state estimation are as follows:

1. FDB report type

2. Facility ID

3. Jurisdiction ID

4. Last updated time in seconds

5. facility location (x,y) = (east, north) in meters

6. facility damage severity vector

7. associated damage report types, reporter IDs, and update report times since beginning of scenario (can be a pointer to a file)

8. Number of total beds (when hospital status report received)

9. Number of beds occupied (when hospital status report received)

10. Percentage of operation level (when hospital status report received)

The hospital status attributes are updated with replacement as received. The severity attribute confidence update equation using the confusion matrix and a priori data is the same as above.

$$P(class\ C|\ S,\ T,\ Y,\ H) =\ [P(C|S,\ Y,\ H)\ P(C|T,\ Y,\ H)/P(C|Y,\ H)]/\ \Sigma_K\ [P(K|S,\ Y,\ H)$$
$$P(K|T,\ Y,\ H)/P(K|Y,\ H)]\quad if\ P(C|H){\neq}0\ [=\ 0\quad if\ P(C|H)=0]$$

1. C is the element of the a posteriori severity class vector being updated,

2. $P(C|S, Y, H)$ is the element of the a posteriori severity class tree from the source report

3. $P(C|T, Y, H)$ is the element of the a posteriori severity class tree from the associated data base track entity

4. $P(C|Y, H)$ is the a priori probability of an entity of type C given only entity location & H, the association hypothesis. This is a user input. If the a priori has a uniform distribution, then we can ignore this term.

5. K is the index over the severity disjoint classes [summed over for normalization],

If a conflict is obtained in the attribute update (i.e., all confidences are 0), then flag and use the next associated report to initiate a new attribute confidence vector.

6.8.2.4 Transportation Link Delay Fusion Node Processing

The transportation fusion node network is a sequence of nodes each treating a single observer set of reports at a single time, just as the casualty and other 3 fusion node segment networks. The road link delays are reported by police or ambulances (e.g., not civilians, nor hospitals). Since link delays can change unexpectedly, the fusion node will associate using the unique link ID and replace the delay information. The transportation link data base (TLDB) states are as follows:

1. TLDB report type

2. Transportation Link ID

3. Jurisdiction ID

4. Last updated time in seconds

5. link delay severity factor between [0,1] (i.e., this factor times vehicle speed is link speed so 1 is no delay and .5 is half speed)

6.8.2.5 Bridge Fusion Node Processing

The bridge fusion node network is a sequence of nodes each treating a single observer set of reports at a single time, just as the casualty and other 3 fusion node segment networks. The bridge damage is reported by police, ambulances, or civilians (type 38). Data preparation converts the facility damage declaration to a severity confidence vector using a table look-up of the vectors computed prior to running the current scenario as described at the end of Section 2. The fusion node will associate using the unique bridge ID and update the bridge damage information. The bridge data base (BDB) states are as follows:

1. BDB report type

2. Transportation Link ID

3. Jurisdiction ID

4. Last updated time in seconds

5. bridge location (x,y) = (east, north) in meters

6. bridge damage severity vector

The bridge damage severity attribute confidence update equation using the confusion matrix and a priori data is the same as above. Namely,

$P(class\ C|\ S,\ T,\ Y,\ H) =\ [P(C|S,\ Y,\ H)\ P(C|T,\ Y,\ H)/P(C|Y,\ H)]/\ \Sigma_K\ [P(K|S,\ Y,\ H)$
$P(K|T,\ Y,\ H)/P(K|Y,\ H)]\quad if\ P(C|H)\neq0\ \ [=0\quad if\ P(C|H)=0]$

1. C is the element of the a posteriori severity class vector being updated,

2. P(C|S, Y, H) is the element of the a posteriori severity class tree from the source report

3. P(C|T, Y, H) is the element of the a posteriori severity class tree from the associated data base track entity

4. P(C|Y, H) is the a priori probability of an entity of type C given only entity location & H, the association hypothesis. This is a user input. If the a priori has a uniform distribution, then we can ignore this term.

5. K is the index over the severity disjoint classes [summed over for normalization],

If a conflict is obtained in the attribute update (i.e., all confidences are 0), then flag and use the next associated report to initiate a new bridge damage attribute confidence vector.

6.8.3 Confusion Matrices

The approximation to the a posteriori severity class vectors from the confusion matrix will be made as follows:

$P(C|S)= P(D|C)\ P(C)/\ \Sigma_K\ \{\ P(D_K\ |C)\ P(C)\ \}$

for each class C where P(D|C) are the elements in the confusion matrix column for the given declaration. The resulting class confidence vector for each scenario run will then be perturbed. The perturbation will be a uniform distribution over ±10% of each of the values divided by the sum of these perturbed values. The a priori vector, P(C), will also be perturbed in the same way.

AFOSR F49620-01-1-0371

For example, with the police casualty severity confusion matrix below and a police report of severity 3, the a posteriori class severity class vector for the a priori casualty severity class vector = [.3, .3, .2, .2] is as follows:

$$P(C=1|S) = P(D=3|C=1)/ \Sigma_K\{ P(D =K |C=1) P(C=1)\} = .03 * .3/ [.03*.3 + .07*.3 + .8*.2 + .38*.2] = .009/[.009+.021+.16+.076] = .009/.266 = .034$$

So, P(C|S) = (.034, .079, .601, .286). Since these values are very rough to begin with, this rough approximation should be sufficient. Note for comparison that if the a priori was uniform P(C|S) = (.02, .06, .62, .3). All confusion matrices will be processed for the given a priori class vector to generate these a posteriori severity class vectors and perturbations will be added for each before each scenario is run.

6.9 Higher Level Fusion

As stated in Section 2 Objectives, a principal focus of this project was the elucidation of a framework for application of higher level fusion processing in the emergency response phase of a man-made or natural disaster. In this section the relationship of L2/L3 fusion to this circumstance is discussed, and the processing of fusion inputs consistent with this relationship described. Attention is given to both the general problem, and the application of the resulting design principles in the DIRE test bed.

It is widely agreed that the great majority of successful data fusion methods to date have focused on low level (Level 0 and Level 1) fusion related to processing information about a single object of interest (see, e.g. [6.9-1]). While effective fusion at the attribute and object levels producing object identification and characterization offers real performance gains in many applications, it does not provide for user situation awareness essential for effective decision making [6.9-1]. Situation awareness requires contextual understanding and interpretation of the events and behaviors of interest, which can be achieved by utilizing higher level fusion processes (situation assessment and impact prediction).

The purpose of higher level fusion (HLF) processes is to infer and approximate the critical characteristics of the environment in relation to particular goals, capabilities and policies of the decision makers. HLF utilizes fused data about objects of interest and relationships between

them, their behavior, dynamic databases, expert experience, knowledge, and opinion for context processing. The result of HLF is a coherent composite picture of the current situation along with prediction of consequences, which provides decision makers with essential information to help them to understand and control the situation and act effectively to mitigate its impact. Disaster response invariably engages several distinct organizations each with different tasking, competencies, technologies and scope of operation. Situation and impact assessment (SIA) has to deliver consistent current and predicted situational pictures, which are relevant to each decision maker's goals and functions.

The main goals of the crisis management in post-disaster environment are to save lives, to control the situation, and to minimize the effects of the disaster. "Multiple distributed decision makers are searching for the answers to the following questions: where the problems are, what kind of problems they are, and what the impact of this problem is" [6.9-2]. HLF is essential for answering these questions since identification, recognition, and attribution of individual objects are not sufficient for an effective coordinated disaster response. There is a need to convert the fused data about individual objects such as damage of individual buildings, roads, bridges, facilities, fires, and casualties into usable knowledge about current and predicted disaster scene [6.9-3, 6.9-4].

The process of building a situational picture comprises dynamic generation of hypotheses about the states of the environment and assessment of their plausibility via reasoning about situational items, their aggregates at different levels of granularity, relationships between them, and their behavior within a specific context. In some cases, assessment of plausibility of more complex hypotheses may require hierarchical processing, which includes not only reasoning about situational items and relationships between them but also includes relationships between hypotheses and assessments of plausibility of lower level hypotheses [6.9-6]. An important component of situation assessment is causal inference aimed at discovery of underlying causes of observed situational items, their attributes and their behavior. Discovery of underlying causes of observed situations is the goal of abductive reasoning [6.9-6, 6.9-7] or "inference for best explanations". For example, in the early post-earthquake response phase, reasoning about situations is contingent on the assumption that most reported casualties and structural damage are the results of the primary earthquake shock incident and reported subsequent secondary incidents

such as fire, flood, aftershocks and Hazmat events. However some secondary incidents such as toxic spills may not be known for a long period of time. At the same time rapid discovery of such incidents is very important since they may have devastating consequences if not responded to quickly.

These unknown secondary incidents are usually manifested by unexpected properties and behavior of situational items inconsistent with the current set of beliefs about the state of the world and therefore belief update may be required. Usually belief update methods give priority to this new information and its consequences and abandon some old beliefs to preserve consequences. In the post disaster environment observations and knowledge about situational items, their behavior and relationships are uncertain and, therefore it is necessary to account for this uncertainty while updating the current set of beliefs. In the uncertain environment the principle of priority of new information may not work even in a highly dynamic environment. In the uncertain dynamic environment belief update can be carried out by first seeking some explanations or underlying causes of these inconsistent observations and incorporating these explanations, if found, into a new set of beliefs. Possible explanations can be found as the result of abduction comprising generation of hypotheses about the underlying causes of these inconsistent observations and reasoning about plausibility of such hypotheses. This report presents a general approach to designing higher level fusion processing as applied to dynamic post-disaster environment.

6.9.1 General approach

The post-disaster environment has specific characteristics, which define requirements for HLF architecture and processes. These characteristics comprise:

1. Noisy and uncertain dynamic environment with insufficient *a priori* statistical information.

2. Geographically distributed damage, casualties, and resources of first responders.

3. Geographically distributed uncertain sources of information often of varying significance, low fidelity, contradictory and redundant.

4. Large amount of heterogeneous information.

5. High probability of secondary incidents such as after shocks and tsunami in the case of earthquakes, hazmat events, flood, fire, etc.

6. Resource and time constraints.

7. High cost of error.

8. Multiple decision makers with different goals, functions, and information requirements. Some of them have tactical missions calling for decisions on direct response to a situation while the others have strategic missions calling for higher level estimation of the situation, impact prediction, and analysis.

9. Multiple agencies in multiple jurisdictions.

These specific domain characteristics call for a multi-agent distributed dynamic HLF processes, which have to be scalable, adaptive to resource and time constraints, new and uncertain environments, and have to be reactive to uncertain inputs. These processes also have to accommodate heterogeneous information (both symbolic and numeric), allow for complex distributed system modeling, efficient information sharing, and incorporating qualitative experts' opinions. It is necessary to note that the post-disaster environment characteristics and HLF processing requirements mentioned above are very common for various applications dealing with unintended threat, which makes an approach to building such processing quite generic.

In the disaster environment, the HLF processes exploit continually associated and fused information on single entities such as casualties, road, building, and facility damage obtained from multiple observer reports, domain knowledge, and the results of domain-specific simulations and models to produce a consistent estimate of the current and predicted state of the environment, which is presented to users. Figure 6.9-1 shows a notional architecture of the HLF processing.

There are several essential components of the fusion processing required for building current and predicted situational pictures:

1. Formally structured and computationally tractable domain representation capturing the basic structures of relevant objective reality and users' domain knowledge and requirements, which further serves as a basis for reasoning about the states of the environment.

2. Dynamic reasoning procedures about objects, attributes, aggregates, relationships and their behavior over time within a specific context.

3. Domain specific simulations and models such as earthquake consequences model (e.g. HAZUS), dispatch and dynamic routing model, plume model, hospital model, model of usual behavior, etc

4. Inter- and intra-fusion level decision state estimation (quality control models).

5. Belief update under uncertainty

6. The remainder of Section 6.9 comprises a description of the processes presented in Figure 6.9-1 in greater detail.
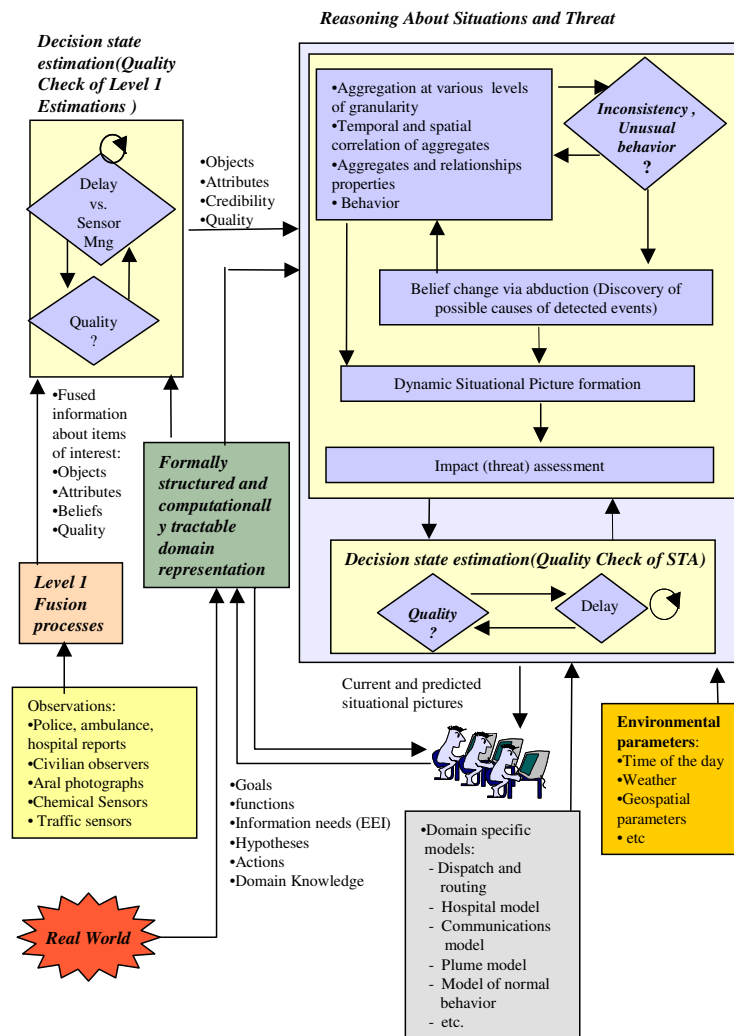
Figure 6.9-1. Notional architecture of the HLF processing.

6.9.3 Domain representation

One of the major challenges of designing the HLF processes is a problem of providing a consistent, comprehensive, and computationally tractable representation of phenomenology of the domain under consideration. A combination of Cognitive Work Analysis (CWA) and formal ontological analysis of a specific domain is designed to overcome this problem and provide sufficient information about decision maker's goals, functions, information needs, types of

AFOSR F49620-01-1-0371

objects, relations between them, and processes to support the domain-specific generation of situational hypotheses and high-level reasoning about these situational hypotheses [6.9-8]. This approach permits decomposition of the complex highly interacting scenario into a set of basic and derived situations as will be next considered.

Users' goals, functions, and information needs are identified by the means of CWA [6.9-9], which is a systems-based approach to the analysis, design and evaluation of an emergency management environment in a post-disaster context. It provides understanding of what content various decision makers require from a situational picture and what information should be represented, possible hypotheses about relevant states of the environment. The work domain model is derived from a variety of information sources, including documents detailing emergency management plans; data collected during and following similar historic disasters (two major earthquakes in California); interviews with emergency management personnel at city, county and state jurisdictional levels, interviews with personnel from the Federal Emergency Management Agency; and observations of a full scale emergency management exercise at the city level [6.9-8]. More details about CWA in the project are given in Section Y

The result of CWA provides answers to the following questions

    A. What are the decision makers expecting from a situational picture?

    B. What information is required for making decisions?

    C. What active alternative hypotheses about the environment can be expected?

Examples of these essential elements of information identified include:

1. Regions of causalities (e.g., location, boundaries, severity of injury).

2. Risks of secondary hazards (e.g., hazardous materials spills, fires).

3. Areas of impeded transportation (e.g., location, boundaries).

4. Resource balance assessments (e.g., available vs. potentially required medical personnel, building inspection teams, search and rescue equipment)

5. Status of critical facilities (e.g. hospitals, bridges, shelters)

6. Status of communications systems (e.g. regions of impeded wire or cell service)

Emergency management in the post-disaster environment has hierarchical organization, in which decisions at certain levels of hierarchy have a tactical character (e.g. activities in direct response to casualties reported at a specific location), while decisions at higher levels have the strategic character of understanding situations related to a larger region or over a longer period of time. This hierarchical structure of emergency management and likely existence of several regional jurisdictions within the disaster area define a hierarchical structure of essential elements of information. For example, tactical decision makers may be interested in location of regions of casualties within a small area, ambulances available within a short time interval, and nearby hospitals with adequate residual capacity. Strategic decision makers may want to know the distribution of casualties within a much larger region and the balance of medical resources over the whole initial response period.

The role of a formally structured domain-specific ontology of the environment under consideration is to provide a comprehensively large, and metaphysically accurate model of situations, through which specific tasks such as situation assessment, knowledge discovery, or the like, can be more effectively performed, since the information necessary for these decision-making aids is contained within the ontology's structure. [6.9-10]. The formal ontology framework is necessary to provide a formal structure for ontological analysis of the specific environment and to assure a certain level of reusability of the designed domain-specific ontology in a different application domain.

Formal ontology of situations comprises two types of items: spatial (situational items) and temporal (processes), together with the relations between and among them. Spatial items, elements of the embedded *snap ontology*, and relations between them are defined by a set of spatial and mereological attributes. The values of these attributes define the state of these items and a corresponding state of the environment. Temporal items, i.e. processes, are elements of the related *span ontology,* which describe the temporal behavior of the situational items and dynamics of attributes and relations. Important characteristic of processes are events representing transition between states of the environment defining situations. They are manifested by

significant (as measured by a selected threshold) changes in attributes and behavior of physical and abstract situational objects [6.9-11]. Events related to a particular situational item could trigger events related to other situational items. Events are always context dependent since "significance" is always context specific. Event discovery is a very important element of the HLF process, which can lead to knowledge discovery about underlying causes of events and the states of the environment.

Each relationship characterizing a situation falls into one of two basic categories: *inter-class relations* and *intra-class relations.* Intra-relations (i.e., internal relations) are spatial, temporal, or functional relations that exist within a given set of ontologically similar items while *inter-class relations* (e.g., external relations) exist *between* various items. A more detailed description of a formal ontology of catastrophic events is presented in Section Y.

The hierarchical structure of the essential elements of information dictates categorization of situations at various levels of granularity. Consideration of situations at different level of granularity also helps to reduce complexity of the reasoning process.

Each relationship characterizing a situation is context specific and falls into one of two categories: inter-class or intra-class. Intra-relations (i.e., internal relations) are spatial, temporal, or functional relations that exist between:

1. physical objects of the same types

2. aggregates of similar situational items at different levels of granularity

3. events of the type

4. aggregates of similar events

5. similar processes

6. aggregates of similar processes

7. events and processes characterizing similar situational object

8. Inter-class relations (i.e., external relations) are spatial, temporal, or functional relations that exist between the following:

9. objects of different types

10. objects and aggregates of different types

11. aggregates of situational items of different types at the same level of granularity

12. individuated aggregates of different types at different levels of granularity

13. events of different types

14. aggregates of events of different types

15. processes of different types

The basic situations (the building blocks of situations) are described by context dependent relationships between physical items of the same category such as casualties, buildings, and ambulances, or similar events such as discovery of casualties of a certain type of injury at a certain time. These basic situations are defined as aggregates (clusters) and are obtained by applying a similarity metric in the feature space. The type of features used for aggregation depends on the information needs of a certain user or a group of users. Context dependent relationships between aggregates at a certain level of granularity define derived situations at the next level of granularity. Events related to aggregates are represented by significant change of the parameters of aggregates, discovery of a new aggregate, or split/merging of aggregates at a higher level of granularity.

Derived intra-class situations created by the composition of basic intra-class situations at specific levels of granularity is called *elementary situations*. Relationships between elementary situations within a selected spatio-temporal setting and overall context comprise a *composite situational picture.*

Relationships between items at various levels of granularity are represented by mereological primitives [9], direction, size, and distance (Table 6.9-1). Relations between events and processes (span relationships) are defined by time point and time interval relationships (Table

6.9-2). Examples of such relations important for reasoning bout current and predicted situations in the post disaster environment are presented in Tables 6.9-1 and 6.9-2.

Table 6.9-1. Temporal relationships

Relation between time points	Before, At the same time, Start, Finish, Soon, Very soon, Resulting in, Initiating, value of time interval
Relation between time intervals	Disjoint, Joint, Overlap, Inside, Equal

Table 6.9-2. Spatial relationships

Topology/mereology	Direction	Size	Distance
Disjoint	Along	Smaller	Not far
Joint	Towards	Larger	Far
Overlap	East	Size difference	Very far
Cover	West		Close
Reachable	South		Very close
Unreachable	North		Distance between clusters
Contain	Similar		Distance between centroids
A part of	Opposite		

Specific contextual examples are:

Close to a hospital,

Cluster A is *larger* than before,

1. Cluster B is *along* the west wind direction

2. Distance between Clusters *A* and *B is smaller than before,*

3. Casualty cluster A *overlaps* with building cluster C.

The value assigned to each relation depends on a specific context and a specific user, for which a situation defined by this relation is considered. For example, for an ambulance dispatcher certain

Figure 6.9-2   Structured domain representation

hospital can be considered far away from a particular cluster due to heavy congestion on the roads surrounding the cluster. The same distance can be small for a helicopter.
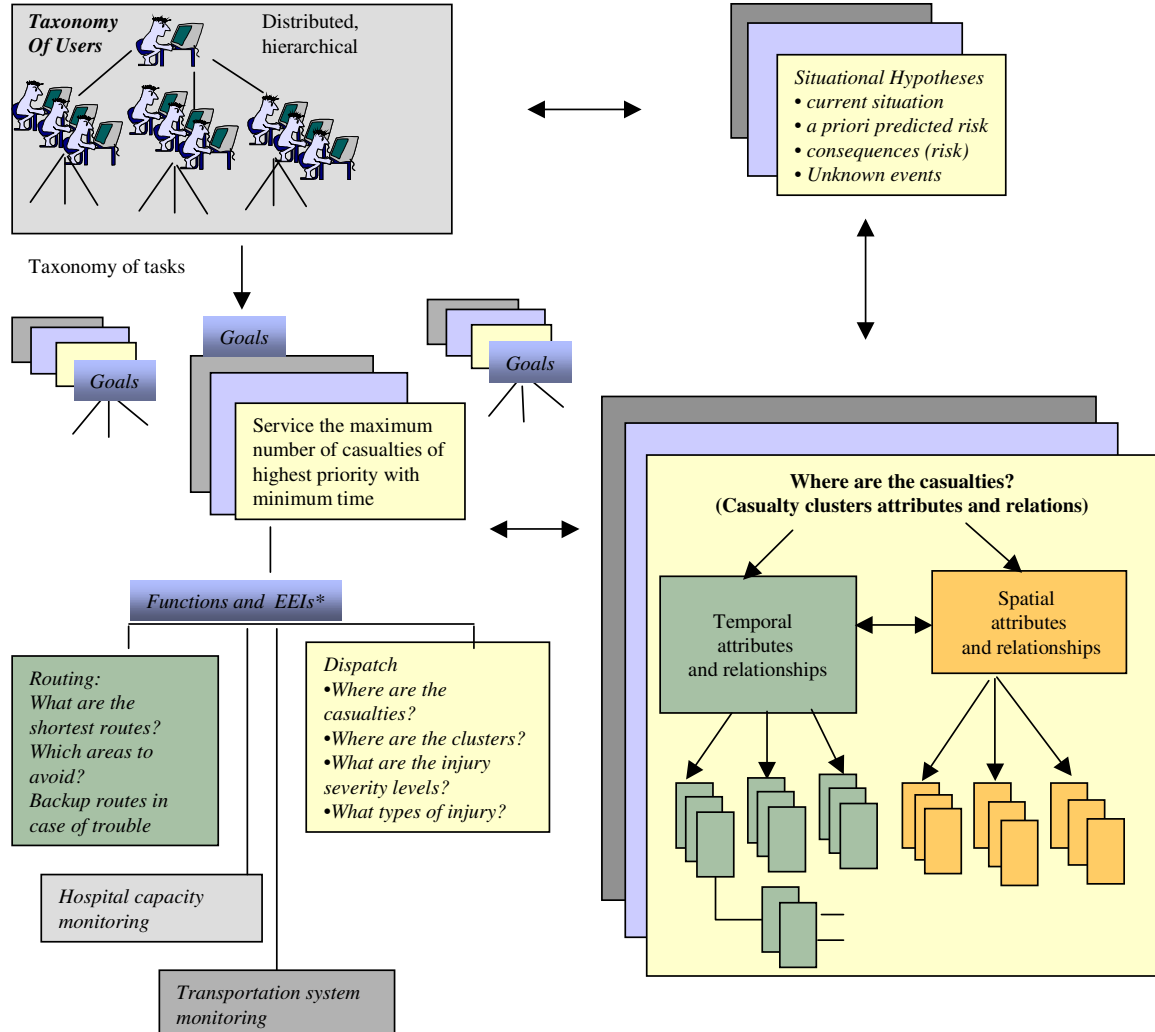
It is necessary to note that all relationships mentioned above are uncertain and vague and can be both numeric and symbolic and, therefore, reasoning about these relationships has to deal with uncertainty and accommodate both types of information. Figure 6.9-2 presents structured domain representation (the result of combination of CWA and ontology).

One of the most important types of situations is an *elementary situation* defined by intra-class relationships between certain situational items of various levels of granularity. Thus, for example, casualty situations may be defined by temporal or spatial relationship between casualty clusters and casualties of certain characteristics or between two casualty clusters.

The most important elementary situations to be considered are

1. Communication system situation

2. Transportation system situation

3. Hazmat situation.

AFOSR F49620-01-1-0371

4. Casualty situation.

5. Hospital situation.

6. Building situation.

7. Ambulance situation.

While there are many essential elements of information, which can be obtained from basic situations there are even more important essential elements of information, which can be obtained only by considering interclass relationships between physical items and aggregates and between different classes of aggregates at various level of granularity. Such interclass situations are called derived situations. One of the most important types of interclass relationships is represented by event relationships "resulting in". Value of this relationship may be qualitative such as increase/decrease, increase/decries with a certain confidence as well as quantitative such as increase decrease by a certain value, increase/decries with a certain confidence. Establishing such relationships on a qualitative and if possible quantitative basis gives a foundation for reasoning about causes and consequences of such events. Figure 6.9-3 shows an example of
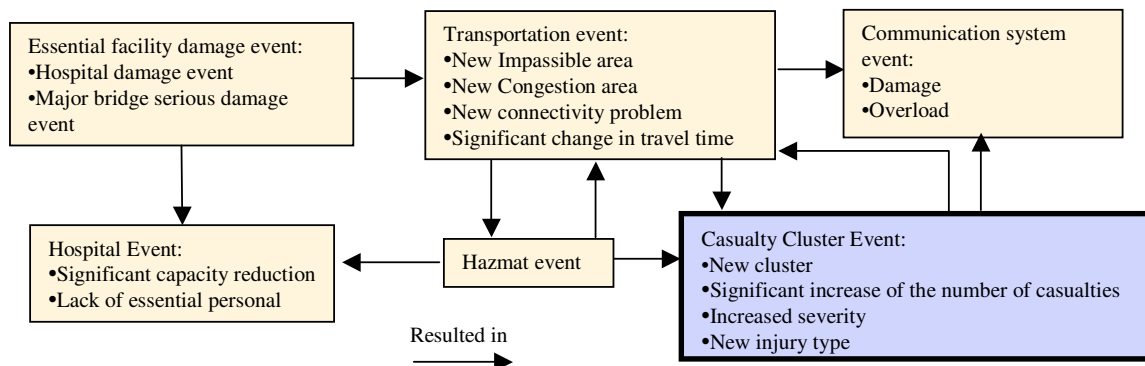
"Resulting in" relationship between events affecting casualty situation.

Figure 6.9-3. "Resulting in" relationship between events affecting casualty situation

The processes of assessing current and future situations are presented in Section 6.9-.5?

AFOSR F49620-01-1-0371

6.9.3 Quality control procedure

The success of dynamic SIA greatly depends on the quality (e.g. uncertainty, vagueness, reliability, and relevance) of individual and integrated in level 1 fusion data as well as information resulting from all interim steps of higher level fusion processing. The information quality considerations play an important role in transferring information within fusion levels as well as between levels [6.9-12]. The strategies for quality control within as well as between fusion levels can include eliminating information of low quality from consideration, incorporating information quality into fusion processing, utilizing process refinement by sensor management, and/or delaying the transferring the results to the next processing level or to decision makers until information of better quality can be obtained as the result of more observations and/or additional computations (anytime processing).

Quality control is highly context specific since the notion of "good", "poor", or "satisfactory" quality greatly depends on context. Incorporation of information quality into SIA processing is a difficult task since it is generally not clear how to measure the quality of the result of many processes and how different dimensions defining information quality are interrelated. Usually the quality criteria and quality factors to be consider depend of the context and in many cases may be defined by users.

Utilization of anytime processing has to take into account the fact that responders in the early post-disaster environment are under severe time and resource constraints, and timely decision making and swift action are required. At the same time the cost of false alarms can be very high since valuable resources might be diverted from the location where it later becomes clear that they are critically needed. The cost of waiting for additional information, or cost of additional computation delay for obtaining results of better quality, has to be justified by the benefits of obtaining better results. This can be achieved by either implicitly modeling expected utility of making decision at a certain moment by accepting the information current quality or by comparing the quality of information achieved at a certain time with a time varying threshold [6.9-13]. The state, in which information to be transferred to the next processing level or to the decision makers is known as a "decision state".

In the current system the level 1 results as well as the results of any interim SIA steps, e.g. of the process of dynamic aggregation, are allowed to be used by other processes or passed on to decision makers if they are of a minimum threshold quality. These processes requires quality criteria and the function defining the time varying threshold, which in some cases can be can be obtained as the result of expert knowledge elicitation. The specifics of the quality control procedures for the level 1 fusion results (preprocessing) will be described in Subsection 6.9-.4.1 while the quality control methods for current and predicted state estimation will be included in the description of the SIA processes. In certain situations, when decisions based on the resulting decision state estimations have very serious consequences, a sensor management process can be employed. For example, a highly reliable sensor, perhaps a policeman or structural engineer, can be tasked to observe the situation in question.

6.9.3.1 Preprocessing

The goal of preprocessing is to define when reports fused by lower level fusion processes are ready to be used by SIA. The quality test for fused reports on casualties or building/essential facility damage is based on:

1. the compound reliability of the associated and fused reports about track ID i at time t ($R_i(t)$).

2. location uncertainty ($\max(\sigma_x^i(t), \sigma_y^i(t))$) provided by the Level 1 fusion module, where $\sigma_x^i(t), \sigma_y^i(t)$ are the x and y location standard deviations

3. Time-varying thresholds for false alarm and location uncertainty ($Th_R(t), Th_\sigma(t)$), monotone decreasing functions of time which guarantee that each casualty will be accepted by SIA before a certain deadline .

The compound reliability R^t is computed as a function of reliability of all reports fused for particular track ID and is computed within the formalism of the Dempster-Shafer theory of evidence [6.9-14].

Let $\Theta = \{\theta_1, \theta_2\}$ is a frame of discernment, where θ_1 is the hypothesis that the fused report is reliable and θ_2 that it is not. Let $r_i^n(t)$ be reliability of reporter n at time t and $N_i(t)$ is the number of reports fused by and including time t to obtain characteristics of track ID i. The report is true if a report source is reliable and it can be either true or false if the report source is unreliable. Then $r_i^n(t)$ is a measure of support for hypothesis θ_1 and yields a basic probability assignment:

$$m_n(\theta_1) = r_i^n(t), m_n(\Theta) = 1 - r_i^n(t) \ \forall n = 1, ..., N_i(t).$$

6.9.3.2 Decision rule

The result of combination of these basic probability assignments represents the total reliability of the characteristics of the track ID i:

$$R_i(t) = 1 - \prod_{n=1}^{N(t)} (1 - r_i^n(t)),$$

The reliability of reported of various classes (police offices, ambulance drivers, and civilians) is provided by domain knowledge, e.g. statistics obtained by the 911 centers. The resulting decision rule is: *use track ID for SIA is the proposition for which the following is true.*

$$(R_i(t) \geq Th_R(t)) \wedge (\max(\sigma_x^i(t), \sigma_y^i(t)) \leq Th_\sigma(t))$$

6.9.4   Situation and impact assessment processing

In this section the data transformations producing the higher-level fusion products are described. Brief justification of the reasoning strategies are provided.

6.9.4.1 Reasoning about situations

Let Ω be a set of possible states of the environment, $\Omega^k \subset \Omega$ be a subset of possible states of the environment relevant to decision maker k, and Ω be a plausibility structure on Ω. At each time t, a situational picture relevant to decision maker k can be described as a set of the plausible states of environment: $S^k(t) = \{\omega_i^k(t) \in \Omega^k \mid Pl(\omega_i^k(t)) > 0)$ [6.9-15]. Thus situation and threat assessment can be defined as a process of identifying and predicting a subset of plausible states

of the environment along with plausibility assigned to each state. It is assumed here that decision makers do not have complete knowledge about all relevant states of the environment and do not exclude the existence of an unknown hypothesis (the open world assumption).

The process of building a situational picture comprises dynamic generation of hypotheses about current and predicted states of the environment and assessment of their plausibility via reasoning about situational items at different levels of granularity and relationships between them within a specific context. These hypotheses include hypotheses about characteristics and behavior of situational items as well as hypotheses about the states of the environment, which explain these characteristics and behavior.

As it was mentioned in the introduction assessment of plausibility of more complex hypotheses may require hierarchical processing, which includes not only reasoning about situational items and relationships between them but also includes relationships between hypotheses and assessments of plausibility of lower level hypotheses [6.9-6]. Such lower level hypothesis may include hypotheses about the properties and behavior of situational items at any even lowest granularity satisfying certain information needs, for example, properties and behavior of basic situations (aggregates at the lowest level of granularity). Higher level hypotheses may include hypotheses about underlying causes of observed situational items. Automatic hypotheses generation is the most difficult part of SIA and it is not discussed in this report, in which it is assumed that hypotheses are generated by the users. Assessment of plausibility of hypotheses about situational items may include assessments of relationships between hypotheses at lower levels of granularity and plausibility of these hypotheses. Figure 6.9-4 shows the process of reasoning and predicting situations.
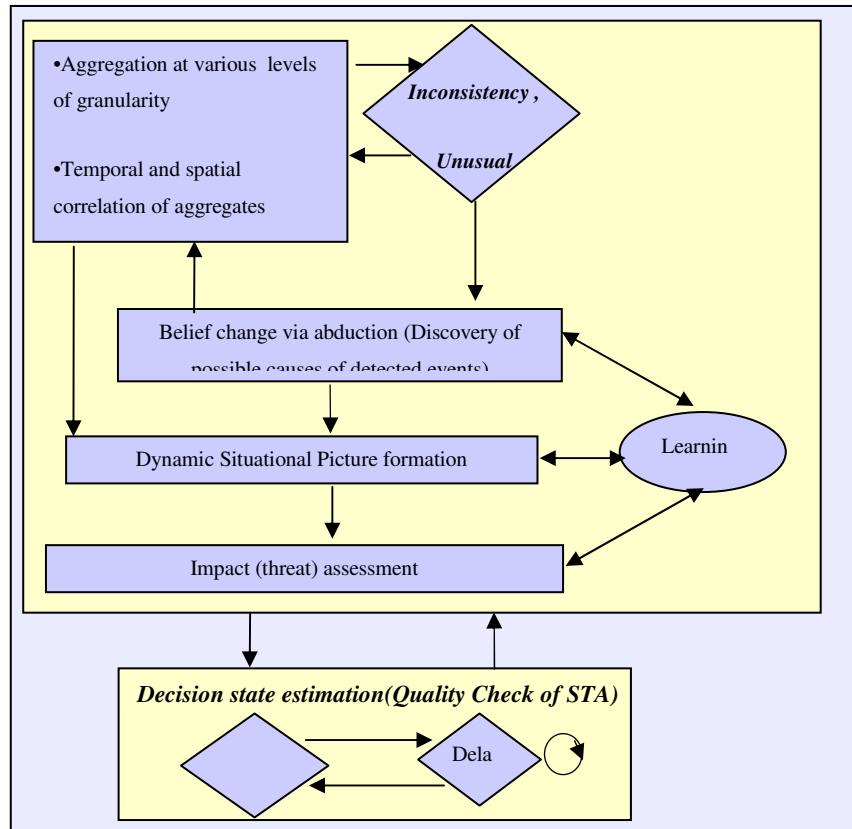
Figure 6.9-4. Processes of assessing current and predicted situations

6.9.4.2 Aggregation (Dynamic anytime clustering)

The process of aggregation is a core situation assessment task, the drawing together of selected objects into a common set. While the criterion for aggregate set membership can in general be perfectly arbitrary, aggregation is most frequently accomplished through the process of clustering, in which a quantitative similarity metric is applied to a population of candidate objects and their relationships, and sets (clusters) selected such that their intra-cluster similarity metric is in some chosen sense greater than the inter-cluster metric. Clusters abstract and summarize the distribution of entities in a selected space, suggesting useful generalization at the cluster level of granularity. Here we describe the clustering procedure implemented for aggregation in high level fusion for DIRE.

Desiderata for clustering schemes in the present application include multiple-resolution, speed, anytime calculation, flexibility and robustness. Clustering over a scale of resolutions is required to support the multiple levels of decision makers to be assisted in the disaster response context.

What a local precinct commander might consider a casualty cluster relevant to his decision making, and what the state emergency operations center commander might, differ widely in scale and resolution yet both determinations must be supported. Speed is a basic consideration in a real-time response environment in which decision aid latency is an important measure of fusion system performance. Since clustering will be applied over various spaces and with different similarity metrics in the varied tasks internal to situation assessment and impact prediction, flexibility of the clustering scheme is required. Finally, given the uncertainties in the physical and reporting environments, robustness of the clustering in the face of noise, delay, reporting and instrumental error is essential.

An acceptable level of each of these four criteria was achieved by employing a dynamic anytime clustering scheme based on Shi, Song and Zhang's Shrink Clustering approach [6.9-16]. Using relaxation dynamics evoking the law of gravitational attraction, individual candidate points scattered about an n-dimensional Euclidean space drift together into clumps, each clump ultimately representing a gravitationally collapsed cluster. The cluster-labeled points are then cast back to their original positions to form the cluster sets. The relaxation algorithm is implemented in parallel on multiple rectilinear lattices in which all candidate points in a lattice cell move as a single rigid body. Multiple lattices at various scales are employed and combined into a single multiresolution array of clusters. As discussed in [6.9-16] the results of this procedure compare favorably in speed, accuracy and robustness to more computationally demanding schemes.

Having identified clusters at each level of resolution using the Shrink Clustering approach, we employ a somewhat different cluster evaluation and cluster set selection process than they advocate. They choose a definition of cluster quality, which is based on compactness measures [6.9-16]. This quality measure may be computationally more intensive than suitable for the large datasets, which must be processed rapidly in the present application, is not monotonic with cluster density, and does not yield identification of a unique overall cluster set that best characterizes the state of entity aggregation at that time.

The cluster quality measure we employ is

$$Q_i = \frac{\sqrt{n_i}\,\sigma_i}{\min_j \left\| \mu_i - \mu_j \right\|},$$

where n_i is the population of the ith cluster, σ_i the unbiased estimate of its variance measured in the feature subspace, μ_i the estimated centroid, and the minimum is taken over all clusters identified at the same level of resolution. Assigning each cluster its quality Q_i, the set of all clusters at all levels of resolution is searched to determine the cluster set Σ^* with the highest average quality. The search is constrained such that the cluster sets considered partition the data with minimal overlap. In general Σ^* contains clusters from different lattices of the original Shrink Clustering procedure, and no other set of clusters has higher average quality.

Dynamic anytime clustering at time t can be accomplished either by reclustering at each time *de novo*, or by updating the previous cluster set $\Sigma^*(t-1)$. The updating approach offers potential computational savings, and has the advantage, within a prediction-correction framework, of being informed by past cluster results. This promises a more stable picture of the cluster dynamics to the decision maker, an important consideration in the disaster response context. We update $\Sigma^*(t-1)$ to the current cluster set in two steps. First, data newly arrived in the update interval [t-1,t) is used to correct the previous clustering, yielding an intial estimate of the clustering at t. Assuming that the update data is a small set of additions, deletions and changes relative to the existing database, a perturbational approach is used. The distribution of centroids of the shrunk clusters from t-1 determine the cluster labels (if any) for new datapoints, deleted data reduce the "mass," or attractiveness, of their previous clusters, and small changes in attributes are assumed not to affect a datapoint's cluster label. Second, the corrected clustering is combined with the predicted using an alpha-beta filter. The resulting cluster set $\Sigma^*(t)$ tends to evolve smoothly over time, with strong changes to the pattern of clusters and their properties only when the predictions based on past cluster results are significantly inconsistent with the most current data. Full reclustering to eliminate errors introduced by the perturbation model is done when the cumulative new data since the previous full reclustering exceeds a selected fraction of the total database at that time.

Depending on user needs aggregations can be performed based on similarity of features belonging to any feature subspace. For example, some decision makers may want to know the pattern of casualty location and therefore, aggregation for obtaining this information can be performed based on casualty location. Other decision makers may want to know the pattern of

location of a certain type of injuries. In this case aggregations will be based not only on x,y coordinates of the casualties but also on the type of injury. Aggregations can be also based on relationship characteristics or time intervals. The aggregation results depend heavily on the quality of results of the Level 1 fusion processes.

6.9.4.3 Correlation of situational items

Temporal reasoning about behavior of situational objects (aggregates at different levels of granularity) requires an association process, which correlates the situational objects identified at a certain time or time interval with situational objects identified at a different time or within a different time interval. This association process corresponds to reasoning about the *identity* of aggregates. Aggregates in the early post-disaster environment may be vaguely defined due to uncertainty associated with characteristics of objects obtained at the Level 1 fusion or characteristics of aggregates at lower level of granularity. The reasoning about aggregate identity is complicated by the fact that the information on the identity of members of aggregates is not known with certainty, and their characteristics and therefore cluster characteristics are also uncertain.

Following [6.9-17], in which the author was concerned about relationships between vague spatial regions, we conduct temporal association of aggregates (temporal reasoning about aggregates identity) by reasoning about such topological relationships as disjoint, touch, overlap (strong and weak), covers, covered by, contains and contained by. Unlike the authors of [6.9-17], we define these relationships not in the physical space but in the aggregate characteristic space, which includes such features as area and distribution of the members of aggregates. As in [6.9-17] we call two aggregates identical if they are disjoint, touch, or weakly overlap, and distinct otherwise (see Figure 6.9-5).

Figure 6.9-5. Spatial relationships between aggregates (from [6.9-17])
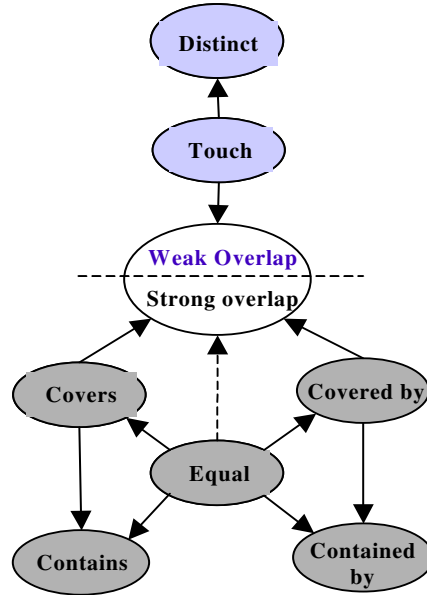
The temporal association thus requires a criterion for distinction between the weak and strong overlap, which is defined in [6.9-17] by the ratio of the area of the regions intersection to the area of the smallest region. We use the following criterion to classify overlap as weak or strong.

Let $Cl_t = (Cl_t^1,.....,Cl_t^N)$ be a set of aggregates identified at time t and $Cl_{t-1} = (Cl_{t-1}^1,.....,Cl_{t-1}^M)$ be a set of aggregates identified at time t-1. $\forall Cl_t^n$ a set of clusters $O(Cl_t^n) = (Cl_{t-1}^{m_i} \mid Cl_{t-1}^{m_i} \cap Cl_t^n \neq \varnothing)$ contains all aggregates at time t-1, which overlap with aggregate Cl_t^n.

Aggregate Cl_{t-1}^m discovered at time t-1 and aggregate Cl_t^n discovered at time t strongly overlap if $Cl_{t-1}^m \cap Cl_t^n \neq \varnothing$ and $F(Cl_t^n, Cl_{t-1}^m) > \alpha$.

α ($0.5 < \alpha < 1$) is a selected threshold and

$$F = \min(\frac{|Cl_{t-1}^m \cap Cl_t^n|}{\min(|Cl_{t-1}^m|, |Cl_t^n|)}, \frac{P(Cl_{\min})|_{Cl_{t-1}^m \cap Cl_t^n}}{\min(P(Cl_{t-1}^m), P(Cl_t^n))}),$$

where $P(Cl)$ is the expected number of members in aggregate Cl given all fused data reported for that cluster (cluster population), $Cl_{min} = \arg(\min(P(Cl_{t-1}^m), P(Cl_t^n)))$, $P(Cl_{min})|_{Cl_{t-1}^m \cap Cl_t^n}$ is the expected number of members of cluster Cl_{min} in $Cl_{t-1}^{m_i} \cap Cl_t^n$, $|Cl|$ the volume of cluster Cl in the feature space. If aggregation is conducted in 2-dimentional space (x,y) $|Cl|$ is the area of aggregate Cl. If $Cl_t^n \cap Cl_{t-1}^m = \varnothing, \forall Cl_{t-1}^m$, Cl_t^n is a new aggregate. If $Cl_{t-1}^m \cap Cl_t^n = \varnothing \; \forall Cl_t^n$, aggregate Cl_{t-1}^m is said to have terminated by time t.

Aggregate identity, and the behavior of its characteristics along with spatial relationships between clusters and their behavior, are used for casualty and damage assessment, resource allocation, discovery of possible underlying causes for assessed behavior, and impact prediction.

6.9.4.4 Characteristics and behavior of situational item

As it was mentioned in section 6.9-.5.1 the process of building a situational picture includes dynamic generation of hypotheses about characteristics and behavior of situational items at different levels of granularity. These characteristics and behavior characterize situations and represent essential elements of information for users at each level of the hierarchy. For instance, information about the location of a damaged bridge and the expected level of reduced capacity of this bridge would be of interest to an ambulance dispatcher, while a list of hospitals inaccessible within a reasonable time from certain clusters due to the reduced capacity of this bridge have to be reported to the overall incident commander.

Below we present the description of characteristics of the casualty situation. The characteristics of cluster situation provides the answer to one of the most important users' question: where are the casualties, and supports optimal ambulance dispatching, prediction of change of the transportation system capacity, prediction of required hospital capacity, and possible existing of hazardous events, etc. At the same time the special attention paid to this elementary situation is explained not only by the fact that this is one of the key elementary situations but also by the problems related to estimating characteristics of casualty aggregations. Aggregates of casualties may lose and gain members due to a certain percentage of casualties being picked up by ambulances, a certain percentage of unidentified casualties being transported by private vehicles and others moving off on foot. Building blocks for casualty situations are

aggregations of casualties obtained as the result of clustering in the feature space of associated and fused casualty reports produced by the lower level fusion modules. There is the following uncertain information characterizing each casualty:

1. probability of false alarm,

2. location coordinates with an uncertainty ellipse and a jurisdiction designator,

3. casualty ID if known,

4. last update time and the type of the last associated report (e.g. report that the casualty has been picked up, delivered to hospital, or simply observed),

5. a vector of probabilities of injury severity level (the description of severity levels are presented in table 3),

6. vectors of probabilities for the casualty reported age, race, and sex,

7. the number of reports associated with this casualty and reporter class for each report.

Detailed description of how these characteristics are obtained is presented in Section 6.2. Major cluster characteristics providing a subset of essential elements of information are discussed below. Casualty clusters characteristics (clustering is based on various features, such as x,y coordinates, level of injuries, type of injury, etc or a combination of these features):

1. Location

2. Boundaries

3. Area

4. Expected level of injuries,

5. Distribution of injury types

6. Survivability,

7. Density

8. Expected number of casualties of certain type

Expected number of casualties in a cluster

Expected number of casualties is computed by taking into account information obtained from Level 1 fusion and domain knowledge about behavior of casualties of various severities. The literature summarizing the experience of a previous earthquake suggests that within first 4 hours after the initial shock 100% of severity 1 casualties, 33.5% of severity 2 and 2.6% of severity 3 can be expected to walk away from their place of injury.

Let α_j be the total fraction of "walk-outs" of severity j, $\Pr_j^i$ be the probability of casualty i to have severity j, T_i is the time, at which casualty i was reported the first time, and $R_i(t)$ be the reliability of the associated and fused reports about track ID i at time t. If $f_j(t)$ is a probability density function for "walk-out" casualties of severity j (assumed to be uniformly distributed in our simulations over the first four hours following the earthquake event) then the expected number of casualties to walk-out from cluster n by time t is $W^n(t) = \sum\limits_{i \in cl^n(t)} \sum\limits_{j=1}^{3} R_i(t) \beta_j \Pr_j^i(t)$,

where $\beta_j = \alpha_j \int\limits_{T_i}^{t} f_j(x)dx$, Then the expected number of casualties in cluster n at time t is

$$E(N^n(t)) = \left\lceil \sum\limits_{i \in cl^n(t)} R_i(t) - \delta^n(t) \cdot W^n(t) \right\rceil,$$

where $\delta^n(t) = 0$ if cluster n is a new cluster at time t, and 1 otherwise.

The ratio $\sum\limits_{i \in cl^n(t)} R_i(t) / |cl^n(t)|$, where $cl^n(t)$ is cluster n at time t, is an important measure of reliability of cluster characteristics.

Expected level of injury severity of cluster n at time t can be approximated

$$E(S^n(t)) = \Sigma_{sl} / E(N^n(t)),$$

where Σ_{sl} is computed as follows:

$$\Sigma_{sl} = \sum_{\substack{i \in cl^n(t) \\ i \notin cl^n(t-1)}} \sum_{j=1}^{4} j \cdot \Pr_i^i + \sum_{i \in cl^n(t) \cap cl^n(t-\tau)} (\sum_{j=1}^{3} (1-\beta_j) i \Pr_j^i + 4 \Pr_4^i).$$

Cluster area

is computed as a number of cells of the highest resolution considered.

Cluster location

is the center of gravity calculated by taking into consideration uncertainty of casualty location (σ_j^i) and expected level of injury of each casualty $E(S_j^i)$.

$$\overline{r}_j^n = \frac{\displaystyle\sum_{i \in cl^n(t)} \frac{\overline{r}_j^i E(S_i^n)}{(\sigma_j^i)^2}}{\displaystyle\sum_{i \in cl^n(t)} \frac{E(S_i^n)}{(\sigma_j^i)^2}}.$$

Cluster density

$$D^n(t) = A^n(t) / E(N^n(t)).$$

Cluster distribution

Spatial (in the features space) cluster distribution at different level of granularity and resolution, e.g. area building damage, area injury level, area survivability level.

Cluster behavior

(at different levels of granularity). Cluster behaviors are presented by the change of cluster characteristics.

Examples of characteristics and behavior of other important elementary situations (aggregations of similar situational items based on location similarity at a certain time and a certain space resolution) are presented in Table 6.9-3.

Table 6.9-3. Examples of characteristics and behavior important elementary situations

Situational items	Characteristics	Behavior
Hazardous Situation (secondary threat)	Where (location, area, boundaries) What kind, level Boundaries of affected area Affected area	Area (increased/decreased) Change of Boundaries (contained, spreading) Speed of spreading Direction of spreading

AFOSR F49620-01-1-0371

Transportation system situation	Regional road damage estimation: Lost of Connectivity between certain areas due to damage (e.g. essential bridge + other roads damage) Impassible areas • Regional road capacity estimation: current and behavior Congestion areas Impassible areas Lost of connectivity between certain areas due to congestion • States of the access points to the disaster area and critical points within the disaster area: location, capacity, condition (level of damage if any)	Regional road damage behavior: Regained Connectivity between certain areas due to damage • Regional road capacity estimation: Congestion areas (increased/decreased) Regained connectivity between certain areas due to eased congestion • States of the access points to the disaster area and critical points within the disaster area: Change in capacity and condition
Ambulance situation	Number of available ambulances in a certain area Location and the number of ambulances with casualties, Number of ambulances are on the way to the hospital/hospitals in a certain area Pick-up/delivery time	Change in the number of available ambulances in a certain area (increased/decreased) Change in the number of ambulances on the way to the hospital/hospitals in a certain area

Building damage situation	Characteristics of clusters of damaged buildings at different levels of granularity Location Boundaries Area Expected level of damage, Distribution of damage types Density Characteristics of regional building damage	- change of area (increase/decrease) change of boundary change of density (increase/decrease) Change of the expected level of damage Change of characteristics of regional building damage
Hospital situation	Characteristics of hospital aggregations (clustering is based on various features, such as location, level of hospital damage, type of hospital damage (structural, electricity, water), etc or a combination of these features) Level of damage if any Available capacity Location Patients in OR, ER Unreachable from certain locations Hospital resource situation (doctors, supplies)	Behavior of characteristics of hospital aggregations: Change f the level of damage change in capacity Change of unreachable locations Change of resource situation

Derived situations

Derived Situations are defined by inter-class relationships between various situational items. Examples of derived situations and their characteristics are:

1. Derived situation defined by spatial relationships between hospitals, clusters of casualties and transportation situation. Examples of characteristics:

2. Location, ID, and characteristics of clusters unreachable from a certain hospital

3. Location, ID, and characteristics of a hospital unreachable from a certain clusters

4. Situation defined by spatial relationship of hazmat area boundary and clusters of casualties. Example of characteristics

5. IDs of clusters within hazmat boundaries

6. Situation defined by spatial relationship of ambulance situation, transportation situation, and clusters of casualties. Examples of characteristics:

7. A cluster of free ambulances close to a certain clusters (distance)

8. Number of casualties exceeds the overall capacity of ambulance available in a certain jurisdiction.

Examples of possible predicted impact of current elementary situations on other elementary situations are presented in Table 6.9-4.

Table 6.9-4. Predicted impact

Casualty situation (pattern of casualty clusters and their characteristics)	Predicted hospital situation (predicted arrival rate and change in capacity) Predicted resource situation (predicted amount of resources required) Predicted transportation situation (congested areas, impassible

	areas)
	Predicted area of overwhelmed communication system
Hazardous situation (toxic spill)	Predicted affected area
	Predicted casualty situation (new casualties clusters)
	Predicted shelter situation (change in capacity)
	Predicted transportation systems situation (e.g., impassible areas, congestion die to evacuation)
	Predicted hospital situation (predicted casualty arrival rate and change in capacity)
	Predicted area of overwhelmed communication system
Transportation situation (transportation facility damage, congestion area)	Predicted impassible areas
	Unreachable hospitals (helicopter is needed)
	Unreachable casualty clusters
	Unrealistic hospital-casualty cluster pair (travel time is too long)
	Change in hospital capacity
	Change in shelter capacity
	Unrealistic hospital-casualty cluster pair (travel time is too long)

Characteristics and behavior of situation items provide a basis for dynamic reasoning about current and predicted plausible states of the environment. Under the assumption that that the underlying causes of the estimated characteristics and behavior of situational items is known, the reasoning about current and predicted situations can be performed by determining the patterns of relationships and behavior of these characteristics and behavior in the context under consideration. At the same time if situational items exhibit some abnormal characteristics and behavior

inconsistent with domain knowledge and characteristics and behavior of situation items within current context, a set of beliefs about the environment may need to be updated. The next section will describe the method of detecting inconsistency and the abductive reasoning method for belief update under uncertainty introduced in this report.

6.9.4.5 Belief update

Characteristics and behavior of situational items are constantly updated by newly processed observations. The current set of plausible states is constantly updated and new hypotheses about the plausible state of the environment (new context) capable of explaining new characteristics and changes in the behavior of situational items are regularly generated and evaluated. These unknown new situations are usually manifested by unexpected properties and behavior of situational items inconsistent with the current set of beliefs about the state of the world and therefore belief update may be required. Many belief update methods give priority to this new information and its consequences and abandon some old beliefs to preserve consequences. In the post disaster environment observations and knowledge about situational items, their behavior and relationships are uncertain and, therefore it is necessary to account for this uncertainty while updating the current set of beliefs. In the uncertain dynamic environment belief update can be carried out by first seeking some explanations or underlying causes of these inconsistent observations and incorporating these explanations, if found, into a new set of beliefs. Possible explanations can be found as the result of abduction comprising generation of hypotheses about the underlying causes of these inconsistent observations and reasoning about plausibility of such hypotheses.

This abductive process of reasoning from effect to cause requires [6.9-6, G18]:

1. constructing or postulating possible hypotheses and the states of the world explaining observations

2. computing plausibility of these hypotheses

3. selecting the most plausible hypothesis from among these.

The process of hypothesis evaluation has to take into account the following considerations [8.9-18]: to what degree is the hypothesis to be selected better than alternatives? How credible is the hypothesis by itself, independently of considering the alternatives, i.e. one should be cautious about accepting a hypothesis even if it is clearly the best one we have if it is not sufficiently plausible in itself. Finally, what is the reliability of incoming data, which requires explanations.

Abductive inference starts with discovery of characteristics inconsistent with the current state of knowledge and behavior of attributes and relationships between the associated situational items. In the present model this anomalous behavior, or data inconsistency is detected by significant deviation in the behavior of attributes and relationships of situational items from expected, given the current state of knowledge. Examples of such behavior may include discovery of a new aggregate or situation, a specific pattern of spatial and/or temporal relationships between aggregates, or a significant deviation of the behavior of one or several characteristics of an aggregate or a situation from the expected average behavior of the characteristics of similar aggregates or situations. Classes of similarity of aggregates can be defined by clustering of the environmental features related to aggregate formation. For example, the expected number of casualties in a cluster depends on the severity and type of damage in the area, time of the day and the rate of casualty discovery, which in turn depends on the possible number of civilians, police, and ambulances reporting the casualties (density of roads, proximity to the hospital or density of population in the area).

Discovery of a deviation from the expected is followed by construction of a set of hypotheses (possible causes of the discovered deviation) about the situation. Then beliefs in each of these hypotheses are evaluated. Resulted beliefs are used to decide whether there is enough information to select one of the hypotheses and which hypothesis to be selected.

Automatic hypothesis generation is the most difficult process to implement within SIA and is not assumed here, so that hypotheses are assumed to be generated by human experts. Below is a set of possible secondary incidents, which may be considered in the early earthquake environment:

1. Aftershocks

2. Unreported facilities damage (e.g., bridges)

3. Hazardous incidents (Toxic spills due to road and bridge damage, damaged hazardous facilities, ruptured gas pipelines)

4. Fire

5. Delayed severe building damage

6. Act of sabotage or bad judgment

After hypotheses are generated and the belief supporting all the hypotheses is estimated, the decision is made on whether we should consider characteristics and behavior of situational items in the context of earthquake only but in the context of earthquake along with a secondary incident. Considering situational items, behavior, and relationships between them allows for better prediction of situational impact and more appropriate and swift actions on mitigating the consequences.

A reasoning framework introduced in this report for SIA in the post-disaster environment is Belief Based Argumentation System (BAS), a generalization of the Probabilistic Argumentation System (PAS) (see, e.g. [8.9-19]), augmented with the set of relevant domain specific models such as hospital models and dynamic dispatch/routing models. Following [6] PAS can be described as an approach to non-monotonic reasoning under uncertainty, combining symbolic logic with probability theory for judging hypotheses about the unknown or future world by utilizing given knowledge. Logic is used to find arguments in favor of and against a hypothesis about possible causes or consequences of the current state. An *argument* is a defeasible proof built on uncertain assumptions, that is, a chain of deductions based on assumptions that make the hypothesis true, or false. Every assumption is linked to an *a priori* probability that the assumption is true. The probabilities can be understood in the traditional Kolmogorov-axiomatized way but also can represent subjective probabilities. The probabilities that the arguments are valid are used to compute the credibility of the hypothesis, which can then be measured by the total probability that it is supported by the totality of supportive and refuting arguments. The resulting degree of support corresponds to belief of the theory of evidence and is used to make a decision whether a hypothesis should be accepted, rejected, or whether the available knowledge is insufficient to form a satisfactory judgment at this time.

In the post-disaster environment accurate *a priori* probabilities that the assumptions are true are rarely available and expert subjective beliefs have to be used. Moreover, due to the high uncertainty characterizing the post disaster environment $P(A)$, expert subjective belief that assumption A is true, is not in general, equal to $1 - P(\neg A)$ and therefore PAS has to be generalized to utilize sub-additive subjective belief measures: $Bel(A) + Bel(\neg A) \le 1$. These subjective beliefs can be expressed in linguistic form, e.g., very high, high, low, very low with subsequent quantization of these linguistic values. The belief measures can be also represented numerically and be approximated by a function of the values assigned to attributes and relationships characterizing the state of environment and related to the assumptions. In some cases these belief measures can be the result of a combination of beliefs based on different characteristics with the Dempster rule. Beliefs in assumptions are combined to obtain beliefs in arguments, which favor and refute the hypotheses. These beliefs in turn are used to gauge the overall credibility of the hypothesis, measured by the total belief that is supported by arguments.

of a set of hypotheses (possible causes of the discovered deviation) about the situation. Then beliefs in each of these hypotheses are evaluated by identifying and combining with the Dempster rule of combination *pro* and *contra* arguments for them. Resulted beliefs are used to decide whether there is enough information to select one of the hypotheses and which hypothesis to be selected.

Let $\Theta = \{\theta_1, ..., \theta_K\}$ be a set of hypotheses under consideration. Given the open world assumption, this hypothesis set is not exhaustive and $Bel(\varnothing) \ne 0$. BAS is a tuple (A, P, ξ, B), in which as in PAS, $P = \{p_i\}$ is the set of propositions, $A = \{a_j\}$ is a set of uncertain assumptions, $\xi \in L_{P \cup A}$ is a knowledge based representing a set of rules (certain and uncertain). At the same time unlike to PAS $B = \{bel_j\}$ is a non-additive dynamic beliefs associated with $A = \{a_j\}$. Argument Arg_{k_m} supporting (or refuting) each hypothesis θ_k are derived from the knowledge base and is a conjunction of propositions and assumptions for which θ_k becomes true (or false): $Arg_n(\theta_k) = \bigwedge_j a_{n_j} \bigwedge_k p_{n_k}$. The support of each hypothesis θ_k is defined as the disjunction of all minimal arguments supporting θ_k: $Arg(\theta_k) = \bigvee_n ArgP_n \bigvee_m ArgC_m$, where

$\vee_{n} ArgP_{n}$ is a disjunction of all arguments supporting hypothesis θ_{k} and $\vee_{m} ArgC_{m}$ is a disjunction of all arguments refuting hypothesis θ_{k}.

Beliefs in support of each hypothesis θ_{k} can be computed by utilizing beliefs in arguments the following way. Beliefs in support of and against of each assumption $a_{n_{j}}$ invoke simple support functions on a frame of discernment $\Omega_{n_{j}} = \{T, F\}$, with a single focal element (assumption i is true or false). Let us consider a mapping $M : \Omega_{n_{1}} \times ... \times \Omega_{n_{N}} \rightarrow \Theta$. Than simple support function μ_{k} with focus θ_{1} in support of argument $ArgP_{n}$:

$$\mu_{\operatorname{Arg}P_{n}}(\theta_{k}) = \prod_{\operatorname{Arg}P_{n} = \wedge_{j} a_{n j}} bpa_{a_{n_{j}}}(T), \quad \mu_{\operatorname{Arg}P_{n}}(\Theta) = 1 - \mu_{\operatorname{Arg}P_{n}}(\theta_{k}).$$

Similarly, a direct sum of the simple support functions over the set $\{\Omega_{m_{j}} \mid \wedge_{j} a_{m_{j}} = \operatorname{Arg}C_{m}, \forall m\}$ is mapped into a simple support function v_{j}:

$$v_{\operatorname{Arg}C_{m}}(\theta_{k}) = \prod_{\operatorname{Arg}C_{m} = \wedge_{j} a_{m j}} bpa_{a_{m_{j}}}(F), \quad v_{\operatorname{Arg}C_{m}}(\Theta) = 1 - v_{\operatorname{Arg}C_{m}}(\theta_{k}).$$

Then belief in each hypothesis, based on arguments pro and contra this hypothesis computed as a combination of μ_{k} and v_{j} for all k and j with the Dempster rule of combination. The result of this combination is used for decision state estimation.

6.9.4.6 Decision state estimation (Quality control)

As it was mention before, decision making on situation assessment requires consideration of decision quality, which has to be evaluated against time required for additional observations/computations. In addition, decision process on any hypothesis under consideration has to take into account that something totally unexpected and not included in he possible causes of the observed situational elements can happened.

Then the decision rule is as follows: *If $Bel^{t}(\varnothing) \geq \max(Bel^{t}(A)), \forall A \subseteq \Theta$ (the level of support for an unknown hypothesis exceeds the level of support for any hypothesis under consideration) then*

an expert is engaged to revaluate a set of hypotheses considered and/or a sensor management process is initiated. For example an expert observer can be dispatched to verify the incoming information. Otherwise, *if* $Bel^t(\Theta) \geq \max(Bel^t(A)), \forall A \subseteq \Theta$ (the of ignorance exceeds beliefs in any hypothesis) *then no decision is made until the next time step* when additional information arrives. Otherswise, if $BetP^t(\theta_k) \geq th(t)BetP^t(\theta_n) \; \forall n \neq k$ than select θ_k. Otherwise wait, Here $BetP^t(\theta_k)$ is the pignistic probability [6.9-20] of hypothesis θ_k at time t and *th(t)* is a time varying threshold.

The form of the threshold *th(t)* is context specific. It is considered within the class of decreasing convex functions and is equal to zero when it achieves a certain maximum value (a deadline). In certain situations, when decisions based on the resulting decision state estimations have very serious consequences, a sensor management process can be rapidly employed. Section 6.9.5.7 will illustrate the reasoning approach described above by applying it to discovery of a Hazmat incident.

6.9.4.7 Identifying unreported hazmat spill

DIRE is configured to model a Hazmat incident in which a colorless, odorless toxic gas is vented to the atmosphere as the result of the accidental or malicious rupture of a chemical storage tank. Dispersion of the material is modeled by a Gaussian plume driven by the wind field, resulting in primarily respiratory casualties. An excess of respiratory casualties in a given cluster, and its growth with the prevailing wind, is supportive of a hypothesis of a secondary Hazmat incident not yet discovered. Discovery of such an incident, as described below permits impact prediction and may drive targeted evacuations as well as additional constraints to the ambulance and police movement.

At fixed time intervals, shrink clustering (Section 6.9-.5.2) is used to identify the current set of casualty clusters Each cluster consists of a connected set of cells, which are used for the hierarchical cluster routine. Discovery of an unreported toxic spill is invoked by detection of unusually high percentage of respiratory injuries within certain casualty clusters at time t, and corroborated by reports of new respiratory casualties in spatial progression downwind of the discovered but unexplained respiratory cluster. In the absence of uncertainty, this new

information calls for update of the current belief that an expected percentage of respiratory injuries due to building damage are not higher than an *a priori* known value. In our case we select this value to be 10%, the number characterizing the fraction of non hazmat-related respiratory injuries reported during the 1994 Northridge earthquake. Due to uncertainty of observations and the current knowledge base it is advantageous to look for a possible underlying reason for this unusually high level of respiratory injuries before updating the current beliefs. We do not ignore the possibility that this high level of respiratory injury is due to building damage as a result of the earthquake.

We consider a two hypothesis frame of discernment $\Theta = \{\theta_1, \theta_2\}$, where θ_1 is a hypothesis that a toxic spill occurred and θ_2 is a hypothesis that the excessive respiratory injuries are the result of structural damage only. We also assume that there might be an unknown cause (open world assumption) and that the plausibility that there is more than one toxic spill within a certain time interval is negligible.

The arguments used to compute beliefs supporting or rejecting a hypothesis represent a conjunction of propositions and uncertain assumptions about characteristics and behavior of "suspicious subclusters" and spatio-temporal relationships between such subclusters as well as between such subclusters and other clusters. Suspicious subclusters at time *t* are subclusters comprising connected cells with the expected number of respiratory injuries in each cell above the threshold defined by the expected value and the deviation of respiratory injuries (7% in our case).

A set of suspicious subclusters at time t, $SC_t = \{SCl_t^i\}$, is represented as a union of 3 subsets: $SC_t = P_t^1 \cup P_t^2 \cup P_t^3$, where P_t^1 is a set of subclusters formed at time t, P_t^2 is a set of subclusters formed before time t but not suspicious at time $t-1$, and P_t^3 is a set of suspicious subclusters, which were suspicious at $t-1$.

Below are definitions of the relationships "between", "close", and "neighbors" used in the reasoning processes. These relationships can describe *intra* relationships between subclusters, between clusters as well as *inter* relationships between clusters and subclusters of other clusters [3]:

1. Clusters Cl_t^i and Cl_t^j are considered *neighbors* at time t if the line connecting their centroids does not intersect any other clusters. Relationship *neighbors* is reflexive, symmetric but not transitive. N_t^i denotes a set of neighbors of cluster i (Cl_t^i) at time t.

2. Clusters Cl_t^i and Cl_t^j are called *"close"* if the distance between the centroids of these clusters is less then a threshold: $Dist_{ij} < \max(W \cdot \Delta t, a \cdot \max_{ij}(D_i, D_i))$, where W is the wind speed, Δt is the time step considered, D_i, D_j are maximum diameters of Cl_t^i and Cl_t^j, respectively, and a is a constant.

3. Cluster Cl_t^k is said to be *between* clusters Cl_t^i and Cl_t^j if $Cl_t^k \in N_t^i \cap N_t^j$ and Cl_t^k is within the box around both Cl_t^i and Cl_t^j, and clusters Cl_t^i and Cl_t^j are close.

Specific propositions considered for Hazmat spill discovery include propositions characterizing wind direction as well as cluster topology and topology temporal behavior (e.g., new, disappearing clusters and subclusters):

$P1$: wind direction

$P2(SCl_t^j) : SCl_t^j \in P_t^1$

$P3(SCl_t^j) : SCl_t^j \in P_t^2$

$P4(SCl_t^j) : SCl_t^j \in P_t^3$

$P5(Cl_t^n, SCl_t^m) : Cl_t^n \in N_t^m$ (cluster *n* is a neighbor of suspicious subcluster *m* at time *t*)

$P6(Cl_t^j) : Cl_t^j \notin SC_t$ (cluster *j* is discovered at time *t)*

It is necessary to note that in the uncertain environment cluster topology and topology behavior declarations are uncertain and represent assumptions. In our pilot study we assume that their truth is known with certainty and consider them propositions.

Assumptions about suspicious subcluster characteristics and their behavior

A1: The expected fraction of respiratory injuries in a subcluster indicate Hazmat (how "suspicious" is this subcluster?)

A2: The expected fraction of respiratory injuries in a subcluster is increasing.

A3: Subcluster area is growing.

A4: Subcluster center is moving downwind.

A5: Subcluster center is moving upwind.

A6: Subcluster centroid is moving downwind.

A7: Subcluster centroid is moving upwind.

A8: Cluster Cl_t^i (subcluster SCl_t^i) is located downwind from subcluster SCl_t^j.

A9: SCl_t^k is *between* clusters Cl_t^i and Cl_t^j.

A10: Cl_t^k is *between* subclusters SCl_t^i and SCl_t^j.

Each assumption is assigned a belief measure, which represents expert belief that this assumption is true. In our example these belief measures are modeled as functions of the behavior of values of suspicious cluster characteristics and mereotopological *intra* relationships between neighboring subclusters and *inter* relationships between subclusters and neighboring clusters. Let $\Omega = \{\omega_1^{Al}, \omega_2^{Al}\}$, where ω_1^l is a hypothesis that assumption l is true and ω_2^l is a hypothesis that assumption l is not true. Then for each assumption we model the measures of belief as follows.

For assumptions $A1 - A3$:

$$bpa(\omega_1^{Al}) = \frac{\lambda_l}{1 + \alpha_l e^{-\beta_l \cdot X_t^l}}, \quad bpa(\omega_2^{Al}) = 0,$$

where $\alpha_l, \beta_l, \lambda_l$ are parameters, *l=1,2,3,9,10*. For $l = 1$ X_t^l is the fraction of respiratory injuries and belief is based on the "level of suspiciousness". For $l = 2,3$ X_t is the relative difference between the change of the subcluster characteristics under consideration

(ΔY_t^l) at time t and the absolute value of an average change of magnitude of these characteristics (avg_{t-1}) up to and including time t-1:

$$X_t^l = |\frac{|\Delta Y_t^t| - avg_l^{t-1}}{avg_l^{t-1}}|,$$

where Y_t^l is the fraction of respiratory injuries, if $l=2$ and the suspicious subcluster area if $l=3$.

For $l=9$ X_t^l is the distance between the centroid of SCl_t^k and the line connecting centroids of Cl_t^i and Cl_t^j. For $l=10$ X_t^l is the distance between the centroid of Cl_t^k and the line connecting centroids of SCl_t^i and SCl_t^j.

For assumptions $A4 - A8$:

$$bpa(\omega_1^{A_l}) = \frac{1 + (-1)^l \cos(\phi_l)}{2} \chi_l, \quad bpa(\omega_2^{A_l}) = 0, n .$$

where ϕ_l is the angle between the wind direction and the direction of movement of the geometrical center ($l = 4, 5$), or the center of gravity ($l = 6, 7$), or the vector from the center of gravity of SCl_t^j to the center of gravity of Cl_t^i ($l = 8$) and χ_l is a scaling parameter.

Finally, arguments built from these propositions and assumptions corroborating and refuting the toxic spill hypothesis are composed. Sets of arguments differ slightly for $P_t^1, P_t^2,$ and P^3 because of the temporal difference in behavior of their characteristics. Below are assumptions considered for subclusters $SCl_t^i \in P_t^2$ (P2).

Corroborative arguments ($ArgP_k$) include:

$ArgP_1$: The expected fraction of respiratory injuries in subcluster $SCl_t^i \in P_t^2$ indicate a toxic spill, this cluster is a *neighbor* of subcluster $SCl_t^j \in P_t^3$ and is located downwind from $SCl_t^j \in P_t^3$: $A1(SCl_t^i) \wedge A8(SCl_t^i, SCl_t^j) \wedge \Pr 2(SCl_t^i) \wedge \Pr 3(SCl_t^j) \wedge P5(SCl_t^i, SCl_t^j)$.

$ArgP_2$: A suspicious subcluster area is growing downwind:

$A3(SCl_t^i) \wedge A4(SCl_t^i) \wedge P2(SCl_t^i)$

$ArgP_3$: Respiratory injuries are growing downwind: $A2(SCl_t^i) \wedge A6(SCl_t^i) \wedge P2(SCl_t^i)$

Arguments refuting the toxic spill hypothesis ($ArgC_j$) include:

$ArgC_1$: A suspicious subcluster area is growing upwind:

$A3(SCl_t^i) \wedge A5(SCl_t^i) \wedge P2(SCl_t^i)$

$ArgC_2$: Respiratory injuries are growing upwind: $A2(SCl_t^i) \wedge A7(SCl_t^i) \wedge P2(SCl_t^i)$

$ArgC_3$: A suspicious subcluster $SCl_t^i \in P_t^2$ is between clusters Cl_t^n and Cl_t^m, which are located along the wind direction and do not contain suspicious subclusters:

$\Pr 2(SCl_t^i) \wedge A9(SCl_t^i, Cl_t^n, Cl_t^m) \wedge A8(Cl_t^n, Cl_t^m)$

Beliefs in support of each assumption i invoke simple support functions on a frame of discernment $\Omega_i = \{\omega_{1i}, \omega_{2i}\}$, with a single focal element ω_{1i} (assumption i is true). A direct sum of the simple support functions over a set $\{\Omega_i^t \mid \bigwedge_i Ai = A\,\mathrm{rg}\,P_k, \forall k\}$ is mapped then into a simple support function μ_k with focus θ_1 (*pro* Hazmat):

$$\mu_k(\theta_1) = \prod_{i: \wedge Ai = A\mathrm{rg}\,P_k} bpa(\omega_{1i}), \quad \mu_k(\Theta) = 1 - \mu_k(\theta_1).$$

Similarly, a direct sum of the simple support functions over the set $\{\Omega_i^t \mid \bigwedge_i Ai = A\,\mathrm{rg}\,C_j, \forall j\}$ is mapped into a simple support function v_j with focus θ_2 (*contra* toxic spill). :

$$v_j(\theta_2) = \prod_{i: \wedge Ai = A\mathrm{rg}\,C_j} bpa(\omega_{1i}), \quad v_j(\Theta) = 1 - v_j(\theta_1).$$

Then belief in each hypothesis, based on arguments built for each suspicious subcluster is computed as a combination of μ_k and v_j for all k and j with the Dempster rule of combination. The final decision is based on the combination of beliefs obtained for subclusters

belonging to clique of neighboring subclusters. Selection of a certain hypothesis is based on the decision process described in Section 6.9.5.6.

Discovered at time t Hazmat clusters are used for prediction of consequences of identified hazmat incidents. First a geometry of the Hazmat clusters, a vector field of the wind direction, and speed direction allow for identifying the affected region and for predicting the region, which will be affected within an hour.

The location and the area of identified current and predicted affected regions provide decision makes with information necessary to make swift decisions on evacuation and closing the affected area for non-necessary traffic. This information with also allows for prediction of new casualties, new congested regions, and change of capacity of hospitals and shelters.

6.10 Layered Hybrid System Architecture

The CUBRC/UB PRET proposal [6.10-20] of February 2001 which was approved for funding by the AFOSR and which asserts the goals and research plan of this project, includes the following language:

A body of knowledge has been acquired that provides a a generalized methodological foundation for the design and development of Level 1 fusion processes across a range of operational requirements and applications. Fusion, however, embodies two other inference-generating levels, Level 2 and Level 3, associated with Situation Estimation and Threat or Impact Estimation. However, research on the technologies and techniques necessary to achieve an automated capability to produce such estimates, the result of associating and fusing considerably larger amounts and wider varieties of data and knowledge, has been much less than for Level 1. The complexity of architecting such systems as well as defining and designing each of the subprocesses is much higher than for Level 1. The dilemma with the research and capability shortfall at Levels 2 and 3 is that mid-level to upper-level Air Force commanders, as distinct from tactical decision-makers, lack the necessary automated tools to deal with the evolving new world risk environment.

The overall objective of the proposed research is to develop, evaluate and document an overall engineering methodology with which to approach these higher level problems; such

methodology will be an important part of a cost-effective, reusable approach to these problems for the fusion community. The combined results of relevant and transitionable techniques grounded in theory and quantitatively evaluated offer the potential to overcome a major shortfall in information fusion science.

In light of the above, the project team considered the general problem of architecture for high level fusion in the emergency response context. The following section presents a relevant and transitionable architectural framework for high level fusion systems. The framework includes general architectural recommendations, analysis of the performance envelopes for the recommended architectural configurations with respect to major problem space attributes, and selection matrices for major categories of fusion application scenarios. Such scenarios include battlespace applications, natural and man-made disasters, intelligence gathering and evaluation, robotics and autonomous vehicles, maintenance of complex systems, patient monitoring systems, air traffic control, intelligent transportation systems and other relevant application areas of information fusion. Less relevant applications domains, such as data fusion for agricultural or land-use management, are not considered.

6.10.1 Background and architectural dimensions

While there have been many data fusion and information fusion papers written, and systems built, there is not a lot of useful relevant literature on architectures and architectural specifications for high level fusion. As pointed out by Kokar [6.10-19], the architectural models which have been presented have not been specific enough to determine how to design systems that comply with their guidelines or whether a given existing system does so. He identifies three interesting informaton fusion architectures: the JDL model, the NBS model, and his own Formal Systems Architecture. The first two are process-flow architectures, his is a system architecture.

6.10.1.1 Process-flow vs. system architectural specifications

The architectures described in the literature are mainly of two categories: process-flow architectures and system architectures. By a process-flow architecture we mean architectures which partition the system in terms of *what* is done, while the system architectures dictate *where* it is done, in which software/hardware modules. If we understand these terms as ordinarily used

correctly, for instance, the JDL model is a process-flow architecture, while a blackboard is a system architecture. In order to accomplish our goals, we believe we need to specify both: the process-flow, or partition of functions to be performed and connectivity among those functions, as well as a system architecture, or partition of hardware/software modules and their data links.

In our understanding, neither architectural specification, the what or the where, goes to *how* the processes or objects are to be computed. That is the issue of implementation. Implementation involves the selection of algorithms, data structures, communications protocols, etc. and does not have to be part of the architectural discussion. Except for the issue of flexibility: is a given architecture suffiiently flexible to support the preferred implementation choices?

6.10.1.2 informaton fusion models

The literature cites dozens of informaton fusion models. Most of them are oriented toward data fusion rather than informaton fusion, that is, they do not adequately support the abstraction of individual tracks and entities to represent the relationships among them, or the understanding of the situations and likely consequences these relationships produce in light of domain knowledge and doctrine. Important examples are Ah-Dhaher's multi-sensor data fusion architecture [6.10-1], Aude's robot contoller [6.10-4], Broder's spatial reasoning system [6.10-6], Kejun's L1 system architecture [6.10-18] and Yang's intelligent transportation system data fusion model [6.10-32]. Gorodetski's system [6.10-15], while capable of high level fusion, is optimized for L1, as is the toolkit they produced. Their basic operation is decision fusion (DEI-DEO in Dasarathy's taxonomy).

Of the models that are relevant for high level fusion, there are four that merit the most consideration:

1. The JDL model [6.10-35]

2. Dasarathy's model [6.10-10]

3. OODA model [6.10-33]

4. Endsley's model [6.10-34]

Three of the four are similar: the JDL, OODA and Endsley model. In each case there is a data-gathering phase (L0/L1, Observe, Perception), an understanding phase (L2, Orient, Comprehension) and a prediction phase (L4, Orient, Projection). Dasarathy's model breaks out three phases as well: data, features, and decisions, and creates a model taxonomy based on which of these phases are the inputs and output of the data fusion system.

Since they are similar, in our architectural choices we can afford to be agnostic among the JDL, OODA and Endsley model. Due to its history and wide acceptance, for most purposes these recommendations are based on the JDL model. Note that selection of this process-flow model does not specify a process-flow architecture, since there are many ways to instantiate the model in an architecture. But it allows us to name and locate the processes that the architecture we select specifies.

6.10.2 Architectural dimensions

Following Allouche [6.10-3], a useful way to characterize informaton fusion architectures is as points in a design space in which the coordinates are *architectural dimensions*. Three classes comprising 8 such dimensions are suggested:

1. Decentralization

    a. of control

    b. of processing

    c. of data

2. Autonomy

    a. of control

    b. of processing

3. Socialization

    a. degree of social reasoning (eg. aggression, persuasion)

    b. of organization (eg. command heirarchy, democracy)

    c.   of communication (eg. speech acts, KQML)

In addition we add these additional dimensions of the architectural problem:

4. Flexibility

    a.   Of implementation

    b.   Of control

5. Reconfigurability

6. Scalability

The use of these dimensions is to develop metrics with which to measure similarities and divergences among architectures, and identify problem domains favorable to regions of the architectural space. For instance a plain-vanilla blackboard architecture has low decentralization of control and data but high processing decentralization. There is low autonomy of processing and high autonomy of control. Low socialization on all dimensions. Moderate implementation flexibility, low control flexibility. High reconfigurability. High scalability. These attributes promote blackboard solutions for certain specific distributed problem-solving applications in which data are naturally centralized, the solution can be worked in small increments each requiring little domain knowledge or judgment.

6.10.2 Alternative architectural framemworks

Here the most successful current approaches to data fusion architecture are briefly reviewed. Elements of several will be employed in the subsequent recommendations.

6.10.2.1 Blackboards

The blackboard architecture is a popular choice for data fusion and has been used for informaton fusion. Llinas [6.10-16] reviewed 13 major systems using blackboard architectures in a paper published in 1993. Since then the intial blackboard conception has grown and been extended to include multiple blackboards, paired data-control blackboards, and networks of blackboards. [6.10-16] generally advocates looking at blackboards for informaton fusion because of their simplicity and low communications bandwidth compared to alternatives. Since then they

continue to be widely used. Valin [6.10-29] and Shabazian [6.10-25] report on the use of a blackboard for informaton fusion in a surveillance aircraft application. The CORTEX software development describe by Macieszczak [6.10-22] and Shabazian [6.10-24] is a knowledge based system whose system architecture is a blackboard which supports a knowledge based system.

Engelmore [6.10-11] is the basic resource for information on blackboard theory and practice at the end of the first generation of blackboard systems, benchmarked by HEARSAY II. Perhaps the most interesting blackboard model we have seen is Sutton's Bayesian blackboard, an architecture advocated for intelligence gathering and processing. Network-fragment theory is used to "quantify" the symbol search system through the construction of Bayes' Net fragments as the basic knowledge representaiton.

A key limitation for blackboard systems is the requirement for centralization of data. If the data is naturally spatially distributed, this can be forced by establishing datapaths to a central location. But this incurs costs: bandwidth, transmission error rates and QOS requirements, robustness, complexity.

A second key limitation is that the knowledge sources (KS) not communicate except via the blackboard. While most other original architectural specifications of the blackboard have been generalized over the years, this has not. This presents, in our opinion, a major inefficiency for certain kinds of what we call *interactive dialogs*. These concern exchanges of information between KSs in which KS1 communicates a small amount of information to KS2, who operates on that and returns it to KS1, who repeats. This can occur when 2 KSs need to interact intimately in order to produce a partial result. When this occurs in a blackboard setting, the blackboard controller is required to structure each communication act to get in on the blackboard and bring it to the other KSs attention. This may require many more cycles than the dialog itself.

A third key limitation is that the KSs not exercise any serious degree of autonomy. The blackboard controller is delegated that authority. Thus the KSs cannot use independent judgment, act intelligently, learn or otherwise behave in ways the blackboard controller did not predict.

6.10.2.2 Multi-agent systems

Intelligent agents (IA) are software objects with goals, intentions, communications capabilities, and some degree of autonomy. In this way they are distinguished from the KSs of blackboard systems, and from ordinary software objects. They can behave in non-transparent ways, and communicate what they choose to other agents or software modules. Multiagent systems are systems incorporating two or more intelligent agents.

There are various types of IAs: reactive, logical, Belief-Desire-Intention, Layered, etc. Of particuar interest for informaton fusion are the 2-pass layered agents [6.10-30]. Their process-flow naturally maps onto common informaton fusion process-flow models such as JDL. There is considerable current R&D community interest in applying IAs to informaton fusion problems.

An imporant limitation in the use of multi-agent systems architectures is the bandwidth requirements associated with messages and message-passing protocols. Languages like KQML are needed to rectify communications among IAs.

Another limitation of multi-agent systems is their opacity. The degree of autonomy of each IA means that its behavior in some situations will be difficult or impossible to predict. This means QOS guarantees will be at best probabilistic and their might be long tails on performance distributions.

There are various multi-agent system design methodologies that are in common use:

1. Gaia: determine agent roles, then interactions. Roles mapped into agent types. Then select service models and interaction models.

2. Wood and DeLoach's MA Sys Eng MASE: first capture goals, then use cases, refine roles, create agent classes, create conversations, assemble agent classes, system design.

3. UML based schemes and extensions

4. Daimler-Chrysler: model the task environment, analyze the model to extract roles, specify interactions, specify agents.

A main attraction of multi-agent systems is that it does not require centrality of data, and is thus consistent with distributed fusion. With heirarchical fan-in system architectural topologies such

as that in Dual Node Network, the natural I/O bandwidth reduction per fusion node makes load-balancing of computation and communication possible.

Brenner [6.10-5] finds blackboards and multi-agent systems the only real general alternatives for general distributed problem-solving. Gatepaille's general multi-agent system data fusion framework makes direct associations between data fusion functions and IAs, partitioning the situation assessment database among the IAs local memory stores. The first is a good idea (which we will adopt in our proposal below), but the memory partition does not distinguish between data that is needed by multi-agent interest groups vs. private data of individual agents. The T-10 demonstrator was built along her recommended lines.

6.10.2.3 Dual-node network [6.12-1]

A limitation is the recursive use of the same tripartite node decomposition for all levels of fusion and for all levels of resource management. It is arguable that as entities assume higher levels of abstraction, so must their transformations.

It appears the Dual Node Network properties should be explored for more general topologies. It may be that a regular fan-in topology becomes a more general network due to faults in the system, for instance, or due to ad hoc comm links established during emergency use. It is not clear what happens to the duality properties in more general network topologies than, for instance, acyclic trees. A strength is the reusability of processes among fusion nodes and between fusion nodes and resource management nodes. A strength is that the Dual Node Network has some of the useful properties of both a system architecture and a process-flow architecture.

6.10.2.4. Pattern-seeking systems

These systems are characterized by the extensive use of methods from pattern recognition, particularly template matching, graph theory and statistical classifiers. The programming style is imperative, with little or no autonomy to the functional units as in multi-agent systems nor a need for a single central data store as in blackboards.

Fountain's scheme [6.10-12] for an L1-L2 high-performance system is of this type, based on the NEAT template-matching paradigm. Clark [6.10-9] describes a cockpit system of this type, giving considerable detail to the systems software aspects as well as the process description. They define, for instance a "friendly force refinement object" in the OOP CORBA context. Svensson's IFD 03 demonstrator is based on fusion nodes producing tracks and aggregates, clusters, force identities, etc. For high level fusion functions such as force identification and aggregation, template-based behavioral methods are used.

Salerno's L2+ Fusion System [6.10-23] divides the input datastream into real-time, near-real-time and non-real-time (archival) data, each processed differently. Using graph models and graph matching, patterns are discovered, validated, and models constructed and validated. These models are used to make predictions and understand the current situation. L2+ has been most extensively developed in the global terrorism framework using global databases.

6.10.2.5 Hybrids

According to Akita [6.10-2] there are only three distinct architectures for informaton fusion: centralized data store, partitioned data store with fusion on demand, and hybrids between these two. This is too coarse a characterization we think, here we mean hybrids between and among blackboards, multi-agent systems, Pattern-seeking systems and Dual Node Networks. Many such have been proposed, notably Gad's maritime surveillance system [6.10-13] and Henrich's data fusion system for the German F124 frigate.

6.10.2.6 Other informaton fusion architectures

Lots of other things have been proposed, of course. Systems built around neural net architectures, for instance Talle [6.10-28] and Chadhuri [6.10-8]. Josephson's six generations of Abductive Machines [6.10-17] which employ Ohio State's Compositional functional modeling language CFML to make predictions and test both models and hypotheses. Carvalho's UML based general architectural framework [6.10-7] is consistent with the JDL model.

6.10.3 The recommended new hybrid architecture

All of these architectures, and many more besides, have demonstrated their usefulness in cooperative or distributed problem-solving. A fundamental question is: what is the unique nature of our informaton fusion environment within the general class of distributed problem-solving? To us it is that we are not doing general problem-solving, our system processes should be decision-directed. The goal of informaton fusion is decision support, and the architecture should be oriented towards decision-making.

From that perspective, we choose the Dual Node Network as the high-level architecture. Nodes are inherently dualistic: they *fuse* to *choose*. The same node that fuses data can immediately execute decision processes based on that (and other) information. And the same node that makes resource allocation decisions within its domain based on its local (and other) resources knows what information is needed to make those decisions effectively.

Reference integrated architecture optimized for disaster-response DF/RM systems.

Hybrid of three established data fusion architectures: blackboard, dual node network, and multi-agent system. It is defined in three heirarchical layers:
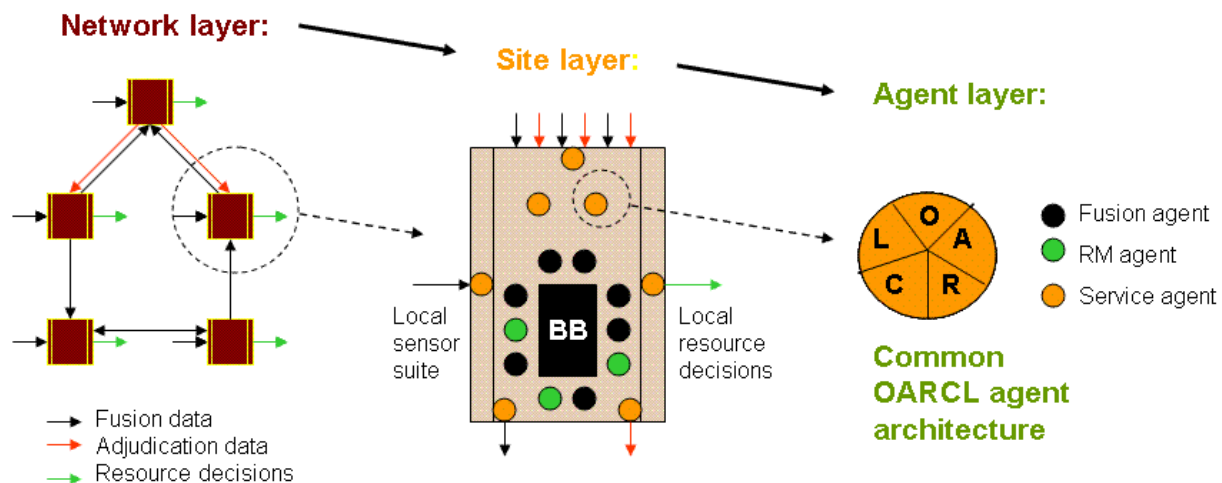
6.10.3.1 High level system architecture

So, at the highest level of architectural abstraction, a Dual Node Network is the choice, but in a form that emphasizes full topological complexity and degrees of freedom. The network of dual

AFOSR F49620-01-1-0371

nodes should be cast as a general directed graph. It may be acyclic or cyclic, it may be connected or contain disjoint subgraphs. And the full Dual Node Network should instantiate two subgraphs with the same node set but distinct links: a fusion graph and a RM/adjudication graph.

An example generalized Dual Node Network is shown below. Note that each node has its own designated local sensor suite and local resource deployment. Adjudication can only flow with fusion, adjudication without fusion makes little sense. But the link between nodes can be fusion-only.
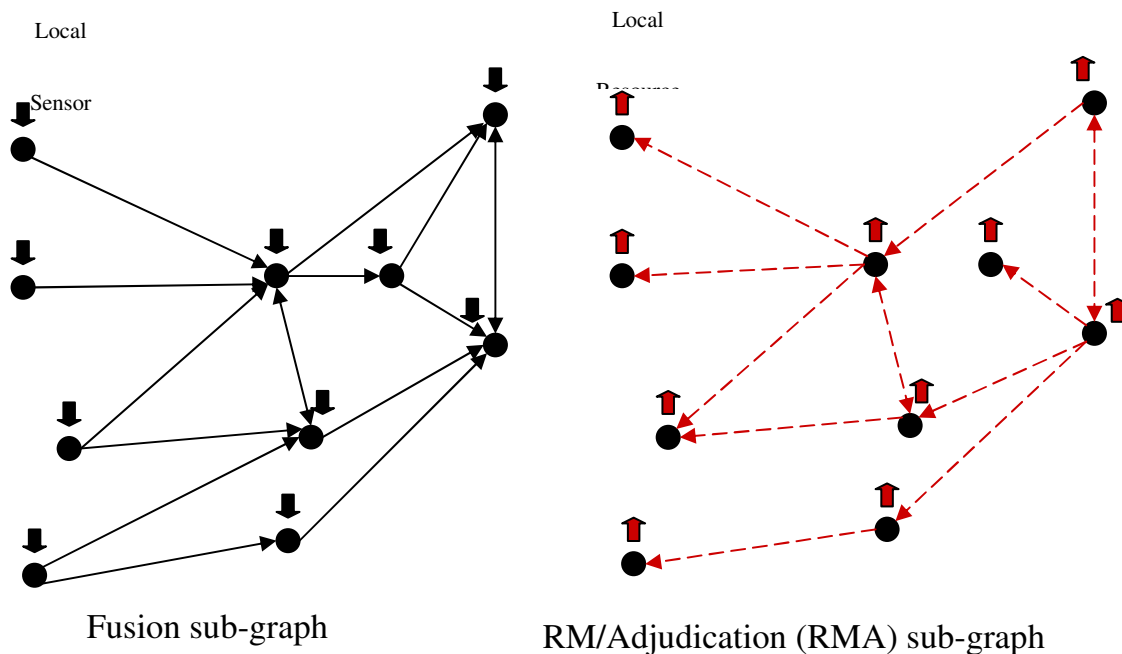
Figure 1: Generalized Dual Node Network HLA

The baseline Dual Node Network configuration is fusion fan-in, RM-fan-out, acyclic tree topology for both fusion and RMA sub-graphs. Here are some other data fusion "commmunities":

- Nearest-neighbor web

- Ad-hoc network formed on an as-available basis

- Censored adjudication links due to unreliable node behavior

- dynamic reconfiguration in the face of network damage

Different informaton fusion applications will require different topologies for fusion, adjudication and resource management. But the duality of the nodes and the dispersion of local sensors and resource deployment will be in common.

6.10.3.2 Node architecture

Within each node a great variety of processes at different levels of abstraction will be taking place. These processes must execute with a high degree of autonomy, asynchronously, and in parallel. This seems to be a natural setting for a multi-agent system.

The first design step in a multi-agent system is the specification of goals and processes the agents must execute, ie. a process flow model. Here we think the JDL model is perfectly well suited, particularly since it does not differ greatly in principle from its related OODA and Endsley PCP models. The next step is partitioning the process space into agent roles. Here again the choice we think is easy: use a human organizational model. How would (are) these same processes partitioned among human agents in organizations tasked with the same decision-oriented goals? This results in a natural decomposition, well oriented towards human decision support. Of course there may be some scaling required in associating human agency and intelligent system agency. In general, machines operate faster and more to rule, while humans operate smarter and more to the current realities.

So the inventory of agents populating each node might include:

- For each fusion level

    o A matrix of update agents, one for each entity type and Region of Responsibility (ROR). These are the agents that update the Situation Awareness Map and the impact prediction database.

    o An input reification agent

    o An input quality agent

    o An input distribution manager

- o   A product quality manager

- o   An intra-level link manager

- o   An output quality agent

- o   An output distribution manager

- o   An adjudication agent

- An inter-node link manager

- A human interface manager

- An RM decision agent

- A resource manager

As an example of the first class of agents in our earthquake scenario, one agent could be assigned responsibility for maintaining that portion of situation awareness which corresponds to the locations, properties and behaviors of all clusters of patients in the northwest quadrant of a specified jurisdiction.

In addition to the usual population of agents and speech act protocols which permit them to communicate, we would propose for the node architecture that a blackboard be instantiated, but that there be no constraint upon communication between the blackboard cognizants, ie. the IAs. That blackboard would contain the current Situation Awareness Map. This map actually constitutes several layers of entities corresponding to the L1-L4 fusion layers, each populated with their own entities and relations. The SAM is required by almost all agents for almost all purposes and should be given universal facilitated access. The blackboard controller in this case would be rather simple, amounting mainly to a scheduler and a publish-subscribe service identifying which areas of the SAM are of interest to what special interest groups of IAs. For instance a L1 casualty agent is only interested in reading and posting on L1 in his ROR, while a L3 evacuation planning agent would have much broader interests and needs to be notified when new information is posted by any L2 agent that operates in the same ROR or related L3 agents in adjacent RORs.
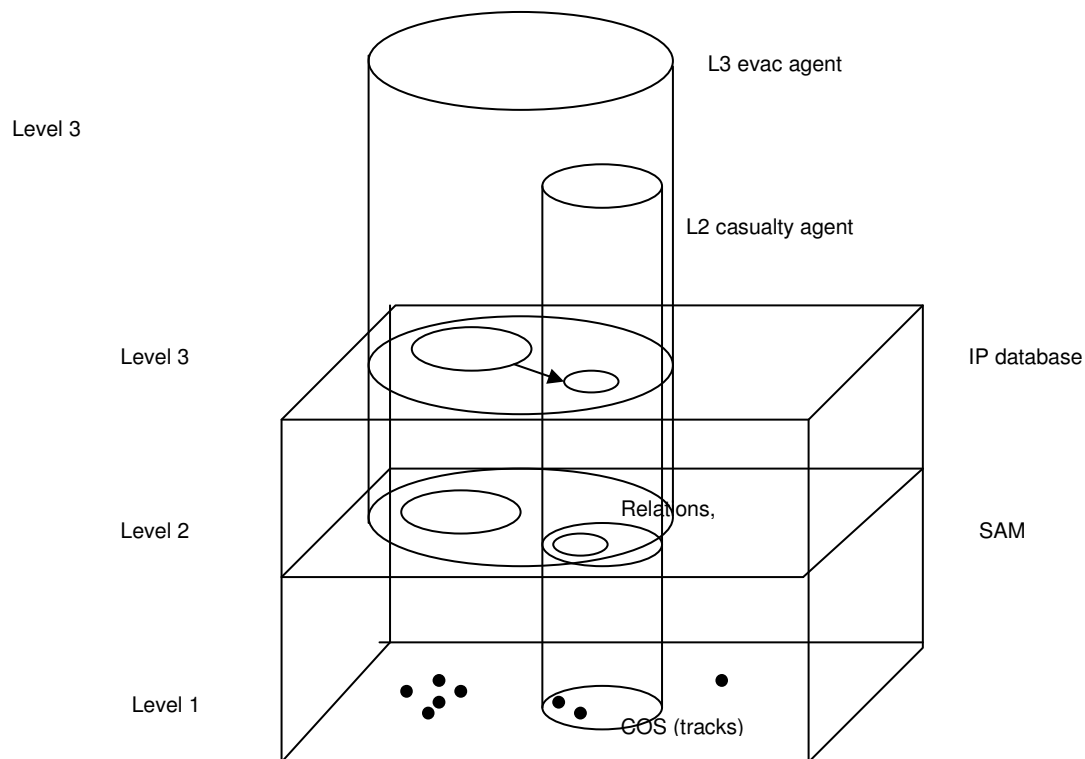
Figure 2: SAM blackboard: component of node architecture for each node

6.10.3 Agent architecture

To make explicit agents we offer five guiding principles. First, agents are entities in the problem domain as opposed to models of functional abstractions. More specifically, agents encapsulate computational units that determine plans and actions as well as the process of exhibiting acting. Second, agents are properly sized so they model entities that are rather modest in mass and time. We are not suggesting to model midgets.   Rather, we are arguing to consider acting units in such a way that the system being modeled will map to a finite number of agents in order to allow us to meaningfully focus on modeling interesting interactions and relationships. If the agent granules are fairly large we would not have the opportunity to examine intricacies of interagent interface. Since an aim in designing agent-based systems is distributed intentionality, our third principle is that agents should be considered to own their local intentionality. System intentionality should be properly divided into a series of proper sets so they can be mapped to intentionality of agent communities and the smallest units map to individual agent intentionality. Our fourth agent design principle is coherent dissemination of information, knowledge, and wisdom. Agents should be provided with methods for caching and sharing results of individually

(i.e., locally) processed and fused data. Properly designed dissemination methods will offer cohesion so that the set of agents will act as a whole, i.e., a hive-mind. Although it is beyond the scope of this article, we need to point out the need for shared or disparate ontologies among groups of agents (NCOR). A more elaborated consideration needs to account for delineate ontology mediation by agents themselves or designated agents. Our fifth principle suggests to consider agents operating in sufficient independence as if they operate in parallel. Despite the obvious need for interdepedendence among agents in any complex system and hence models of relationships and interactions suggested herein, agents need to be designed to operate concurrently as opposed to sequentially. The large number of agent architectures in existence does not imply maturity in the discipline. Instead, it reflects a rush to capture and document features of interest. In the following section we will review agent architectures and propose a need for parsimony and a return to original conceptions of modeling agents.

6.10.3.1 Review of Agent Architectures

One of the earliest and the most influential agent architectures is the BDI paradigm. The Stanford group of researchers in mid-1980s suggested capturing mentalistic notions such as belief, desire, intention [6.10-19]. There was a heavy leaning toward grounding BDI in formal modal logics partly to inherit the properties of soundness and completeness and partly to gain expressive power of treating BDI as modalities. The expressive power gained came at the expense of lack of tractability. Along with many researchers we have implemented a limited form of BDI in our labs with partial satisfaction. The best known implementation is often attributed to Kinney and Georgeff, 1991). BDI shortcomings are well-documented and we will avoid repeating them here. Instead, we point to the need to preserve Bratman's claim that rational agents strive to adopt and maintain conflict-free intentions. All rational agent reasoning will service for avoid detraction from adopted intentions and on methods for manifesting desired objects of intention. We wish to explicate the primacy of *intention* with the need for *attention* as we will see in the following section. A more pragmatic agen t architecture is MaSE [8-11-21]. Rooted in BDI, DeLoach has not only provided expressive power of modeling roles and communication in MaSE but also provided a blow by blow methodology that was lacking in BDI. Tropos is a recent agent architecture that offers both a methodology and rich expressivity [6.10-24].

Human Analyst

Observe ⟹ Attend ⟹ Reason ⟹ Communicate ⟹ Learn
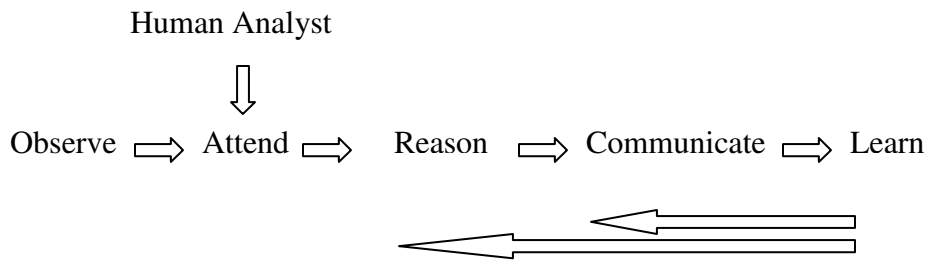
Figure 3. OARCL Components

6.10.3.2 OARCL Reference Architecture

OARCL is comprised of Observe, Attend, Reason, Communicate, and Learn (OARCL) components. Similar to the OODA loop concept proposed by Col. John Boyd [6.10-31], it is complex and involves many internal loops and nontrivial processes that supervise and keeps the system inline with various measures of effectiveness and performance. There are abundant asynchronies and nonlinearities (see Figure 3). One of our aims in introducing OARCL is to urge a return to the original conceptions of agents, which were anthropomorphic modeling of a sense of acting. Another aim is to put in the spotlight the salient properties of agency by OARCL components. At a metaphorical level, an OARCL agent is a cognitive entity with functionalities suggested by pre-attentive functions captured in the module Observe, attention generation and maintenance in the module Attend, inferential capabilities in the component Reason, intentional message generation in the module Communicate, and abilities to modify perception, attention, and reasoning processes in the component we call Learn. Next we provide general description for each component.

In order to coarsely filter out irrelevant data, the Observe component performs agent input data distribution management functions. Agent who live in a highly distributed environments need to control the volume and the nature of input they process. We choose to consider the tasks of what kind and how much data to process as a pre-attentive function of an agent. The best example is human visual processing where visual input is rapidly and automatically filtered. Supported by recent findings in human visual processing that suggests *attention* plays an important cueing role even in early, *preattentive* visual stage we have selected that as our next component [6.10-25] .

Attend receives references from the human supervisor, all agent requests from the Observe component as well as from the Communicate module. References might be associated with specific sensory-identified objects or targets or it might be in terms of features, patterns, and conditions. The selection process, which includes nontrivial reasoning is modeled in this component. We will deliberately leave out discussion of the obvious connection between attention to awareness and consciousness [6.10-20]. Despite this omission, we point out that this connection is the most elusive, intriguing, yet essential character of agency. There is an inextricable relationship between defining characteristics of independence and pro-activity of computational agents and self-awareness encapsulated in attention. The notion of individuality in attention plays a crucial role in guiding as well as controlling inference and logical reasoning. The rationality principles of Allen Newell and Nick Jennings are addressed in our reason module [6.10-18]. The *reason* component performs the primary inference in OARCL agents.

Communicate performs external information management including the speech act using standard agent communications language (ACL) for outputs (e.g., requests) and input information and requests from other agents.

Finally, the Learn component generates performance assessment that drives its process management of all agent components. Next, we will discuss applications that lead to development of our reference architecture.

One of the problems that motivated OARCL was the SIGINT man on the loop with the aim to design an agent-based software that aids and automates intelligence analysis, technically known as SIGINT analysis [6.10-23]. The SIGINT activity encompasses all of command, control, communications, human intelligence, surveillance, and reconnaissance. The second motivating problem was modern disaster response with the aim to design an agent-based software that fuses information at varying levels of abstraction in order to rapidly assess situations at a high level. Our approach preserved the highly distributed and disparate loci of information gathering and synthesis.

Finally, we review OARCL modules and briefly discuss the software engineering tasks therein. For Observe one must broadly gather techniques for selecting data sources and channels with details beyond the scope and interest of this report. Attend for both problems must be flexible to allow for changes in the human analyst's goals and targets. Generically, a human analyst will set and revise conditions and targets of interest for the remainder of the system (shown in Figure 3). The reference points selected in attend will be used to place filters on data gathered by Observe. Reason will need to identify threats and opportunities of interest n the command and control, which is the core function of a typical human analyst. Communicate will consist of (a) all conditions for triggering messages to other system functions in support of SIGINT as well as disaster response, and (b) types and formats of messages corresponding to triggering conditions. Learn will embody metrics and sets of adjustments for internal functions as well as interactions among OARCL modules.

6.11 Testing and Evaluation

To explore the performance of the L0/L1 fusion scheme described in Section 6.8 and the higher level fusion in Section 6.9, a set of Monte Carlo runs of the simulation environment DIRE were performed. The test parameters, results and discussion of these tests are contained in this section.

The platform for all DIRE tests described here consisted of a network of multiple Pentium 3 and 4 Windows machines variously running NT, Windows 2000 and Windows XP, all located in the CMIF Lab at the University of Buffalo. Clock rates of the Pentium 3 cpu's is 700 MHz – 1 GHz, the single Pentium 4 is rated 3 GHz. The simulation database was located on the University of Buffalo's Fluids server two floors beneath the CMIF Lab in Bell Hall and connected via high-speed ethernet cable. Typically six machines would run simultaneously in each federation run, exchanging messages and maintaining consistency via the Run Time Interface protocol as described in Section 6.2.

For the first set of DIRE runs which employed the final forms of all the federates, the simulation's temporal ratio, defined as the ratio of simulated (logical) time to wall-clock execution time, was approximately 0.05. In other words, each hour of simulated time in the DIRE environment required approximately 20 hours of run time on these machines to produce it. In order to improve the temporal ratio, sections of the code were rewritten to permit maximum

AFOSR F49620-01-1-0371

pre-compilation and pre-execution of that code which could be shared among multiple machines and multiple runs without changing the logic or statistical design of the runs. For example, a set of templates of the spatio-temporal evolution of plume material concentration were pre-computed for several wind speeds and source release temporal profiles using canonical wind direction and source strength. In a given DIRE run containing a secondary Hazmat incident, the appropriate template sets were laid down on the Hazmat source location which was randomly selected for that run, rotated according to the initial wind direction specified in the ini file, and fit to the spatial grid using bilinear interpolation. These concentrations were then scaled by the randomly selected source strength initialization parameter for that run. The result of several code modifications of this type was improvement of the temporal ratio to about 0.25, ie. fifteen minutes of logical time per hour of wall-clock execution time.

This final temporal ratio was still considerably lower than planned, substantially lower than that necessary to complete the full planned set of T&E simulation runs, following the performance evaluation design presription of Rawat et al [6.11-1], within the time period which remained available to the project after these code improvements. There are two principal causes for this circumstance. Early in the project, choices were made for the software environment (languages and compilers, data base management software, graphical information system) in which to produce the ground truth, generate the reports, and drive the dynamic objects such as ambulances and walk-in casualties forward. The resulting code executed less efficiently than anticipated. Even after spending considerable time to optimize this code, the choices, in particular the initial selection and use of Visual Basic 6, limited the execution efficiency more than anticipated. Rewriting the code using a language producing faster compiled code was considered, but there was not time within the scope of this project for the complete re-write of thousands of lines of code and re-testing this would requre. The second reason was the fact that the final versions of all federates were not available for integration into DIRE and acceptance testing until later than planned, leaving insufficient time after "code lock-down" for the full planned suite of testing and evaluation runs.

The tests completed and the evaluation of those tests, while less than a full suite, do permit observations to be made concerning the appropriateness of the L0/L1 and L2/L3 fusion schemes

advocated here, and their performance in the DIRE emergency response environment. In addition, they suggest ways in which further work may be of benefit.

6.11.1 Test suite

1. Base tests

Test No.	L0/L1	L2/L3	Plume start time	Pr(CivFlsRept)
01_001_01	Y	Y	1500	0.20
01_001_02	Y	Y	1500	0.20
01_001_03	Y	Y	1500	0.20
01_002_01	Y	N	1500	0.20
01_002_02	Y	N	1500	0.20
01_002_03	Y	N	1500	0.20
01_003_01	Y	N	1500	0.20
01_003_02	Y	N	1500	0.20
01_003_03	Y	N	1500	0.20
01_004_01	Y	Y	600	0.20
01_004_02	Y	Y	600	0.20
01_004_03	Y	Y	600	0.20
01_005_01	N	N	600	0.20
01_005_02	N	N	600	0.20
01_005_03	N	N	600	0.20
01_006_01	N	N	600	0.20
01_006_02	N	N	600	0.20
01_006_03	N	N	6000	0.20

2. Full Fusion False Report Sensitivity Tests

Test No.	L0/L1	L2/L3	Plume start time	Pr(CivFlsRept)
02_004_01	Y	Y	600	0.00

02_004_02	Y	Y	600	0.10
02_004_03	Y	Y	600	0.20
02_004_04	Y	N	600	0.30
02_004_05	Y	N	600	0.40
02_004_06	Y	N	600	0.50

3. Fusion Level-False Report Interactions Tests

Test No.	L0/L1	L2/L3	Plume start time	Pr(CivFlsRept)
03_001_01	Y	Y	1500	0.05
03_001_02	Y	Y	1500	0.05
03_001_03	Y	Y	1500	0.05
03_002_01	Y	N	1500	0.05
03_002_02	Y	N	1500	0.05
03_002_03	Y	N	1500	0.05
03_003_01	Y	N	1500	0.05
03_003_02	Y	N	1500	0.05
03_003_03	Y	N	1500	0.05
03_004_01	Y	Y	600	0.05
03_004_02	Y	Y	600	0.05
03_004_03	Y	Y	600	0.05
03_005_01	N	N	600	0.05
03_005_02	N	N	600	0.05
03_005_03	N	N	600	0.05
03_006_01	N	N	600	0.05
03_006_02	N	N	600	0.05
03_006_03	N	N	6000	0.05

AFOSR F49620-01-1-0371

All tests were run for logical times exceeding one hour. Some were run for multiple hours, but in order to establish a common basis, all results shown here are for the first hour following the initial earthquake event (logical time 3600 seconds).

6.11.2 Report profiles

Immediately following the initial earthquake event, casualties are laid down according to the Hazus statistical estimates for casualty counts in each census tract given the selected geo-spatial earthquake parameters. Within DIRE, reports begin to flow. Civilians phone in reports of casualties down. Police and ambulance drivers radio in reports. These reports include varying amounts of information, which may include the victim's reported location, estimated severity of injury, age, sex and race, and in some cases name identification. These report attributes are related to the ground truth by a confusion matrix as described in Section 6.2, and also by a false report probability. By a false report we mean a report not based on observation of a ground truth casualty. This is distinguished from a confused true report. The attributes reported in all true reports, those based on observation of a ground truth casualty, are subject to confusion matrices reflecting the limited ability of reporters to accurately assess what they see. This is particularly a problem for data fusion in the emergency response scenario, in which stress increases the probability of error for all reporters, whether civilian, police, ambulance EMT personnel, or other emergency responders.

6.11.2.1 Casualty report profiles

Figure 1 shows the accumulated number of casualty reports received from all sources over time. A Monte Carlo average of 9 runs for the Base Case 01_001_01 – 01_003_03 was used.
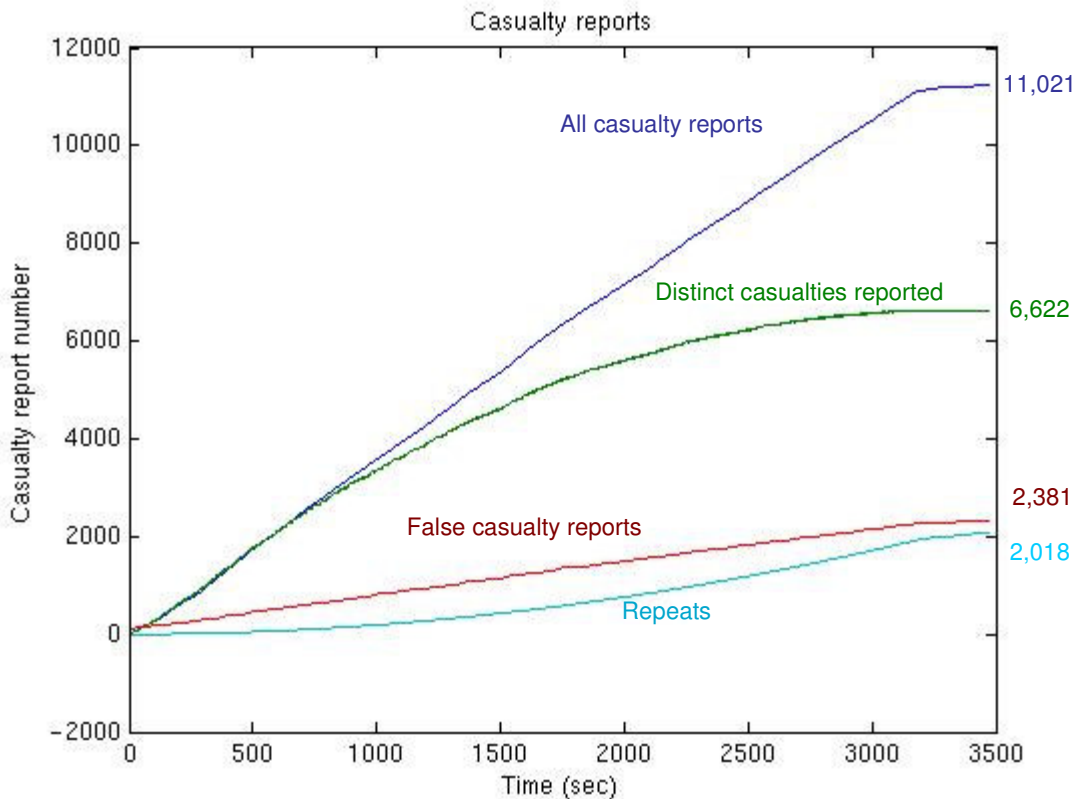
Casualty reports

Over the first hour following the the earthquake event, there is an average of 11,021 casualty reports received. The sources of these reports are 10,614 civilian reports (mostly phoned reports) and 407 emergency responder reports (mostly radio). All these reports are confused, and some are false. The false reports amounted to 2,381, almost entirely from civilian calls. Of the average number of total casualties laid down in ground truth, which was 16,766 distributed throughout the earthquake zone according to Hazus statistics, 6,622 were reported in one or more report. Thus in the first hour following the earthquake, some information was available on 39.5% of the casualties. In the case of many casualties, more than one report was received. The number of repeats is 2,018, ie. there were on average 2,018 reports on casualties who had already been reported. Since all information in these reports is confused, this presented both an opportunity and a challenge to L1 fusion to determine which reports to associate into unified tracks, and with what attributes such as location and severity. If each report is treated as a separate track, resource decisions would not be made correctly.

The constant rate of casualty reports (for all but the last few minutes) shown in Figure 1 reflects the modeled capacity of the phone system. It was saturated with calls for most of this first hour,

thus there were delays in many reports. Towards the end of this first hour, on average at the 51 minute mark, the telephone system was no longer overused and the backlog of calls was actively reduced. The rate of first report of a new casualty dropped steadily over the first hour, as an increasing fraction of the reports received were repeats.

6.11.2.2 Hospital arrival report profiles

Among the 54 distinct report types listed in Section 6.3 which were being exchanged within the federation during this first hour, perhaps the most important are the casualty reports summarized above. The casualty reports drive the low level fusion casualty track creation and maintenance process, which is a key input to the higher level fusion dynamic aggregation, situation assessment and impact assessment processes. Perhaps the second most important class of reports are the hospital arrival reports. The core goal for early-phase emergency response operations is the saving of lives. The principal life-saveing tool modeled in DIRE is the intelligent dispatch and routing of ambulances to casualties with a critical need for hospital services, and their subsequent dispatch and routing to the most appropriate hospital. Along with dispatch decisions, the determination of the most effective routes to pickups and thence to hospitals for the ambulance to take is important. An efficient hospital delivery system requires these primary factors:

1. Accurate L0/L1 fusion specification of casualty tracks;

2. Accurate L0/L1 estimation of casualty track attribute uncertainties

3. Accurate L2/L3 assessment of dynamic casualty clusters

4. Accurate L2 assessment of dynamic casualty cluster attribute uncertainties

5. Effective dispatch algorithm based on clusters, tracks and dynamic hospital capacitites

6. Effective routing algorithm based on road damage reports and dynamic traffic conditions

Thus the hospital delivery metric is a useful indicator of the accuracy and effectiveness of the combined fusion elements operating in a given test run.

Figure 2 shows the arrival time of each casualty of severity 1 and severity 2. There are 20 local hospitals included in the DIRE data base, this graph represents the Monte Carlo average sum of all arrivals at all hospitals.
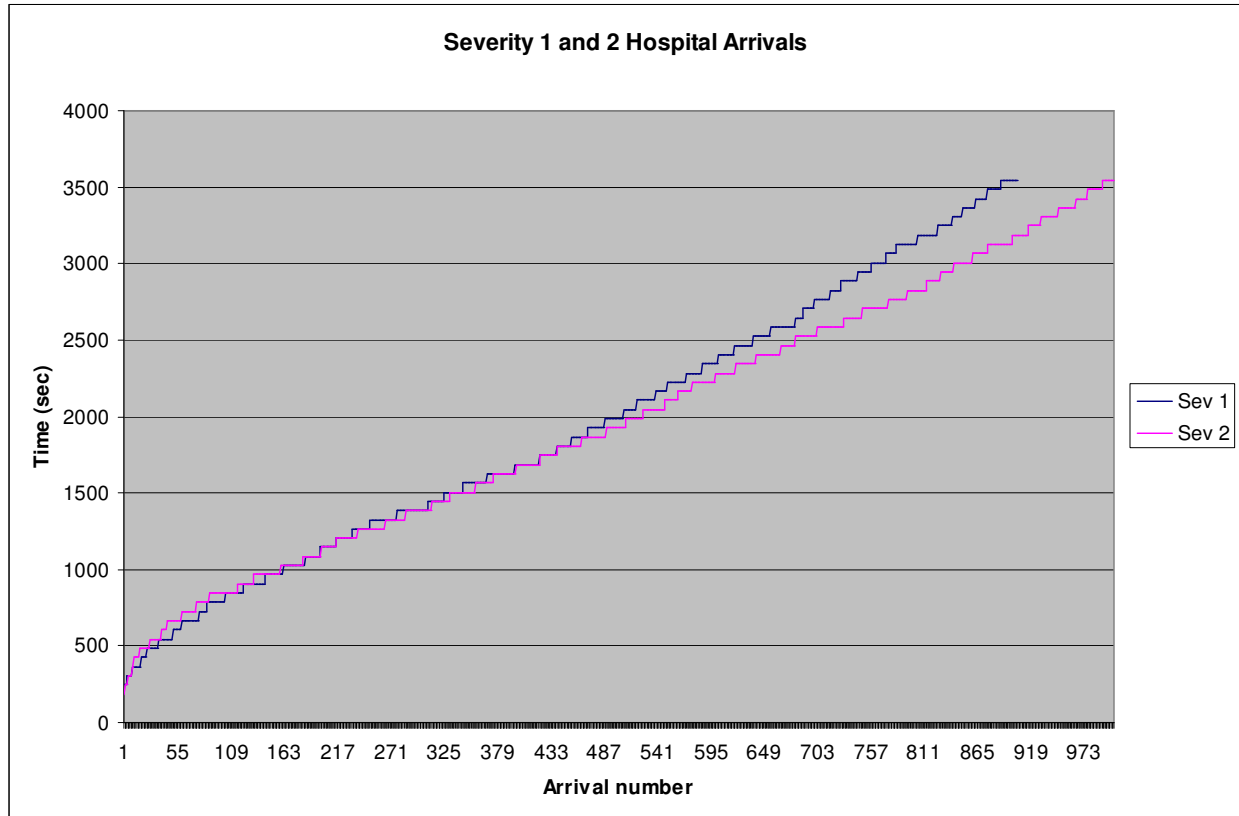
Figure 2: Arrival time of all severity 1&2 hospital arrivals

This Figure shows a relatively constant arrival of casualties of both severities during this first hour, with an average of 878 of severity 1 received in that hour, and 1002 of severity 2. Severity 1 casualties are characterized as not actually requiring hospital treatment. Severity 2 casualties require hospital care, but not surgery or bottleneck lab tests such as blood work or x-rays which limit hospital service capacity. Examples of severity 1 include bruising and pain, severity 2 would include stitches or severe psychological complaints. Since the dispatch decision rules employing fusion require that neither severity 1 nor severity 2 patients be ambulanced during this critical intial phase, the time profile of hospital arrival reports for these low-severity casualties is largely independent of the type or level of fusion done, or whether fusion is done at all. The sensitivity to data fusion is contained in the dynamics of the more severely injured casualties.

Figure 3 shows the first-hour hospital arrival time of each casualty of severity 3 under three sets of Monte Carlo averages: runs in which no data fusion is used, where only the low-level data fusion scheme of Section 6.8 is used, and where all levels of fusion as detailed in Sections 6.8 and 6.9 are employed.
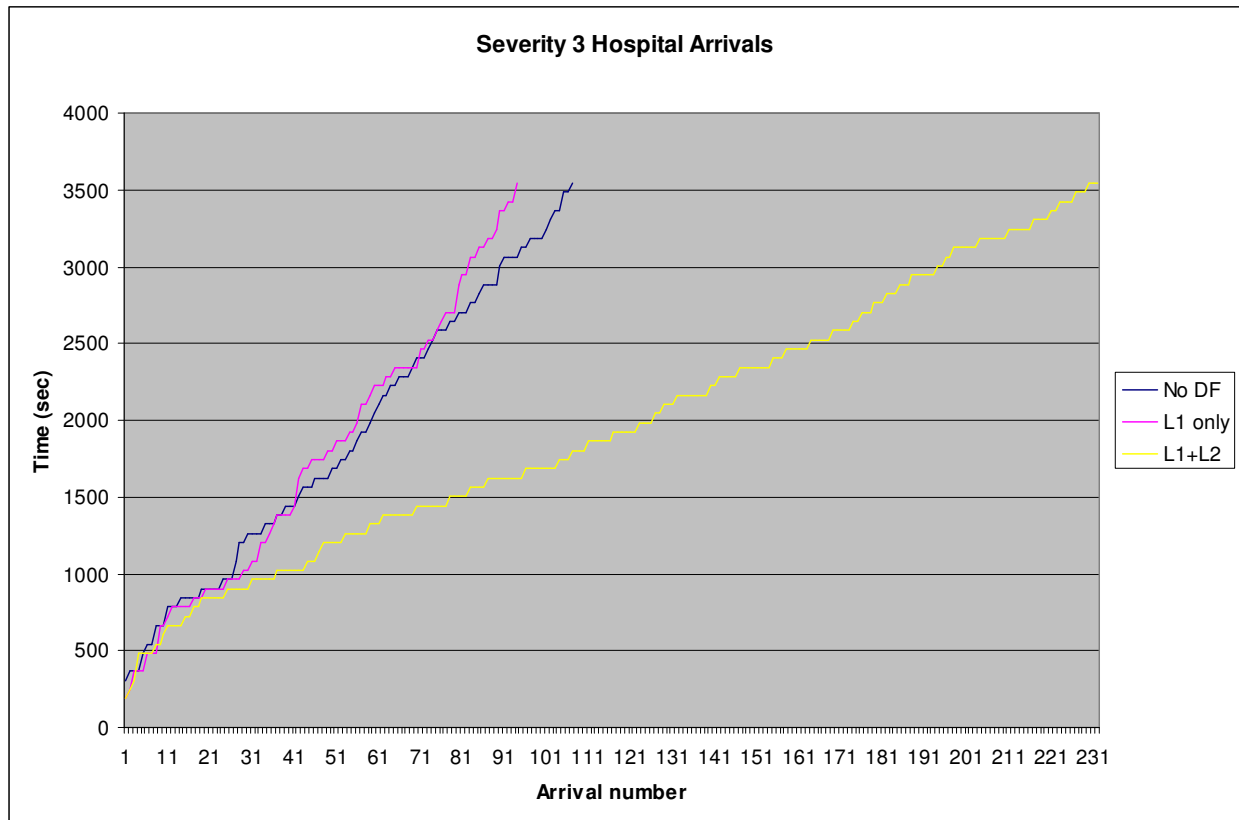
Figure 3: arrival time of all severity 3 hospital arrivals

When no fusion is used, then as they come free, ambulances are dispatched to the nearest reported casualty 3 location and this action repeated until they are full or they time-since the initial pickup, thence to the nearest hospital with anticipated residual capacity to service those casualties at the estimated time of arrival. With lower level fusion only, tracks are used instead of reports. With full fusion, account is taken of the clustering of patients and the anticipated elementary hospital situation, elementary road situation, and their relationships. As described in Section 6.5, ambulances are dispatched to the rim of a near-by cluster containing multiple casualty 3's, if such is available. When the first casualty 3 is found and picked up, the ambulance searches locally for other casualty 3's to capacity. If this is not discovered in a limited period of

AFOSR F49620-01-1-0371

time, the crew radios for further instructions. In any case, when departing, they are given either a hospital target and a route to that hospital by the dispatcher, or a second nearby cluster rim location to search.

Figure 3 shows the dependence of the number of severely injured hospital arrivals on the level of implementation of fusion in the system guiding dispatch and routing. In the case of no fusion, on average 106 such patients are treated in the first hour. Where only low level fusion is implemented, the average number is 95. Thus, with respect to this imporant measure of effectiveness, using only low-level fusion (of the type recommended) is worse than using no fusion at all. Examination of the log files for these runs shows that the track association accuracy may be inadequate in this highly uncertain environment, with many false reports, universal report attribute confusion, and the movement of casualties from their reported locations. Ambulances are frequently dispatched to locations where in ground truth there are no severity 3 casualties to pick up, nor any to be discovered nearby. Thus the efficiency of that ambulance's service is compromised in two ways: increase in the average time from dispatch to arrival at the hospital, and decrease in the average number of casualties on-board upon arrival.

The situation is quite different for the full-fusion trace shown in Figure 3. On average 232 severity 3 patients are delivered to the hospital when both low and high level fusion are implemented. This is more than double the number of severely injured patients served by the system during this first hour. It is difficult to estimate the number of lives that might be saved among the 126 severely injured casualties that arrive in the first hour when fusion is employed, as opposed to arriving at some undetermined later time when it is not, but it suggests that the saving of lives in such a scenario as the Northridge 1994 Earthquake by the used of the form of high level fusion recommended here would be significant.

6.11.3 Other metrics

Here we include calculations of other measures relevant to system evaluation which were derived from the test data noted in Section 6.11.1. As detailed by Blasch [6.11-2] there are five classes of metrics: confidence, accuracy, timliness, throughput, and cost which should each be represented in a comprehensive test and evaluation scheme. While cost has not been modeled in DIRE and thus cannot be measured here, we present results in each of the remaining categories.

6.11.3.1 Measures of Performance

MOP	Metric	No fusion	L0/1	L0/3
Positional uncertainty	Max Cov matrix Eval	1.000	1.072	1.072
Cas Sever uncertainty	P(3|<3)	1.000	1.153	1.153
Identification Accuracy	Pc	1.000	1.033	1.033
Dimensional reduction	Trace/Rept (Clust/Trace)	1.000 (1.000)	1.414 (NA)	1.414 (133.7)
Report Ass'n accuracy	Pc	NA	0.61	0.61
Cluster detection	PD (PFA)	1.000 (1.000)	0.866 (1.012)	3.811 (2.665)

Table 1. Dimensionless Performance Gain for three levels of data fusion

The mechanics of most of these calculations are self-explanatory. The Positional Uncertainty Gain is determined, for instance, by the ratio of the average of the maximum x-ycovariance matrix eigenvalue of a casualty track to average of the maximum covariance matrix eigenvalue of a casualty report. When reports are fused, positional uncertainty is reduced as errors are reduced by averaging. The identification accuracy is only increased by 3.3%, reflecting the fact that most of the casualties are not reported with positive ID's and thus correct data association in most cases does not help identification. The large Cluster/Trace dimensional reduction of 276.9 with the use of high level fusion is explained by the fact that on average, over time and Monte Carlo runs, there are only 25.13 clusters but 3361 traces. The report association accuracy is relatively low (61%) because the threshold of association confidence required to declare an association has been set high. In the emergency response setting there is high cost in underestimating the number of distinct severe casualties in a given area, since it may result in delays in supplying the transport needed to get them to the hospital. It was felt that it was better

to err on the side of caution, starting a new track where there was reasonable doubt whether the given report was a repeated report on an existing track. For cluster detection, the comparisons were between the shrink algorithm operating on reports, the shrink algorithm operating on tracks, and the full L2 cluster detection scheme, which uses topological relationships and dynamics.

6.11.3.2 Measures of Effectiveness

MOE	Metric	No fusion	L0/1	L0/3
Enrollment latency	Time	1.000	1.043	1.091
Ambulance util factor	Occupancy ratio	1.000	1.161	1.142
Hazmat detection	PD (PFA)	NA	NA	1.000 (0.133)
HAZ detec latency	600/Time (sec)	NA	NA	0.699
HAZ cluster detection	PD, PFA	NA	NA	0.803 (0.221)

Table 2. Dimensionless Effectiveness Gain for three levels of data fusion

Enrollment latency is defined as the duration of time between that when a given ambulance patient arrives at the hospital and the (earlier) receipt of the first report on that patient. In cases in which a report had never been received for that casualty, he/she was not counted. The average occupancy of the ambulances upon arrival at the hospital reduced slightly with the addition of higher level fusion (Effectiveness Gain of 1.142 vs. 1.161). This is an unexpected result and at this point we can offer no clear cause. Perhaps the manner in which the ambulance searches its locale after finding the first casualty in a cluster needs tuning.

Secondary Hazmat incidents were always detected, with a false-alarm rate of 13.3%. The average latency between the initial release of hazardous material and detection of the hazmat incident was 858.1 seconds, about 14 minutes. The maximum acceptable latency was taken to be 10 minutes, thus the Effectiveness Gain was less than unity, in fact 0,699. What was not modeled in DIRE were direct reports on Hazmat, such as sighting a burst storage tank or smelling a burning odor in the air. To test our methods, we assumed a worst-case secondary Hazmat scenario, in which the toxic material was colorless and odorless, and the rupture of the storage vessel was unreported for the first hour (eg. a tank in a tank field ruptures with no tank inspections for a period of time after the earthquake). The clues that the abductive reasoning system could use to declare a Hazmat incident were, roughly, an overabundance of respiratory injuries reported in one or more clusters, and those clusters growing downwind as described in detail in Section 6.9.

6.11.4 Hazmat plume propogation zonal estimates

The graphs below demonstrate the manner in which the belief-based argumentation system forming the core of the high-level fusion module operates to detect a Hazmat incident and determine the geographic zone which should be immediately evacuated, and that zone which should be warned to prepare to evacuate.

A stand-along simulation distinct from DIRE was implemented in which there were four stationary Gaussian sources of casualty reports. Each had difference locations and covariance matrices, but the same fraction of respiratory injuries, 0.10. Then a fifth source was started, modeled as a Gaussian plume generating additional casualties, all of which were respiratory. Figures 4(a) depicts the casualties at the time the Hazmat incident began, 4(b) the first detection, 4(c) the estimated state some minutes later and 4(d) next status an equal interval later.
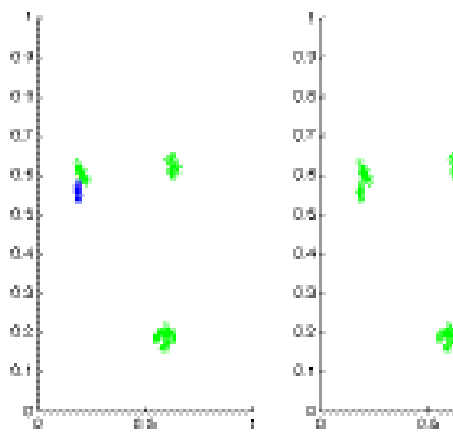
Figure 4(a) Before detection   Figure 4(b) At detection
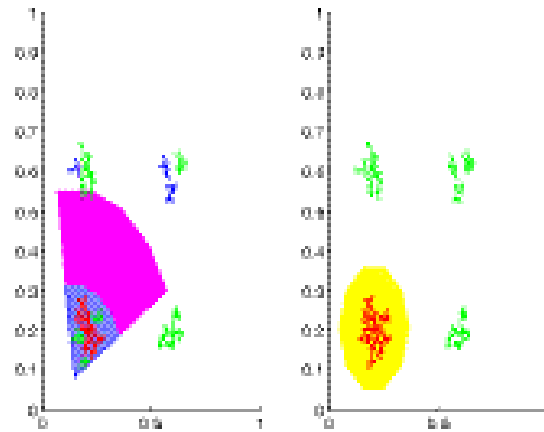
In each Figure, the left graph represents the L2/3 estimates of the situation and the impact of that situation, while the right graph represents the ground truth. The red dots are respiratory casualties and yellow ellipse the ground truth respiratory cluster. The blue region indicates estimates of the geographic zone already above toxic dosage, and the magenta region the area likely to sustain toxic dosages over the next hour. In this simulation the wind was blowing from the southwest (200° and the site of the Hazmat spill was at (0.1,0.1). Blue dots are newly discovered clusters, which are under consideration as evidence for or against the Hazmat hypothesis, and which are governing the approximation of the actual source location and sector of toxic threat for possible evacuation order.
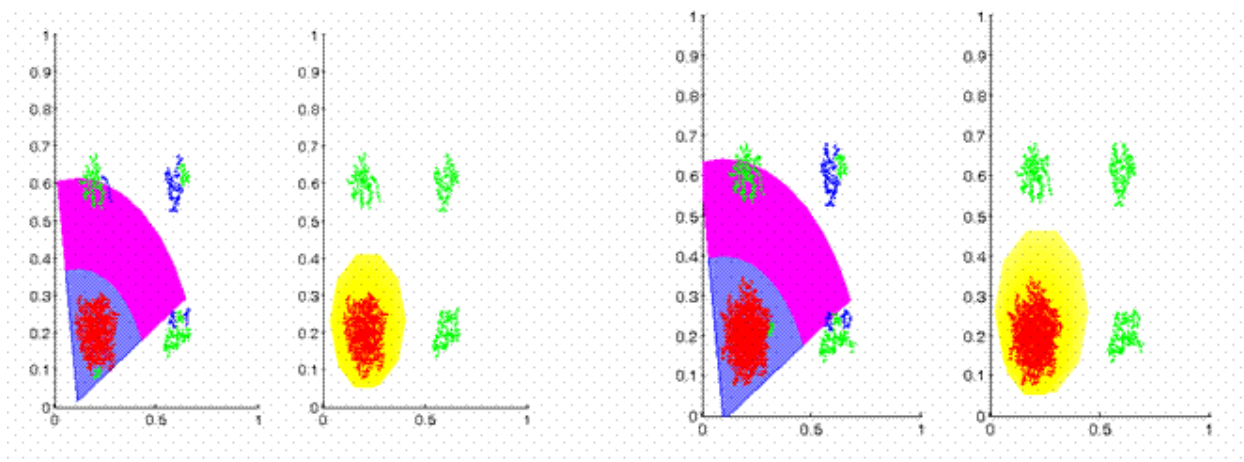
Figure 5(a) After detection   Figure 5(b) Well after detection

Note that the existence of the Hazmat secondary casualty cluster is confirmed by the growth of the red cluster upwind, and that none of the other clusters are misidentified as Hazmant clusters.

6.12 Track Confidence and Adjudication

In this section design strategies for the support of online update of track confidence estimates and adjudication management are presented within the framework of the Dual Node Network architecture [6.12-1]. Both track confidence and adjudication processes are necessary to assess and maintain the reliability of fused reports for the distributed lower-level and belief-based higher level fusion methods presented in Sections 6.8 and 6.9 of this report.

On-line track confidence estimation is the real-time process for propagation and updating the probability that each output track is false and the probability that the detectable entities in each coverage area are represented in the fused picture (i.e., a tracker "Receiver Operating Curve (ROC)). On-line track confidence estimates enable a rigorous basis for scoring track initiation, propagation, and deletion versus input data association. On-line track confidence estimation enables the users of fused track files to better combine these tracks with other information. The payoff is the trustworthiness of each fused track and completeness of the fused picture being recursively maintained. The role for track confidence estimation in distributed fusion is described by applying the Dual Node Network (Dual Node Network) Architecture to the distributed fusion problem. No fusion systems have been developed with on line track confidence estimation capability. Some fusion systems use track birth & death statistics and track association confidences in place of this capability. The problem is how to propagate & estimate these confidences on line based upon the current sensor ROC curve statistics and data fusion decisions.

The Baseline approach described herein is an approximate solution that includes the following:

1. uses Pd and Pfa a priori information from each source to initiate track Pd and Pfa

2. propagates track Pd and Pfa for each source update

3. generates track Pd and Pfa to support track association

4. updates track Pd and Pfa based upon source ROC curve and association confidences

Augmentations to this approach considered include use of a priori track birth and death Poisson statistics and use of data mining of fusion experimental results.

Alternative approaches considered include:

1. Ad Hoc: table lookup of performance based upon exhaustive testing and all foreseeable operational conditions (e.g., sources fused, environment, etc.)

2. Possibilistic: evidential or fuzzy knowledge combination to treat the uncertainty in the-uncertainty

3. Logic/Symbolic: rules or scripts to yield track confidences based upon the each operational situation

4. Neural Networks: nonlinear pattern recognition of approximate track confidences based upon simulations and exhaustive testing

Specific Bayesian equations for CTP track confidence have been derived and will be presented.

The objective of Adjudication Management (AM) is to maintain consistency of the call for service data bases across distributed jurisdictions. The role for adjudication in Disaster Assessment (DA) and the baseline approach are described. The tasks to achieve this capability are then explained. The baseline approach for distributed AM is hierarchical where the EOC local commander reconciles all the incoming track related information to improve upon his subordinate jurisdiction's Consistent Tactical Picture (CTP). AM evaluates and selects significant changes in each jurisdiction call for service data base for adjudication across supporting the jurisdictions. Each jurisdiction fusion network creates 5 level 1 fusion call for service data bases corresponding to casualties, emergency vehicle location (i.e., police and ambulance), facility (i.e., hospital and emergency facilities), transportation link delays, and bridge damage. AM resolves conflicts among these inputs and generates directives to the jurisdictions to maintain the CTP. AM also determines the advisements necessary to be sent from each jurisdiction to the EOC commander to keep him informed of relevant and significant new information to him. This prioritization and culling of the real-time CTP changes to determine commander and subordinate-specific call for service track set modifications to enable bandwidth

efficient updating of the relevant CTP's to support all levels of decision making. The commander adjudicates these inputs to maintain the jurisdiction CTP track set so as to support higher levels of fusion and coordinated operations.

6.12.1 Track Confidence Estimation

On-line track confidence estimation is the real-time process for propagation and updating the probability that each output track is false and the probability that the detectable entities in each coverage area are represented in the fused picture (i.e., a tracker "ROC" curve).

On-line track confidence estimates enable a rigorous basis for scoring track initiation, propagation, and deletion versus input data association.

On-line track confidence estimation enables the users of fused track files to better combine these tracks with other information. The payoff is the trustworthiness of each fused track and completeness of the fused picture being recursively maintained.

The reliability in each fused track needs to be recursively maintained to support decision making, so track confidence estimation provides the probability that each fused track is false in real time. In addition the probability that the fused picture will contain a track on each detectable entity in each coverage area is updated after each batch of input data is fused. As a result track confidence estimation provides a rigorous basis to updating the completeness of the fused picture for track maintenance and resource management decisions. In general these track confidences vary with each track's location, report association history, entity type, etc.

No operational distributed fusion systems have been developed with a rigorous on-line track confidence estimation capability although some fusion systems use track birth & death statistics and track association confidences in place of this capability. The problem is how to propagate & estimate these confidences on-line based upon the current sensor roc curve statistics and data fusion decisions. The baseline approach defined herein is an approximate solution that includes the following:

1. uses Pd and Pfa a priori information from each source to initiate track Pd and Pft

2. propagates track Pd and Pft for each source update

3. generates track Pd and Pft to support track association

4. updates track Pd and Pft based upon source ROC curve and association confidences

The track confidence estimation equations are given for each updated, propagated, and pop-up initiated track. These equations provide track file confidences for distributed users much like those needed when using data directly from sensors. As a result sources with lower confidence reporting thresholds (e.g., providing more timely kinematics, ID, and their own probability of detection and false alarm statistics) can now be rigorously associated [or not] with distributed site track files, maintaining their own coverage and individual track confidence statistics. Such lower threshold source reporting is needed, so as to provide on-line situation awareness to the commander for a clear and actionable picture of his area of interest. The resulting data fusion product also drives retrospective analysis and collection management to fill information gaps in real-time, so as to continuously improve upon the situation estimate. The progress in track confidence estimation for fusion extends to support solutions to the fusion performance evaluation (PE) problem, since PE is another fusion problem. Namely, the development of PE software entails designing a PE network of fusion nodes that solve the track-to-truth data preparation and association as well as measures of performance (MOP) estimation problems. Track confidence estimation is applicable to achieve more accurate track-to-truth association and MOP estimation. In summary, just as the sensor ROC curve (i.e., the measurement Pd & Pfa) is required to determine if a sensor report should associated or not, the tracker ROC curve (i.e., the track Pd & Pft) is required to determine if a track should be associated or not.

6.12.1.1 Role for Distributed Data Fusion

The distributed integration of information and distributed management of a timely response to the situation is the objective of distributed data fusion (DF) and resource management (RM). Automated DF is needed to extract the significant information from the tremendous volume of diverse data. The distributed DF process should include a balance between centralized situation assessment and coordination and fast reaction distributed solutions to enable effectively integrated and timely responses. The accelerating tempo of the situation will force interoperability of netted C^4I systems with automated DF and RM software. The role for automated DF and RM is depicted in Figure 1. The driving requirements are affordability and

mission robustness. To be affordable, the solution must capitalize on recent advances in telecommunications, computers, and standardized software architectures.
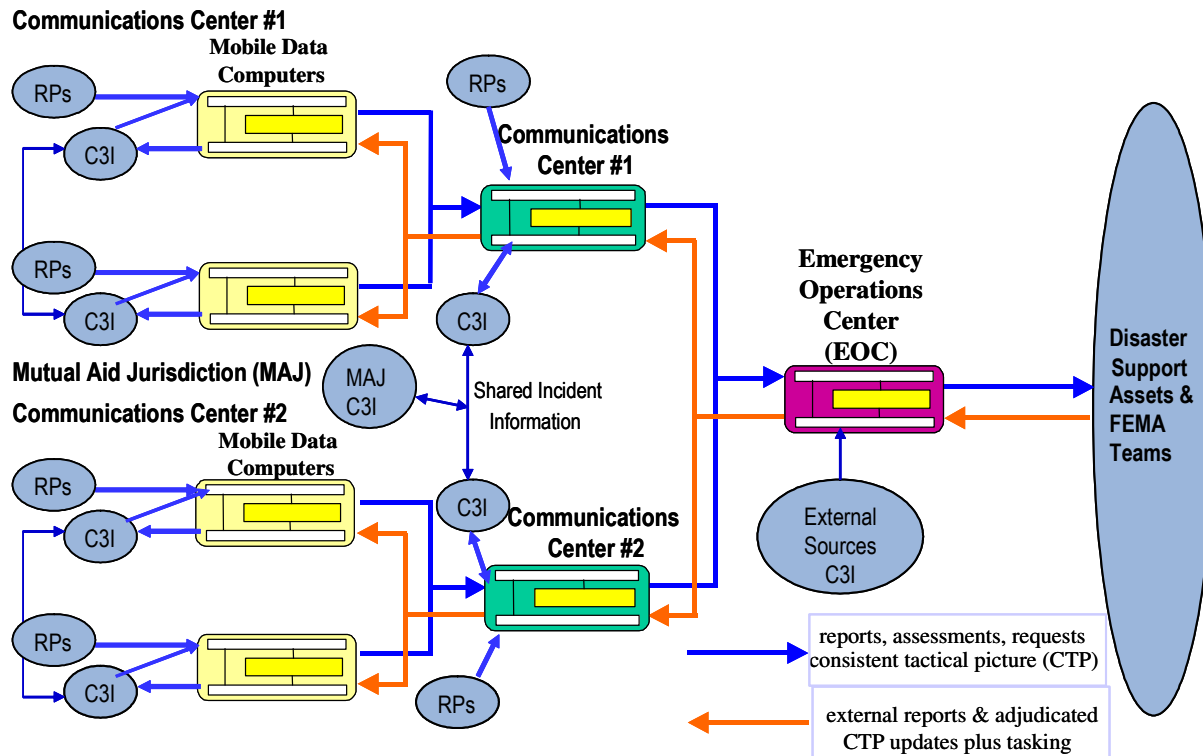
Figure 1: Track Confidence Estimation Is Needed to Support Distributed Data Fusion and Resource Management (DF&RM) Nodes in Dual Node Networks

The distributed fusion parts of the network specify how the data is to be batched (e.g., by sensor, time, and report/data type) and the order in which the batches are to be processed by applying the fusion node paradigm of the Dual Node Network architecture. The fusion and management network is selected to divide-and-conquer the problem so as to achieve the knee-of-the-curve in performance versus complexity/cost. The typical fusion part of the network is a "fan-in tree" that is interlaced with a dual "fan-out tree" for resource management. The result provides local more accurate feedback at higher rates as well as slower and broader situational awareness and coordinated management of the resources [6.12-8], [6.12-11]. The criteria for network optimization include: performance communication bandwidth, processor load, flexibility, fault tolerance, and cost.

AFOSR F49620-01-1-0371

The track confidence equations for fusion are organized using the DF&RM Dual Node Network (Dual Node Network) technical architecture to allow reuse of their implementation patterns via the Dual Node Network toolbox. The baseline approach defined herein enables a priori source probability of detection and false alarm data, to be used for data association scoring (i.e., more accurate report-to-track association) and for the update of the track probability of detection and false track confidence. This allows the reporting confidence thresholds from each of the sources to be reduced (i.e., with higher probability of false alarm albeit at faster update rates and/or higher probability of detection) and for these confidences to be accounted for in the association scoring and state update. As a result, higher confidence fused tracks can be more rapidly presented to the user and with a higher probability of detection in the coverage area. The values of the track kinematics, entity ID, as well as track confidence for all tracks, are updated in the state estimation function in each fusion node. However, the track confidence probability of detection (Pd) and probability of false track (Pft) estimates are mediated and propagated in data preparation and then used in data association to improve the accuracy of the 5 association hypothesis scores. These track confidence equations will need to be tailored to each fusion node and to support higher fusion levels and resource management within the distributed Dual Node Network design.

6.12.1.2 Distributed Fusion Node Processing

The objectives of the track confidence propagation and update techniques are the following:

1. to propagate and update the probability that each track is not a valid entity of interest based upon the current input source a priori statistics and how the current source reports were associated to the track file.

2. to propagate and update the probability that the fused track file is missing a true entity in the current input source field of view (FOV).

Track confidence computations are needed in each of the fusion node components (i.e., data preparation, data association, and state estimation). The probability of detection and probability of false alarm performance statistics initially available from the sources are used to initiate the track confidences at a common time in data preparation. These track confidences are then used

for association scoring in data association. Based upon the association of the source data to the track file, the track confidences are updated in state estimation. The track confidences are then propagated as needed to support the fusion of the next set of tracks or source data in the data preparation of the next fusion node or in state estimation for the user. The Bayesian mathematics necessary to derive the rigorous equations to accomplish this requires an area of mathematics that has not been solved yet. However, useful the approximations that can be used in hypothesis evaluation portion of data association are specified below. The full derivation will be given in the detailed track confidence derivations in Section 3.4.

The most widely used rigorously-based association scoring approaches are the max a posteriori (MAP) criteria for data association and state estimation. The most common of these is the MAP deterministic data association criterion used to select the 'best' hypothesis, which then used to generate the MAP estimate of the system state. Another MAP approach updates the track state confidence for each report based upon its relative association confidence score. This has been termed probabilistic data association. A third criterion is a joint hypothesis and state optimization (JHSO) criterion [6.12-6], [6.12-10]. A comparison of various Bayesian association hypothesis scoring approaches is described in [6.12-5], [6.12-6].

To ease the development of track confidence estimation the Bayesian max a posteriori (MAP) scoring is selected as the baseline approach. This overall MAP report-to-track score is the product of the following 3 report-to-track score components:

1. kinematics and commensurate scoring: P(Y), usually a product of Gaussian density points,

2. noncommensurate sensor attribute scoring: P(Z), a sum of class confidences, P(K), times the priors for the attributes,

3. a priori hypothesis scoring: P(H) as a product of association hypothesis types.

These 3 scores are defined as follows:

$$max\ P(H|R) = max\ \{P(R|H)\ P(H)\} = max\ \{P(Y|H)\ P(Z|Y,H)\ P(H)\}$$

$$= max\ [\Pi_T\ \{P(Y(S)|Y(T),H)\ P(Z(S),\ Z(T)|Y(S),\ Y(T),\ H)\ P(H)\}] \qquad (1)$$

where

1. the maximization's are over all association and non-association hypotheses, H,

2. H is the set of feasible association or non-association hypotheses,

3. R are the track and source report data,

4. Y is the set of kinematics from the tracks, Y(T) and the source reports, Y(S)

5. Z is the set of all parameters & attributes from both which are not available,

6. the product is over all independent labeled track hypotheses (i.e., of all 5 types),

7. the P(Y(T)|H) term is dropped as constant with respect to the maximization,

8. Z(T) & Z(S) are the parameters and attributes from the track & source reports, and

9. P(H) is the a priori confidence in the hypothesis.

The total scene hypothesis score is the product of the individual hypothesis scores for how all the given batch of reports and the CTP tracks are associated (i.e., for each of the 5 types of hypotheses). These scores are summarized below using the 0th order approximation for the P(H) term as follows:

1. Association Hypotheses

$P(Y(S)|Y(T),H)P(Z(S),Z(T)|Y(S),Y(T),H) \ P(H) = \{|V|^{-1/2}\}exp[-\{I^T \ V^{-1} \ I \}/2] \ \{\Sigma_K[P(K|Z(T),Y(T), H)P(K|Z(S),Y(S), H)/P(K|Y(T),Y(S), H)]\}[1-P_{FA}(S)][1- P_{FA}(T)]P_D(S)P_D(T)$ *(2)*

2. Pop-up Hypotheses

$P(Y(S)|H)P(Z(S)|Y(S),H)P(H) = \{|E(V)|^{-1/2}\}exp[-\{\mu\}/2]\cdot[1-P_{FA}(S)] \ [1- P_D(T)] \ P_D(S)\binom{UR}{J}$ *(3)*

3. False Alarm (FA) Hypotheses

AFOSR F49620-01-1-0371

$$P(Y(S)|H) \ P(Z(S)|Y(S), H) \ P(H) = \{|E(V)|^{-1/2}\} \ exp[-\{\mu\}/2] \cdot P_{FA}(S) \ P_D(S) \binom{M}{F} \qquad (4)$$

4. Propagation Hypotheses

$$P(H) = [1-P_{FA}(T)] \ [1- P_D(S)] \ P_D(T) \binom{UT}{L} \qquad (5)$$

5. Track Drop Hypotheses

$$P(H) = \ P_{FA}(T) \ P_D(T) \binom{N}{D} \qquad (6)$$

1. the first term after the sum is the class K element of the entity track ID disjoint class tree

2. the second term after the sum is the class K element of the source report ID tree corresponding to that entity since the conditioning on Y(T) can usually be dropped due to Y(S),

3. the third term after the sum (i.e., in the denominator) is the a priori probability of that class K

PD (S) is the probability of detection of this object reported by the source, which is determined by source testing. Its primary use is in scaling the probability of track propagation, since it appears in all of the report hypotheses. It is estimated as the probability of redetection for the association hypothesis, and as a result is usually high (e.g., >.9). In the hypothesized case of an initial detection by a source, the term in the pop-up, FA, and propagate hypotheses is the probability of detection of a new object.

PFA (S) is the probability of false alarm (FA) of the source for this type of report, which is also determined by source testing. It can be approximated as the expected number of false alarms (i.e., under these report conditions) divided by the number of detected objects plus this expected number of false alarms over the field of view (FOV),

PD (T) is the probability of detection of this object in the central track file, which is the combined probability of detection of this object by any of the sources contributing to the track

file multiplied by [1- P(new object appearing during this time interval)]. If this is very near one, then this term is dominated by the [1- P(new object appearing during this time interval)] term. This term is where Poisson arrival statistics can be used. Equations for the propagation and update of this term will be tailored to the fusion node.

PFA (T) is the probability that this track is a false alarm, which can be estimated by maintaining the track existence confidences over time plus considering the probability of track death during this time interval. The former FA probability will usually decrease over time due to increased tracking confidences. If this resulting track confidence is very near one, then this term is dominated by the probability of track death (i.e., dying in the FOV or moving out of the FOV). This is where Poisson track death statistics can be used. The propagated and updated value for this term from the last fusion node will also be tailored to this fusion node.

To achieve a 0th order approximation to the P(H) term the binomial combinatorics terms can be dropped. This is sufficient for most applications. For the 1st order approximation the values in the combinatorial expressions are:

1. UR (UT) = # uncorrelated reports (# tracks) in H

2. J (L) = # new tracks (# propagated tracks) in H

3. M (N) = # reports (# tracks) in H

4. F (D) = # false alarms (# dropped tracks) in H

For the non-association report hypotheses (i.e., pop-up initiation, and false alarm) the expected value of the kinematics score is used. Namely, the kinematics score equation is used except that the chi-square statistic (i.e., {IT V-1 I }) is replaced with its mean, μ.

For the non-association report hypotheses the innovations covariance is the report covariance, R, for which the inverse square root of the determinant is taken for the up-front multiplier in the kinematics equation above. The noncommensurate term for the non-association report hypotheses is constant with respect to the maximization, since the class tree term sums to one. So it is ignored here. Thus, the non-association report hypothesis score is the product of their (i.e.,

pop-up and false alarm) a priori score given above and their kinematics term with the above two values used in its "V" innovations terms.

For the non-association track hypotheses (i.e., propagation, and drop track), the kinematics, P(Y(T)), and noncommensurate terms are all constant with respect to the maximization. The non-association report hypotheses (e.g., the pop-up and the report false alarm) scores have the a priori terms plus an additional expected value report kinematics multiplicative term. The report noncommensurate term expected value is assumed constant (e.g., using the a priori class tree as the expected type the constant is unity). Each association hypothesis has all three of the terms defined above, where the noncommensurate term is unity whenever either the report or the track does not provide an entity type tree.

The scene with the highest association score (i.e., product of each of its hypotheses scores), as found in Hypothesis Selection, is selected for use in state estimation. The values for the track probability of detection and false alarm terms (i.e., PD (T) and PFA (T)) are those maintained by the track confidence equations defined herein and when the source is a multi-source fusion node the track confidence values are used for the PD(S) and PFA (S) terms as well.

6.12.1.3 Baseline Track Confidence Fusion Node Equations

The track confidence equations will be defined herein using the standard fusion node processing functional flow described above. The inputs to the fusion node are described in terms of the current distributed Consistent Tactical Picture (CTP) track set and a source report batch. Each has a header and a list of contents. The CTP track set header contains summary descriptors common to all tracks in the track set to include the following:

1. track set name and number of tracks contained in CTP track set.

2. track set state time and coordinate system (e.g., geometric and angular axis center vectors) For the Baseline there is only one 3-d coordinate system center.

3. CTP track file probability of detection and false alarm (e.g., average or parameterized by area of interest (AOI)).

For the baseline there is an average probability of detection and false alarm for the CTP track file. A similar computation can be made for the different source coverage areas as needed. A track dependent probability of false alarm is associated with each track number as defined below. Also, the track birth and death statistics (i.e., probability of a new entity pop-up and an entity death over a delta time) are assumed constant over a scenario, and

1. a priori entity ID trees, as available in IPB for mission

The components of each CTP track state are described as follows:

1. track number and false alarm confidence

2. track continuous parameters (kinematic, length, etc.)

3. track discrete attributes (entity ID tree, relationships, etc.)

Common Referencing: Time and Detection & False Alarm Propagation

After the ID/attributes and kinematics/parametrics data are put into a common time frame and known misalignments compensated for, the CTP track file average probability of detection and false alarm, as well as each track probability of being false is propagated to the current source report time. The average probability of detection for the track file is propagated as follows:

P(inclusion of a true entity of interest in the CTP track file at t + Δt) ≡ P_D (at time t+ Δt) = 1 - P(true entity not in track file at t + Δt) (1)

where

P(true entity not in track file at t + Δt) = P(true entity not in track file at t + Δt | entity is a new arrival over Δt) P(entity is a new arrival over Δt) + P(true entity not in track file at t + Δt | entity is not a new arrival over Δt) P(entity is not a new arrival over Δt)

= 1.0 ∗ (expected # of new / [expected # of new + # of tracks at t]) + [1 - P(detection at t)] ∗ [1 - (expected # of new / [expected # of new + # of tracks at t])]

$= P(new) + [1 - P_D \,(at\ time\ t)] * [1 - P(new)]$

    (2)

This P(new) term is where Poisson arrival statistics, if available, are used. This computation can be done per AOI (i.e., maintain a P_D per AOI selected by source coverage or entity type (e.g., air, ground, sea, and signature))

The probability that each CTP track is false is propagated separately (i.e., per track) as follows:

P(track is false at t + Δt) $\equiv P_{FA}$ (track at time t+ Δt)

$= P_{FA} \,(track\ at\ time\ t+\ \Delta t|\ false\ at\ time\ t)\ P_{FA}\ (track\ at\ time\ t)\ +\ P_{FA}\ (track\ at\ time\ t+\ \Delta t|\ not\ false\ at\ time\ t)\ [1 - P_{FA}\ (track\ at\ time\ t)]$

    (3)

$= P_{FA} \,(track\ at\ time\ t) + P(track\ death\ from\ time\ t\ to\ t + \Delta t) * [1 - P_{FA} \,(track\ at\ time\ t)]$

The average probability of false alarm is propagated in the same way. This P(track death from time t to t + Δt) term is where Poisson track death statistics, if available, are used. This term can vary per entity type, time, and source.

Hypothesis Generation

This function performs the ID/attributes and kinematics/parametrics gating to determine the feasible association hypotheses. No track confidence computations are performed here. However, the size of these gates around the CTP tracks (e.g., 5 sigma gates) are partially based upon the track confidences, since these confidences imply a gate beyond which a report is not feasibly associated with a track. This is due to the probability that the CTP track file may not have the reported entity in track.

Hypothesis Evaluation

Bayesian max a posteriori (MAP) scoring is the Baseline approach, since it provides a rigorous combination of kinematics, attributes, and a priori data. For the Baseline this overall MAP report-to-track score is the product of the following 3 report-to-track score components:

1. kinematics & commensurate scoring: P(Y), usually a product of Gaussian density points,

2. noncommensurate attribute scoring: P(Z), a sum of class confidences, P(K), times the priors for the attributes,

3. a priori hypothesis scoring: P(H) as a product of association hypothesis types.

These 3 scores are defined as follows:

max P(H|R) = max {P(R|H) P(H)} = max {P(Y|H) P(Z|Y,H) P(H)}

$$= max \ [\Pi_T \ \{P(Y(S)|Y(T),H) \ P(Z(S), Z(T)|Y(S), Y(T), H) \ P(H)\}] \qquad (4)$$

where

1. the maximization's are over all association and non-association hypotheses, H,

2. H is the set of feasible association or non-association hypotheses,

3. R are the CTP track and source report data,

4. Y is the set of kinematics from both,

5. Z is the set of all parameters & attributes from both which are not available,

6. the product is over all independent labeled track, T, hypotheses (i.e., of all 5 types),

7. Y(T) are the track kinematics, the P(Y(T)|H) term is dropped as constant with respect to the maximization,

8. Y(S) are the source report kinematics,

9. K are the elements of the disjoint class tree,

10. Z(T) are the parameters and attributes from the track,

11. Z(S) are the parameters and attributes from the source report,

12. P(H) is the a priori confidence in the hypothesis.

The track confidences developed herein are applied in the last of these association score components (i.e., the P(H) a priori scoring). Before deriving the P(H) score it is useful to define the other scores, so that the relationships can be understood. These 3 scores are defined in more detail below.

Kinematics Association Scoring

The association hypothesis kinematics scoring for a new incoming source report, Y(S) to an existing track, Y(T) assumes a Gaussian distribution [ellipsoid], with a CTP track covariance P which models the error in the track location due to possible motion. Then the kinematics score is computed as follows:

$$P(Y(S)|Y(T), H) = \{1/ (2\pi)^{d/2} \ |V|^{1/2} \} \ exp[-1/2\{I^T \ V^{-1} \ I \}] \tag{5}$$

where

1. Y(S) are the source report Gaussian kinematics with covariance R,

2. Y(T) are the track Gaussian kinematics with covariance P,

3. H is the hypothesis that the report and track are associated,

4. d is the dimension of the Gaussian kinematics state,

5. |V| is the determinant of the innovations covariance, $V=[\ \phi P \phi^T + Q] + R$,

6. is state transition matrix, Q is the noise covariance, and the measurement matrix, H, is the identity,

7. I is the innovations vector, $I = Y(S) - Y(T)$.

When all the covariances remain constant then the first term can be dropped. This yields the classic Mahalanobis distance measure in the exponent after taking the log and multiplying by (-2). When doing so these conversions also need to be applied to the noncommensurate and a

priori scores given below. In general the covariances are not constant and the MAP scoring used here is <u>not equivalent</u> to Mahalanobis or the integral of the tail of the chi-square.

Noncommensurate Attributes ID Association Scoring

To use noncommensurate scoring requires the attributes and parameters, Z, in the report and track data to be independent when conditioned on the feasible entity ID classes. Namely, information about Z(T) does not help estimate Z(S) when the entity class K is known for each class K. Under this assumption for each report and track pair, the second term scores an entity track ID tree with a source report ID tree as follows:

$$P(Z(S),Z(T)|Y(S),Y(T),H)=P(Z(S)|Y(S), H) \; P(Z(T)|Y(T), H) \; \{\Sigma_K[P(K|Z(T),Y(T), H) \; P(K|Z(S),Y(S), H)/P(K|Y(T),Y(S), H)]\} \qquad (6)$$

where

1. the first two terms in front of the sum are constant with respect to the maximization when they appear in every hypothesis (i.e., as they do in this option, so they are ignored here), also the kinematics conditioning has been restricted to each report and track, respectively,

2. the first term after the sum is the class K element of the entity track ID disjoint class tree, since the conditioning on Y(S) can usually be dropped due to Y(T),

3. the second term after the sum is the class K element of the source report ID tree corresponding to that entity since the conditioning on Y(T) can usually be dropped due to Y(S),

4. the third term after the sum (i.e., in the denominator) is the a priori probability of that class K [note: when denominator is 0 for an ID class K, then whole term in the equation sum is 0], and

5. the term components are as described above.

Note that inclusion of the a priori term enables for example a higher score for the association of a rare entity with its corresponding track rare entity (i.e., when the denominator is small then a

match in ID in the numerator terms will increase the score for association). The class tree for each source and each report is conditioned on only its own kinematics and attributes. Thus, it is derivable from each source individually. Also, when either the report or track noncommensurate attributes do not contribute to the ID, these noncommensurate terms in the equation sum to one (i.e., the class K terms in the tree are disjoint and cover all possibilities).

Also, note that this term only rigorously applies when the current source report attributes are noncommensurate with the track attributes. If previous report attributes have already been fused (i.e., integrated) with the track attributes, then these previous attributes would implicate corresponding attributes in the current report even given the entity class K. Thus the report and the track attributes would be commensurate. When such attributes are available, it is better to use the commensurate scoring in both the report and track (e.g., entity length, pulse descriptors, IR signatures, etc.). Commensurate scoring is data dependent, but is usually performed like the kinematics scoring defined above. In many applications these source attributes are not available. Thus, for the case of previously fused source reports generating an ID the approximate pedigree methods described later in this document are used to generate noncommensurate ID trees that are used in the term defined above.

A Priori Association Hypothesis Scoring

The a priori hypotheses terms, P(H), use the following 1st order approximate scoring equation for each source report S and track T hypothesis.

$$
\begin{aligned}
P(association) &= [1 - P_{FA}(S)][1 - P_{FA}(T)]P_D(S)P_D(T) \\
P(pop-up) &= \{[1 - P_{FA}(S)][1 - P_D(T)]P_D(S)\}\binom{UR}{J} \\
P(FA) &= [P_{FA}(S)P_D(S)]\binom{M}{F} \\
P(propagate) &= \{[1 - P_{FA}(T)][1 - P_D(S)]P_D(T)\}\binom{UT}{K} \\
P(drop) &= [P_{FA}(T)P_D(T)]\binom{N}{D}
\end{aligned}
\tag{7}
$$

where

1. $P_D(S)$ is the probability of detection of this object reported by the sensor, which is determined by sensor testing. Its primary use is in scaling the probability of track

propagation, since it appears in all of the report hypotheses. It is estimated as the probability of redetection for the association hypothesis, and as a result is usually high (e.g., >.9). In the hypothesized case of an initial detection by a sensor, the term in the pop-up, FA, and propagate hypotheses is the probability of detection of a new object.

2. $P_{FA}(S)$ is the probability of false alarm (FA) of the sensor for this type of report, which is also determined by sensor testing. It can be approximated as the expected number of false alarms (i.e., under these report conditions) divided by the number of detected objects plus this expected number of false alarms over the field of view (FOV),

3. $P_D(T)$ is the probability of detection of this object in the central track file, which is the combined probability of detection of this object by any of the sensors contributing to the track file multiplied by [1- P(new object appearing during this time interval)]. If this is very near one, then this term is dominated by the [1- P(new object appearing during this time interval)] term. This term is where Poisson arrival statistics can be used. Equations for the propagation and update of this term will be tailored to the fusion node.

4. $P_{FA}(T)$ is the probability that this track is a false alarm, which can be estimated by maintaining the track existence confidences over time plus considering the probability of track death during this time interval. The former FA probability will usually decrease over time due to increased tracking confidences. If this resulting track confidence is very near one, then this term is dominated by the probability of track death (i.e., dying in the FOV or moving out of the FOV). This is where Poisson track death statistics can be used. The propagated and updated value for this term from the last fusion node will also be tailored to the fusion node.

5. UR (UT) = # uncorrelated reports (# tracks) in H

6. J (K) = # new tracks (# propagated tracks) in H

7. M (N) = # reports (# tracks) in H

8. F (D) = # false alarms (# dropped tracks) in H

Hypothesis Selection

Hypothesis selection (HS) searches the scored association hypotheses to select a consistent set (i.e., a consistent scene) to be used for state estimation (i.e., scene update). In Baseline fusion nodes where more than one report can associated with a given track, but one report or track cannot be used in two distinct hypotheses, HS becomes a <u>labeled set partitioning</u> problem (a subclass of set covering problems which are a subclass of 0-1 integer programming problems). For problems where for example one call for service cluster report can be associated with two distinct call for service tracks, the problem would become one of set covering (i.e., that is allowing multiple track associations per report). The labeling arises from the association hypotheses being declared as true or false (i.e., associations, propagations, and initiations versus deletions and false alarms). Since the field of operations research has developed numerous hypothesis search algorithms, this function will not be emphasized herein. All these hypothesis selection search functions rely upon the association scores that are influenced by the track confidence measures derived above.

State Estimation

The kinematics, ID, and track detection and false alarm confidences are updated each time. The kinematics state and its covariance are updated using a Kalman filter. Namely,

$$Y(T)(k) = Y(T)(k-1) + K[Y(S)(k) - H Y(T)(k-1)] \qquad (13)$$

$$P(k) = [I - KH] P(k-1)$$

where

- K is the Kalman gain, $K = P(k-1) H^T [HP(k-1)H^T + R]^{-1}$

- I is the identity matrix

- P(k) is the covariance of the track kinematics state Y(T) at time k

Entity ID State Update

The entity ID is only updated when a new set of attributes is associated with a CTP track and other requirements are met. The ID update applies another pedigree method to prevent double counting. The Baseline ID update tests include tests for uniqueness and independence that will be described later.

The equation for updating each ID class, C, in the disjoint track class tree with the current entity sensor report class tree (i.e., two ID trees) is given as follows:

$$P(class\ C|T, S, H) = [P(C|T,H)\ P(C|S,H)\ /P(C|Y(S),\ H)]\ /\ \textstyle\sum_K\ [P(K|T,H)$$

$$P(K|S,H)/P(K|Y(S),\ H)] \qquad if\ P(C|H) \neq 0 \qquad\qquad (14)$$

$$= 0 \qquad if\ P(C|H)=0$$

The track detection and false alarm confidences are updated in Baseline as defined below. The CTP track probability of detection, $P_D(T)$, is approximated as follows:

$$P_D(T) = 1 - P(track\ missed\ after\ source\ fusion) \qquad\qquad (15)$$

where

P(track missed after fusion) = P(track missed after fusion| detected before) P(detected before) + P(track missed after fusion| not detected before) P(not detected before)

= P(track dropped and source detection not declared a pop-up| detected before) P(detected before) + P(source missed detection or source detected but report called false| not detected before) P(not detected before)

= P(dropped track| detected before) P(source missed detection or source detected but report called false| detected before) P(detected before) + [P(source missed detection | not detected in track before) + P(report called false| source detected and not detected in track before) P(source detected| not detected in track before)] P(not detected in track before)

In many cases false alarms are rare enough that the following hold:

1. the probability of false alarm for the source is low enough that a detection starts a track, and

2. the probability of false alarm for the CTP track file is low enough that no tracks are dropped.

In these cases the first assumption makes the second term in the second part of the sum above zero (i.e., P(report called false| source detected and not detected in track before) = 0) and the second assumption makes the first term zero (i.e., P(dropped track| detected before) = 0). Thus the update for the CTP track probability of detection, P_D (T), is approximated as follows:

P_D (T updated) = 1 - [P(source missed detection| not detected in track before) P(not detected in track before)]

= {1- [1- P_D (T propagated from t-Δt)] [1- P_D (current S)]}   (16)

= P_D (T propagated from t-Δt) + P_D (current S) - P_D (current S) P_D (T propagated from t-Δt)]

where the first term in the product is one minus the probability that the track was not detected before (i.e., at t-Δt) after being propagated to the current time, and the second term in the product is one minus the average probability that the track was not detected by the current source during the current source time interval.

Probability of Detection without Propagations and Pop-Ups

When the probability of false alarm is higher, then the probability of detection decreases since true tracks may be dropped or true reports may be labeled as false alarms. More specifically, when both the following hold:

P_{FA} (S) >[1 - P_{FA} (S)] [1 - P_D (T)], so that unassociated reports are declared to be false alarms,

P_{FA} (T) >[1 - P_{FA} (T)] [1 - P_D (S)], so that unassociated tracks are dropped,   (17)

then the probability of detection is updated using the P(track missed after fusion) computed from the equation above where,

P(source missed detection or source detected but report called false| detected in track before) = 1,

AFOSR F49620-01-1-0371

[P(source missed detection | not detected in track before) + P(report called false| source detected and not detected in track before) P(source detected| not detected before)] = 1

Also, the P(dropped track | detected in track before) (i.e., not given the reports/tracks) can be approximated by the current P(drop)/[P(drop) + P(propagate) + P(associate)] from the current P(H) statistics. So, for this high probability of false alarm case the $P_D(T)$ is updated as follows:

$$P_D(T \text{ updated}) = 1 - P(\text{track missed after fusion}) \tag{18}$$

$$= 1 - \{P(\text{dropped track | detected in track before}) \; P(\text{detected in track before})$$
$$+ P(\text{not detected in track before}) \}$$

$$= 1 - \{[P_{FA}(T) / \{P_{FA}(T) + [1 - P_{FA}(T)] [1 - P_D(S)] + [1 - P_{FA}(S)] [1 - P_{FA}(T)] P_D(S) \}] P_D(T) + [1 - P_D(T)] \}$$

$$= P_D(T) \{1 - [P_{FA}(T) / \{P_{FA}(T) + [1 - P_{FA}(T)] [1 - P_D(S)] + [1 - P_{FA}(S)] [1 - P_{FA}(T)] P_D(S) \}] \}$$

Note, this results in a reduction in the probability of detection due to unassociated tracks being dropped and unassociated reports being labeled as false alarms. A better estimate for the track file probability of detection can be achieved by estimating P(dropped track | detected in track before, current reports and tracks) from the hypothesis selection significant candidate scene scores. The equations for this estimate remain to be derived.

Probability of Detection with Propagations and Without Pop-Ups

If condition #2 above does not hold (i.e., unassociated tracks are propagated) and condition #1 still does (i.e., unassociated reports are declared false), then $P_D(T)$ does not change in the update. In the case where there are a mixture of track propagations and track droppings, then the proportion of these is used to weight between this and whichever of the alternatives above or below applies (i.e., both of these alternatives cannot hold since pop-up criterion is all or none). Thus, once the track file probability of detection gets high enough, the probability of pop-up approximation equation yields no more pop-ups which will stop increases in track file probability of detection. The track probability of detection will remain high until report false alarms are declared. If the number of these false alarms exceeds expectations, then the track file

probability of detection is estimated (i.e., reduced) from the current statistics. Namely, if # of false report declarations, N_{FA}, is greater than $P_{FA}(S)$ x total # of reports, N_R, (i.e., $N_{FA} > P_{FA}(S)$ x N_R), then the new estimate for $P_D(T)$ is:

$$P_D(T \, new) = 1 \, - \, \{ \, [N_{FA} - P_{FA}(S) \, x \, N_R] \, / \, [N_T + N_{FA} - P_{FA}(S) \, x \, N_R] \, \}$$

and the update equation is:

$$P_D(T \, updated) = P_D(T) \, + K_D \, [P_D(T \, new) - P_D(T)]$$

where

1. # of false report declarations in fusion node = N_{FA}

2. total # of current source reports = N_R

3. total # of current tracks in track file = N_T

4. prior track file probability of detection propagated to current time is $P_D(T)$

5. K_D is the gain for how much weight to give the new estimate. K_D could be fixed (e.g., at 1 or .5) or could be estimated (e.g., using the running total of previous updates in this mode divided by the total number of times in this mode). For Baseline, $K_D = .5$ should be sufficient.

Probability of Detection Without Propagations and With Pop-Ups

If the reverse happens (i.e., unassociated tracks are dropped, but unassociated reports are declared pop-ups), then

$$P_D(T \, updated) = 1 - P(track \, missed \, after \, fusion) \tag{19}$$

$= P(dropped \, track| \, detected \, before) \, P(source \, missed \, detection| \, detected \, before) \, P(detected \, before) + P(source \, missed \, detection| \, not \, detected \, in \, track \, before) \, P(not \, detected \, in \, track \, before)$

$= 1 - \{ [P_{FA}(T) / \{P_{FA}(T) + [1 - P_{FA}(T)] [1 - P_D(S)] + [1 - P_{FA}(S)] [1 - P_{FA}(T)] P_D(S) \}] * [1 - P_D(S)] * P_D(T) + [1 - P_D(S)] [1 - P_D(T)] \}$

Note, in this case the potential of dropping a valid track reduces the track file probability of detection, whereas the initiation of non-associated current source reports increases the track file probability of detection. In all of these cases the average probability of false alarm for the track file computed as described below can be used for $P_{FA}(T)$.

Probability of False Alarm for Associated Tracks

The individual track confidences depend upon whether the track was associated, propagated, or declared a new pop-up track as described below. First, if the track was just associated with a report, then the confidence that the track is a false alarm can be approximated as follows:

P(associated track is false)

= P(associated track is false| track was false before) P(track was false before) + P(associated track is false| track was true before) P(track was true before)   (20)

= P(associated report is false| track was false before) P(track was false before)

$= P_{FA}(Source) P_{FA}(Track)$

1. $P_{FA}(Source)$ is the probability of false alarm for the source report,

2. $P_{FA}(Track)$ is the probability of false alarm for the associated track before update, and

3. an associated track that was true before (i.e., as an input to the fusion node) is not false even if it has been associated with a false report or with the wrong valid report. Thus the second term in the sum above is zero.

Probability of False Alarm for Propagated Tracks

Second, if the track was just propagated (i.e., not associated, but kept in the track file), then the confidence that the track is a false alarm can be approximated as follows:

P(propagated track is false) = 1 - P(propagated track is true)

= 1 - P(propagated track is true| true before) P(true before)

(21)

= 1 - {P(propagated track is true| not detected and true before) P(not detected| true before) +

P(propagated track is true |detected and true before) P(detected| true before)} P(true before)

Where P(propagated track is true | not true before) = 0 and P(propagated track is true| not detected and true before) = 1.

The value of P(propagated track is true| detected and true before) is essentially the probability of the detected entity report being nonassociated with this track (e.g., due to its being statistically to far away) or misassociated to another track (e.g., due to missed detections, misalignments, and other errors). If we let this probability of nonassociation or misassociation be equal to $\wp$, then the P(propagated track is false) is computed as follows:

P(propagated track is false) = 1 - { [1 - P_D(Source)] + $\wp$ P_D(Source) } [1 - P_{FA}(Track)]

= P_{FA}(Track) + (1 - $\wp$) P_D(Source) - (1 - $\wp$) P_D(Source) P_{FA}(Track)   *(22)*

where

1. P_D(Source) is the probability of detection of the source,

2. P_D(Track) is the probability of detection of the track file before update,

3. P_{FA}(Track) is the probability of false alarm for the propagated track before update, and

4. P(propagated track is true | detected and true before) = $\wp$

$\wp$ is estimated by the product of the probability of the track propagation with the sum of the probability of the propagated track's report detection either being incorrectly associated to another track or its being declared a pop-up track (i.e., in the low P_{FA}(source) case) versus its being declared a false alarm (i.e., in the high P_{FA}(source) case). As such the $\wp$ term can be approximated as the probability of correct track propagation from the candidate scene probabilities in hypothesis selection. This is done by adding all the track propagation scenes

scores and dividing by these plus all the remainder of the candidate (e.g., significant) scene scores (i.e., with the track associated). .

If non-associated reports are declared pop-ups, then for the propagated track to be true the detected report must be declared to be a pop-up, so based upon the a priori statistics the approximation becomes:

P(propagated track is true| source detected and true before) = P(track propagation) P(report pop-up) / [P(report-to-track association) + P(track propagation) P(report pop-up)]

= [1 - P_D (Track)] [1 - P_D (Source)] / {1 + [1 - P_D (Track)] [1 - P_D (Source)]}

Thus P(propagated track is false) is approximated from a priori data as follows:

P(propagated track is false) = 1 - { [1 - P_D (Source)] + ([1 - P_D (Track)] [1 - P_D (Source)] P_D (Source) / {1 + [1 - P_D (Track)] [1 - P_D (Source)]}) } [1 - P_{FA} (Track)]   *(23)*

where

1. P_D (Source) is the probability of detection of the source,

2. P_D (Track) is the probability of detection of the track file before update,

3. P_{FA} (Track) is the probability of false alarm for the propagated track before update, and

4. P(propagated track is true | not true before) = 0

5. P(propagated track is true| not detected and true before) = 1

6. P(propagated track is true| detected and true before) = [1 - P_D (Track)] [1 - P_D (Source)] / {1 + [1 - P_D (Track)] [1 - P_D (Source)]} which is an approximation for the a priori probability of a propagation and a pop-up.

For the case of high source probability of detection and low probability of false alarm the following assumption is made:

P(track propagation) P(report pop-up) << P(report-to-track association)

In the case where the non-associated reports are declared to false alarms (i.e., P_{FA} (Source) > (1 - P_{FA} (Source)) (1 - P_D (Track)) then

P(propagated track is true| detected and true before) = P(track propagation) P(report false alarm) / [P(report-to-track association) + P(track propagation) P(report false alarm)]

= [1 - P_D (Source)] P_{FA} (Source) / (1 - P_{FA} (Source) + [1 - P_D (Source)] P_{FA} (Source) = [1 - P_D (Source)] P_{FA} (Source) /[1 - P_D (Source)P_{FA} (Source)]   *(25)*

In this non-associated report false alarm case the P(propagated track is false) is approximated as follows:

P(propagated track is false) = 1 - { [1 - P_D (Source)] + ([1 - P_D (Source)] P_{FA} (Source) P_D (Source) / [1 - P_D (Source)P_{FA} (Source)]} [1 - P_{FA} (Track)]
(26)

where the variables are defined as described above. Note that when P_{FA} (Source) = 1, this last equation reduces to P(propagated track is false) remaining unchanged. Also note that for the case of no current reports (i.e., P_D (source) = 0), P(pop-up track is false) also remains unchanged. Of course, the current propagated individual track probabilities of false alarms are used in the track association and propagation hypothesis updates.

Probability of False Alarm for Pop-Ups

If a report was just declared to be a pop-up (i.e., not associated with any tracks, and used to initiate a track in the CTP track file), then the confidence that the pop-up track is a false alarm can be approximated as follows:

P(pop-up track is false) = 1 - P(pop-up track is true) =

= 1 - P(pop-up track is true| true report) P(true report)   *(27)*

= 1 - {P(pop-up track is true| not detected before and true report) P(not detected before| true report) + P(pop-up track is true | detected before and true report) P(detected before| true report)} P(true report)

Where P(pop-up track is true | not true report) = 0 and P(pop-up track is true| not detected before and true report) = 1.

The value of P(pop-up track is true| detected before and true report) depends upon the probability of the prior track being non-associated with this report (e.g., due to its being statistically to far away) or misassociated to another report (e.g., due to prior missing tracks, misalignments, and other errors). If we let this probability of nonassociation or misassociation be equal to $\mathfrak{R}$, then the P(pop-up track is false) is computed as follows:

$P(pop\text{-}up\ track\ is\ false) = 1 - \{ [1 - P_D(Track)] + \mathfrak{R}\,P_D(Track) \} [1 - P_{FA}(Source)]$

$$= P_{FA}(Source) + (1 - \mathfrak{R})\,P_D(Track) - (1 - \mathfrak{R})\,P_D(Track)\,P_{FA}(Source) \qquad (28)$$

where

1. $P_D(Source)$ is the probability of detection of the source,

2. $P_D(Track)$ is the probability of detection of the track file before update,

3. $P_{FA}(Source)$ is the probability of false alarm for the source report, and

4. P(pop-up track is true| detected before and true report) = $\mathfrak{R}$

$\mathfrak{R}$ is estimated by the product of the probability of the track pop-up with the sum of the probability of the prior track either being misassociated to another report or its being declared a propagated track (i.e., in the low P_{FA} (track) case) versus its being declared a false alarm (i.e., in the high P_{FA}(track) case).

If non-associated tracks are propagated, then for the pop-up track to be true the prior detected track must be propagated, so based upon the a priori statistics the approximation becomes:

$P(pop\text{-}up\ track\ is\ true|\ detected\ and\ true\ before)\ =\ P(track\ propagation)\ P(report\ pop\text{-}up) / P(report\text{-}to\text{-}track\ association) + P(track\ propagation)\ P(report\ pop\text{-}up)$

$$= [1 - P_D(Track)]\,[1 - P_D(Source)] / \{1 + [1 - P_D(Track)]\,[1 - P_D(Source)] \} \qquad (29)$$

In this track propagation case the P(pop-up track is false) is computed as follows:

$P(pop\text{-}up\ track\ is\ false) = 1 - \{ [1 - P_D(Track)] + ([1 - P_D(Source)] [1 - P_D(Track)] P_D(Track)$
$/\{1 + [1 - P_D(Track)] [1 - P_D(Source)] \}) \} [1 - P_{FA}(Source)]$ (30)

$= P_{FA}(Source) + P_D(Track) \{ P_D(Source) + P_D(Track) - P_D(Track) P_D(Source)\} [1 - P_{FA}(Source)]$

where

1. $P_D(Source)$ is the probability of detection of the source,

2. $P_D(Track)$ is the probability of detection of the track file before update,

3. $P_{FA}(Source)$ is the probability of false alarm for the source report, and

4. P(pop-up track is true | not true report) $= 0$

5. P(pop-up track is true| not detected before and true report) $= 1$

6. P(pop-up track is true| detected before and true report) $= [1 - P_D(Track)] [1 - P_D(Source)] /\{1 + [1 - P_D(Track)] [1 - P_D(Source)] \}$ which is an approximation for the a priori probability of a propagation and a pop-up

In the case where the non-associated tracks are declared to be false alarms (i.e., $P_{FA}(Track) > (1 - P_{FA}(Track)) (1 - P_D(Source))$), then

$P(pop\text{-}up\ track\ is\ true|\ detected\ and\ true\ before) = P(pop\text{-}up\ track)\ P(track\ false\ alarm) / P(report\text{-}to\text{-}track\ association) + P(pop\text{-}up\ track)\ P(track\ false\ alarm)$

$= [1 - P_D(Track)] P_{FA}(Track) / (1 - P_{FA}(Track) + [1 - P_D(Track)] P_{FA}(Track) = [1 - P_D(Track)] P_{FA}(Track) / [1 - P_D(Track) P_{FA}(Track)]$
(31)

In this non-associated track false alarm case the P(pop-up track is false) is computed as follows:

P(pop-up track is false) = 1 - { [1 - P_D (Track)] + ([1 - P_D (Track)] P_{FA} (Track) P_D (Track) / [1 -
P_D (Track) P_{FA} (Track)])} [1 - P_{FA} (Source)]   *(32)*

where the variables are defined as described above. Note that when P_{FA} (Track) = 1, P(pop-up
track is false) remains unchanged. Also note that for the initialization case of no prior tracks
(i.e., P_D (Track) = 0), the above expression for pop-ups reduces to the P_{FA}(S) as expected (i.e., ,
P(pop-up track is false) remains unchanged in this case also). Of course, the current propagated
individual track probabilities of false alarm are used in the track association and propagation
hypothesis updates.

The average track file probability of false alarm is computed as follows:

P_{FA} (T) = P_{FA} (T | propagated track) P(propagated track) + P_{FA} (T | associated track) P(
associated track) + P_{FA} (T | pop-up track) P(pop-up track)
         (33)

= { P_{FA} (Track) + P_D (Track) P_D (Source) - P_{FA} (Track) P_D (Track) P_D (Source)} P(propagated
track) + {P_{FA} (Source) P_{FA} (Track)} P(associated track) + { P_{FA} (Source)   + P_D (Track) P_D
(Source) - P_{FA} (Source) P_D (Track) P_D (Source)} P(pop-up track)

 where

1. P(propagated track) = the # of propagated tracks divided by total # tracks in updated CTP
    track file

2. P(associated track) = the # of associated tracks divided by total # tracks in updated CTP
    track file

3. P(pop-up track) = the # of pop-up tracks divided by total # tracks in updated CTP track file

6.12.2 Adjudication Management

The role for data fusion, as depicted in Figure 2, is to combine sensor and source data as directed
by the resource manager to assess the situation and predict its impacts for the user. Data fusion is

not responsible for how its outputs are interfaced to the user, but does update the CTP and enables adjudication flagging of significant CTP information for the user interface function.
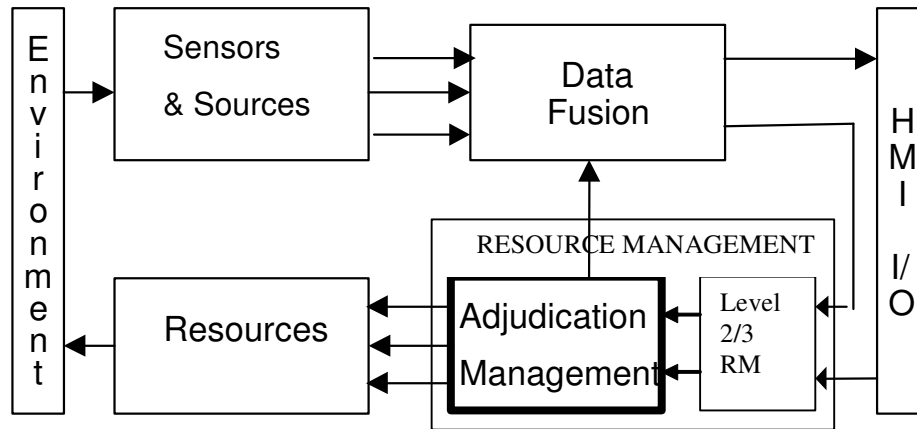
Figure 2: Adjudication under the Direction of the L2/3 RM Manages the Consistency of the Distributed Fusion Track Files via Directives and Advisements within the Communications Bandwidths

To achieve a Consistent Tactical Picture (CTP) requires distributed fusion and adjudication management processes. Adjudication management is a subfunction of Resource Management that is the process of planning/controlling response capabilities to meet mission objectives. Figure 3 depicts a sample distributed fan-in fusion network and fan-out resource management network design that contains adjudication management functions for DA.

The distributed adjudication management node design is based upon the Dual Node Network (Dual Node Network) DF&RM Architecture. The Dual Node Network architecture provides standard components, interfaces, and engineering guidelines for its application. Adjudication can involves Human Machine Interface (HMI) and automated processes and is impacted by organizational guidelines, communications, disaster planning, and other factors. The baseline approach focuses on automated adjudication management processing and makes simplifying assumptions about these and other factors.

AFOSR F49620-01-1-0371
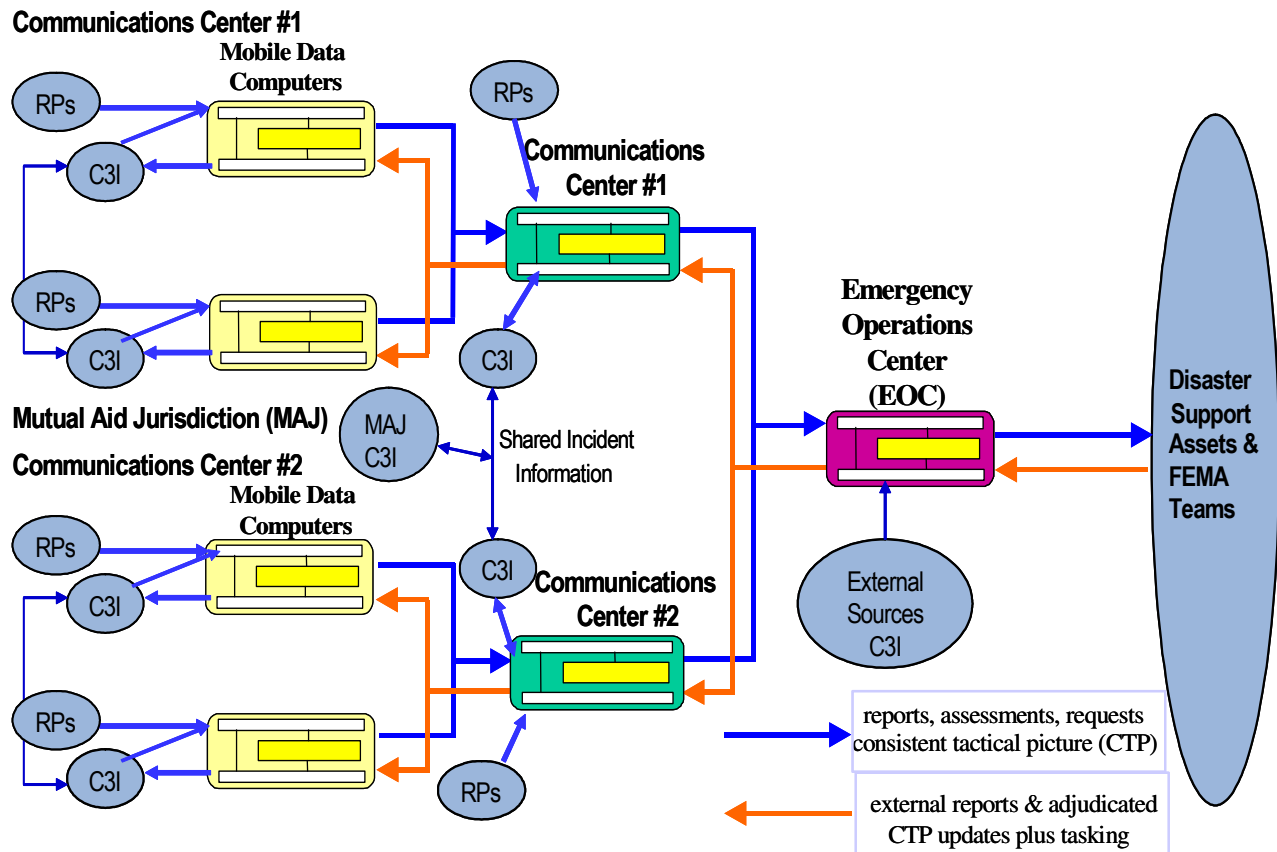
Figure 3: Distributed Fusion Creates Situation Picture and Adjudication Maintains Consistency

6.12.2.1 Adjudication Management Needs

Adjudication management needs to maintain a tactical picture that is consistent across jurisdictions to within the communication and processing time delays by sending adjudication messages between the jurisdictions and the EOC. In future operations this can be extended to include mobile data computers on first responders (e.g., police), jurisdiction communications centers, and the EOC such as depicted in Figure 4.

The tactical pictures of two or more jurisdictions are considered consistent if they have consistent information on entities in their overlapping area of interest (AOI). Information is considered consistent if the following hold:

1. there is a mapping between the pictures (e.g., enabling entity designations at one site to be indicated at the other sites).

2. the errors between the corresponding entity kinematics and attributes are within an acceptable error range (e.g., due to measurement errors, misalignments, and time delays),

3. entities in the picture at one site are represented within an acceptable time delay (e.g., due to communications and processing delays) at other sites
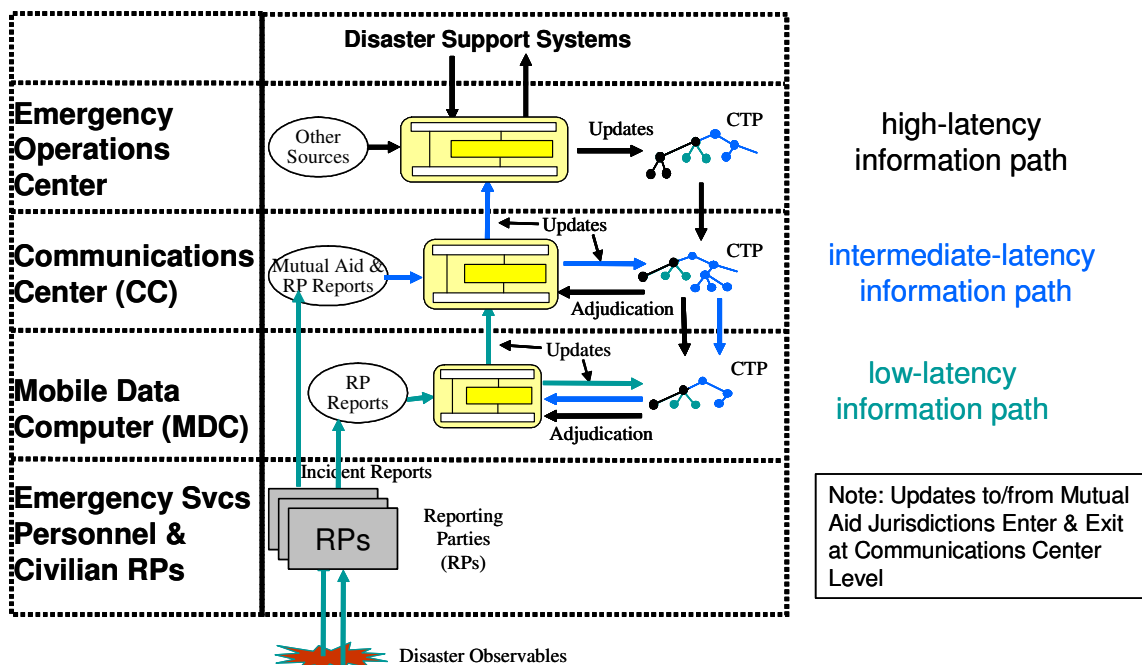
Figure 4: Fusion Updates the CTP with New Data and Adjudication Management Maintains Consistency of Each Commander's Subordinates

Level 4 fusion (i.e., Process Assessment) is applied to compare consistency of the kinematics (position, velocity) and attributes of the given CTP (e.g., the subordinate's CTP) against the commander's CTP. The adjudication process then needs to determine the significant and generate the adjudication directive and advisement response tasks. The measures of performance for this process include the following:

1. Consistency accuracy will be measured by average percentage of non-matching CTP tracks after a suitable time communications delay and the error in the matching CTP tracks.

2. AM performance will trade-off the consistency accuracy of the CTP versus the number of adjudications made, the bandwidth used, and its processing load averaged over a range of scenarios as described by event sequence diagrams such as shown in Figure 5
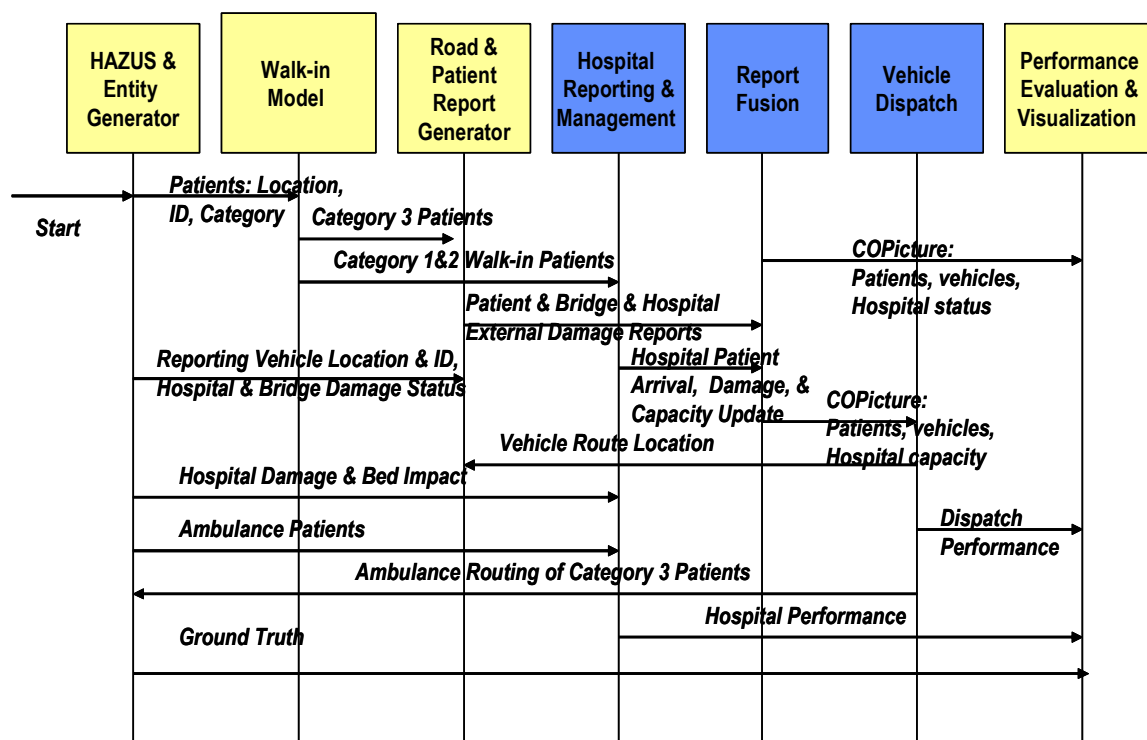
Figure 5: Event Sequence Diagrams Describe Test Scenarios

Adjudication management needs to be able to perform in user selectable modes such as the following:

1. *Independent Mode* – wherein a network node only maintains the CTP locally based upon its fusion of all available input data (e.g., the current PRET distributed L1/2/3 fusion software). Adjudication is primarily performed by the EOC commander or jurisdiction user. No automated adjudication is performed unless commanded by the EOC. The bandwidth

available for adjudication messages will be assessed to determine which mode should be selected during a disaster.

2. *Commander Mode* - wherein the EOC node has been designated as responsible for maintaining consistency of a group of jurisdictions in the network and achieves this by sending CTP management directives to subordinate jurisdictions and by fusing CTP track files from subordinate jurisdictions. Significant changes to the CTP may be sent up echelon (e.g., to FEMA or State) as advisements.

3. *Participant Mode* - wherein each jurisdiction will accept the received CTP management directives of an EOC commander to faithfully modify its CTP and send significant CTP advisements up to the EOC.

4. *Hybrid Mode* - the system element will function across the network as a participant and as a commander simultaneously, with the objective of maintaining consistent tracks.

Under bandwidth limited operations, the AM function needs to be capable of evaluating the feasible adjudications and to score the significance of the changes in call for service data base states. The AM function needs to include the capability to automatically select adjudications based upon these scores as compared to their respective thresholds derived from the communications resource manager inputs. For a command process this includes the ability to select underscore{directives} that add, remove, and modify information in a subordinate's CTP track data (e.g., change a previously reported kinematics, identity, track number, or confidence value). For a subordinate process this includes the ability to select underscore{advisements} that suggest improvements or modifications to the EOC commander's CTP track data.

The AM Function needs to provide adjudication command prioritization to assist in priority-based utilization of available communication assets. Through generating advisements and directives, the AM Function will support the sharing of information with higher and lower levels in the command hierarchy. The output of the AM Function will be adjudication messages (i.e., directives and advisements as well as flagging of mission significant changes) sent to the communications management function. The AM Function will be able to adapt to current operating conditions by increasing or reducing the processing throughput and memory required

and by increasing or decreasing the amount of bandwidth needed. The AM will be able to resume based upon a saved internal state from persistent storage as needed.

6.12.2.2 Blackbox Role Design for Adjudication Management

Adjudication management alternatives that have been considered include the following:

1. Hierarchical Adjudication (Baseline) – 'commanders' maintain consistency of their subordinates via COP advisements from subordinates and directives to their subordinates

2. Centralized Adjudication – one network node performs all adjudication and shares global adjudicated CTP file

3. Organic Data Sharing Self-Adjudication – each node generates its picture based on each node sharing only its organic data

4. Like-Process Adjudication – relies on common fusion algorithmic processing of a fast flat broadcast of all reports

5. Internet-Like Adjudication – sharing of CTP data with all subscribers relying on user initiative for adjudication

6. Hybrid Adjudication – wherein different components of the system use variations of the above approaches.

Of these, Hierarchical Adjudication has been selected as the baseline for this PRET Disaster Assessment effort since it most conducive to the overall DA needs. For this approach adjudication management is distributed over the DA space according to the EOC DA command echelons. The CTP is created at each site by its fusion engine, which combines all available sensor reports at the jurisdictions or tracks at the EOC (i.e., from the jurisdictions) into a picture of the site's AOI. The tactical picture consistency for each jurisdiction is maintained by the adjudication management process located at the EOC. The adjudication process at the EOC sends directives to the jurisdictions to update and correct their CTP's. It also sends advisements to higher commander(s) (e.g., FEMA or State), if there have been any mission significant changes made to the local CTP. This approach to adjudication uses a reduced amount of

bandwidth for large echelons (as opposed to broadcasting the entire CTP or a centralized approach), supports flat dissemination for rapid information dispersal (e.g., by adjudicating duplicates away), and results in a consistent tactical picture. Missing directives can be detected by the system (e.g., using a protocol) and requests for resending can be made. Missing advisements can also be detected and in addition can be compensated for by the adjudication process recognizing that there is still a significant difference between its local CTP and the advisements sent from its subordinates. This process causes another directive to be sent downward. Adjudication management is a subfunction of resource management function that also includes mission, sensor, process, communications, vehicle, and other management functions.

The inputs to Adjudication include the following:

1. CTP tracks just prior to state estimation in fusion and associated CTP track updates due to fusion for each batch of data fused.

2. Adjudication commands from the resource manager (RM) (e.g., threshold values, mission mode commands, adjudication mode management, reassignment of jurisdictions and adjudication commanders) and implementation status from communications manager (e.g., bandwidth mode, link availability, significance level mode)

3. A priori information (e.g., adjudication significance parameters, command structure, communications capability)

4. Saved internal state from persistent storage to reinitialize Adjudication.

The CTP for each update contains a header and information on each track to include the following:

1. CTP header: jurisdiction name/number, time, CTP update number, coverage area changes with probability of detection for each (e.g., due to communications outages)

2. CTP updates per track

Outputs from Adjudication include the following:

1. Adjudication directives to the subordinate jurisdictions via the communications manager

2. Adjudication advisements to the EOC commander via the communications manager

3. Mission significant changes to the CTP that need to be considered for highlighting to the users

4. Prior adjudication decisions that impact the current or future adjudication decisions

5. Self adjudications to modify own CTP

6. Adjudication implementation status to RM (e.g., which can be passed on to the commander per his dictates)

7. Saved internal state to persistent storage

6.12.2.3 The Distributed Fusion and Adjudication Management Network Design

To achieve the requirements described above for adjudication management will require an interlaced network of Level 4 Fusion (i.e., Process Assessment) and Level 1 Management (i.e., Adjudication Management) nodes. The network provides a batching of the functions by management level, site, input type, time, adjudication task type, and entity type such as depicted in Figure 6. A sample management network partitioning by management level and role is shown in Figure 7.
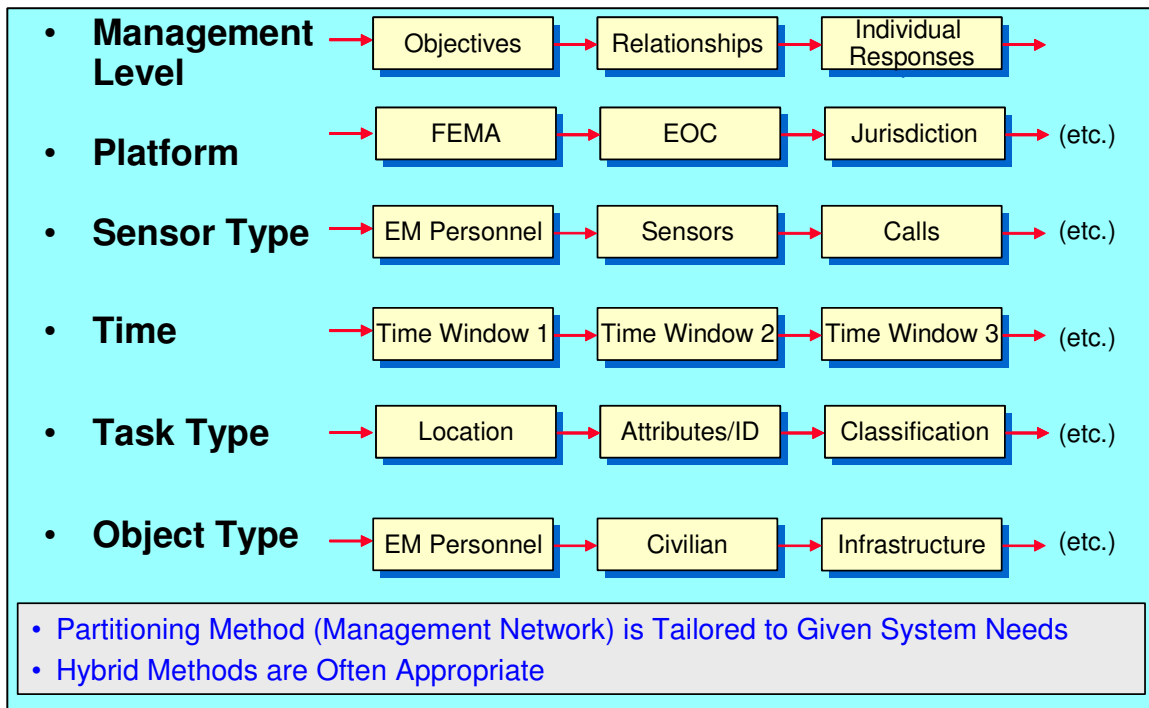
Figure 6: Representative Management Network Partitioning Schemes
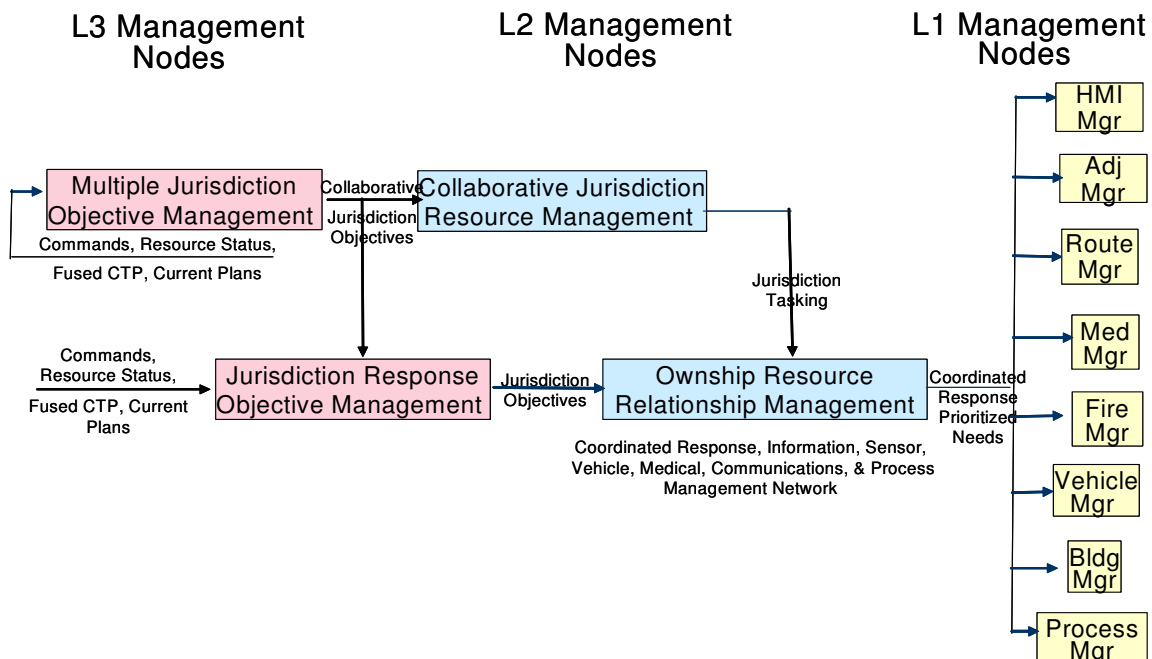
Figure 7: Sample Management Node Network Batched over Management Levels

AFOSR F49620-01-1-0371

As the DF&RM node networks partition the functions into smaller batches the resulting complexity and optimality is reduced. Thus the goal in the DF&RM network design is to achieve the knee-of-the-curve in DF&RM mission performance versus cost and complexity. A sample distributed consistency assessment node network is shown in Figure 8 that batches the jurisdiction CTP's over time to achieve a better estimate of truth upon which to base the consistency assessment.
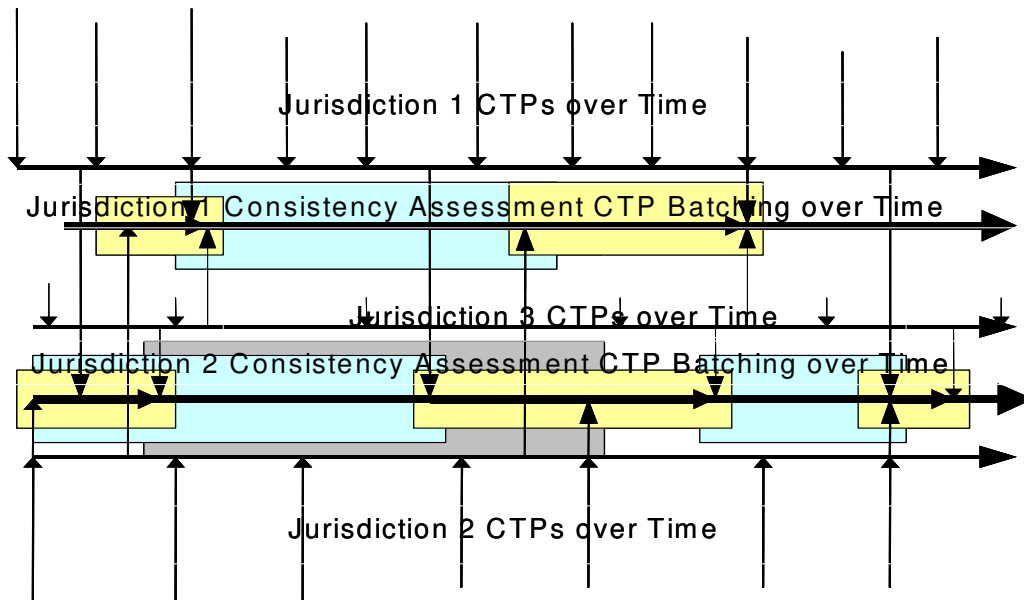
Figure 8: Sample Distributed Consistency Assessment Node Network that Batches CTP's over Time

The baseline distributed consistency assessment and adjudication management node network contains fusion and management nodes tailored to each site (i.e., jurisdiction CC or EOC. The Baseline Adjudication Management network processing is partitioned into nodes that operate on the Consistency Assessment Fusion node anomalous inconsistencies, as depicted in Figure 9. The fusion and management network at each site is also sequential over the CTP updates received at each site. The track updates output by Fusion are stored in the Consistent Tactical Picture (CTP) database for use by the next Fusion node and are used by the Consistency Assessment along with the most current CTP being compared for consistency (e.g., a subordinate CTP). So each time a set of reports is fused, it is used to update the CTP and the update is used

AFOSR F49620-01-1-0371

to detect any significant differences that need to be adjudicated across users. Adjudication Management can also receive directives and advisements via an RM node from others. AM generates, evaluates, and selects the adjudication plan and then creates the prioritized adjudication tasks that are sent to the communications manager (CM) for delivery. Examples of these output adjudication tasks for improved CTP locations, attributes (e.g., medical and damage severity), missing and false tracks that support higher level fusion and RM includes the following:

1. Track Initiation and Drop Adjudications: generated for missing and duplicate tracks

2. Kinematics State Adjudications: generated when a track has fallen outside of the prescribed kinematics accuracy constraints or confidences not sufficiently correct based on track type.

3. ID/Attribute State Adjudications: generated when a track has incorrect ID or attributes state or confidences based on the track type (e.g., medical condition or damage severity).

4.
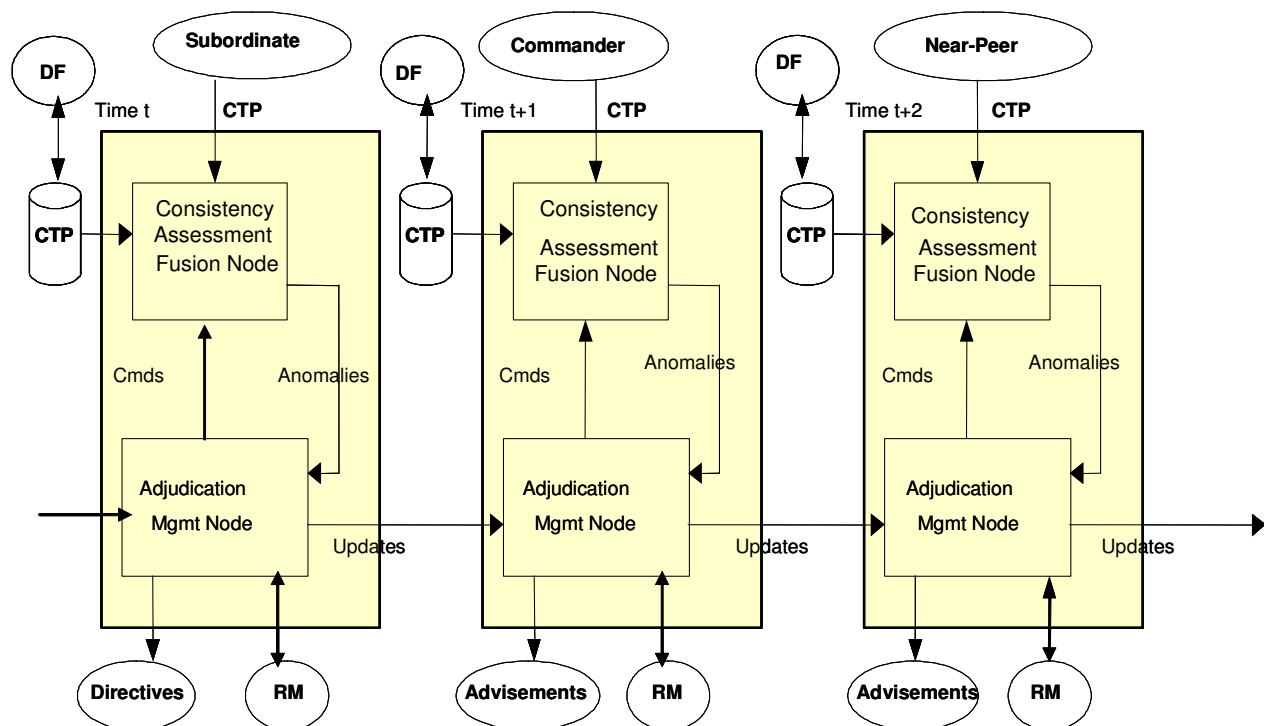
Figure 9: Baseline Distributed Consistency Assessment and Adjudication Management Node Network

For the Baseline DA Fusion and Adjudication Management network outputs (e.g., the advisements and directives) are considered for tasking for CTP implementation by the information management RM node each site. These RM nodes include the algorithms for complying with directives, determining if advisements have already been updated, and resolving conflicts in advisements or between late directives with the current CTP.

This final Adjudication Management network design needs to be selected to achieve the knee-of-the-curve in performance versus cost for Adjudication. The performance of this network will be measured by how well it meets the derived requirements with minimum cost and complexity. The primary accuracy and performance drivers are how consistent the user CTP's are and how much power and bandwidth adjudication requires, while cost will be driven by the life-cycle cost of the software. As more optimality is needed in the adjudication decisions, the Consistency Assessment and Adjudication Management network can take into consideration larger batches of Fusion updates (e.g., over a sliding time window as described above). In this way the adjudication decision on the significance of an update and which adjudications to send can be based upon improved CTP track update histories, instead of just the current CTP track updates. This will result in a combination of improved consistency and/or reduced power/bandwidth at a higher complexity cost per Adjudication node.

6.12.2.4 Adjudication Management Node Processing

According to the Dual Node Network technical architecture the major component functions performed in each Consistency Assessment and Adjudication Management node have a 'dual' correspondences, as described in Figure 10. Namely, each consistent assessment fusion node first determines prepares the data for fusion (e.g., by converting the given CTP's to a common format, time, and reference frame for association decisions as shown in Figure 11. Secondly, the overlapping information is used to generate, evaluate, and select associations of the site's best estimation (e.g., its CTP) with the current comparison (e.g., subordinate) CTP as depicted in Figure 12. Thirdly, the association decisions and the information are used to update the CTP consistency estimates,. These Consistency Assessment Fusion nodes are tailored to the CTP's that are assessed. A detailed explanation of how this is performed as a Level 4 Fusion process is

described in the Appendix: Performance Evaluation Methods for Data-Fusion Capable Tactical Platforms.
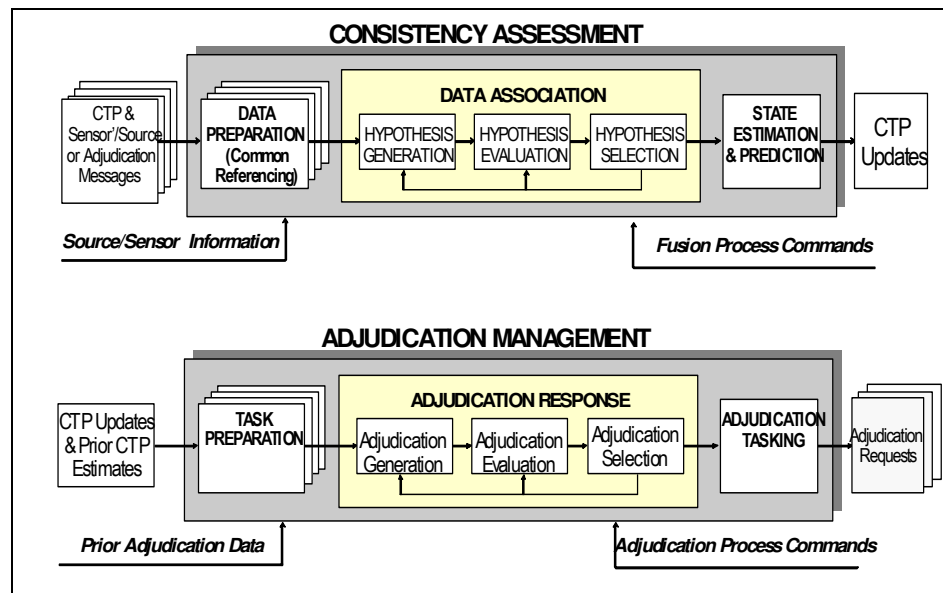
Figure 10: Each Fusion Node and Dual Adjudication Management Node Is Tailored from a Dual Set of Functional Components
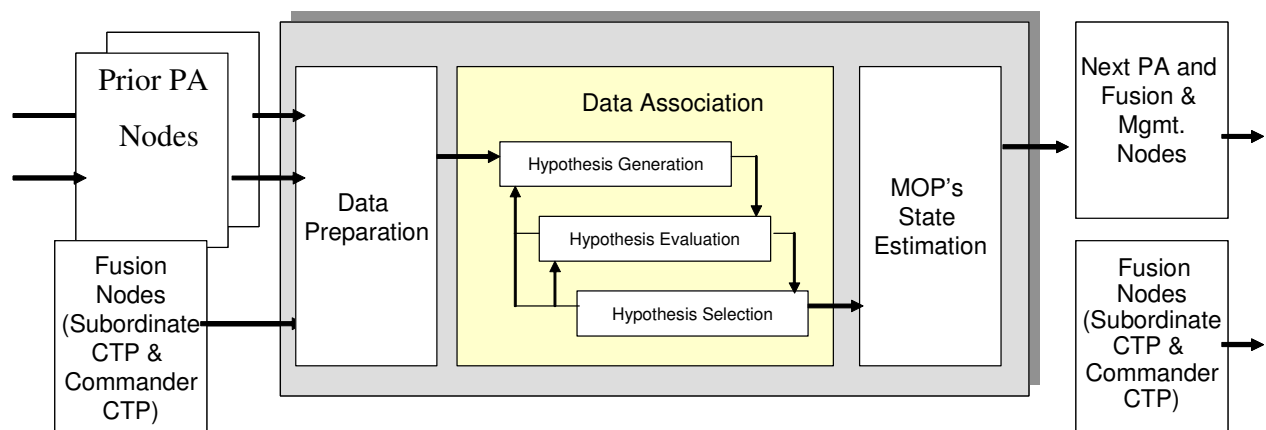
Figure 11: Performance Assessment (Evaluation) (PA) Nodes According to DF&RM Dual Node Network Technical Architecture

Commander's CTP:
25 Objects of Interest

Subordinate's CTP:
27 Objects of Interest

Correspondence Between CTP's Yields Consistency Anomalies
Performance is measured in terms of MOPs:
number missing & false states and errors in matching track states
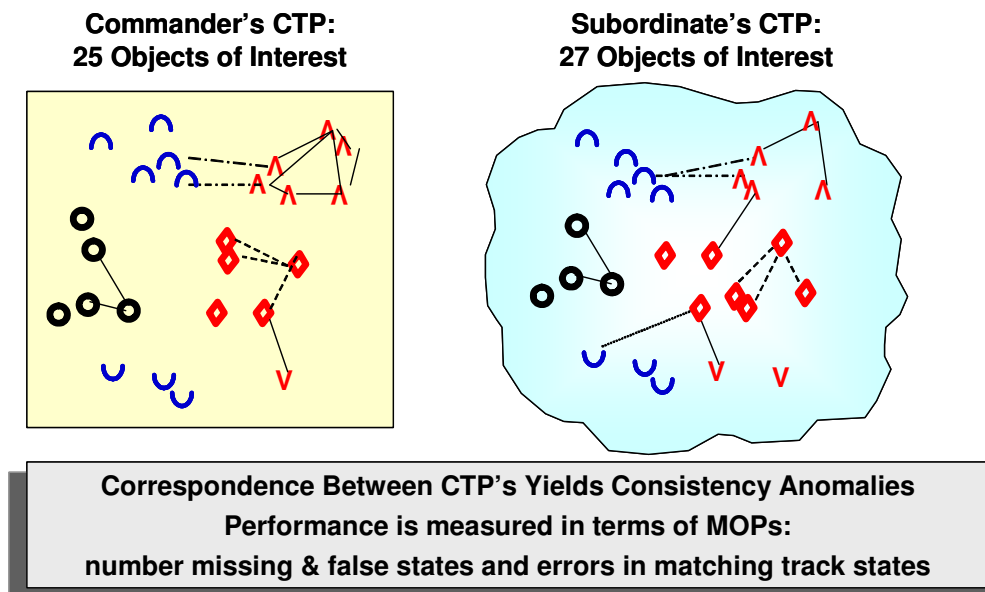
Figure 12: Consistency Assessment Determines the Association of the Local Commander's Best CTP to Each of the CTP's for which Consistency Estimates Are Desired (e.g., a Subordinate's CTP)

In a corresponding manner as per the Dual Node Network architecture, the Adjudication Management nodes perform three 'dual' subfunctions. Namely, each management node performs Task Preparation, Response Task Planning, and Response Tasking as described in Figure 13.

Task Preparation determines which subordinate (or higher commander) CTP track inconsistencies need to be considered (e.g., due to mission significant differences) for adjudication tasking along the chain of command. For these Task Preparation defines and scores or ranks the candidate adjudication tasks. This includes CTP track initiation, track drop, and location plus attributes state changes.

Response Task Planning generates, evaluates, and selects the adjudication response plan that balances the need for adjudication tasking (e.g., directives and advisements based upon the significance of the CTP differences) against the available bandwidth. Task planning also needs to consider latency constraints and the dynamic problem to include how often similar adjudication tasks are sent. Most generally, response task planning is a dynamic labeled set covering decision problem with a set of tasks allocated to each resource having currently assigned or not labels. For

the baseline adjudication task planning each subordinate, near-peer, and higher commander will be assigned a set of candidate adjudication tasks labeled as active or not to be tasked

Based upon the response plan Adjudication Tasking creates the prioritized adjudication task messages to be output to the communications manager for dissemination. In addition the CTP adjudication status is fed back to the RM as needed. As with Fusion, the Adjudication nodes are tailored to the data types being adjudicated. Examples of cases for which tailored adjudication node processing may be needed include Commander CTP inconsistency advisement, Subordinate or own CTP update directive, and Significant CTP event advisement.
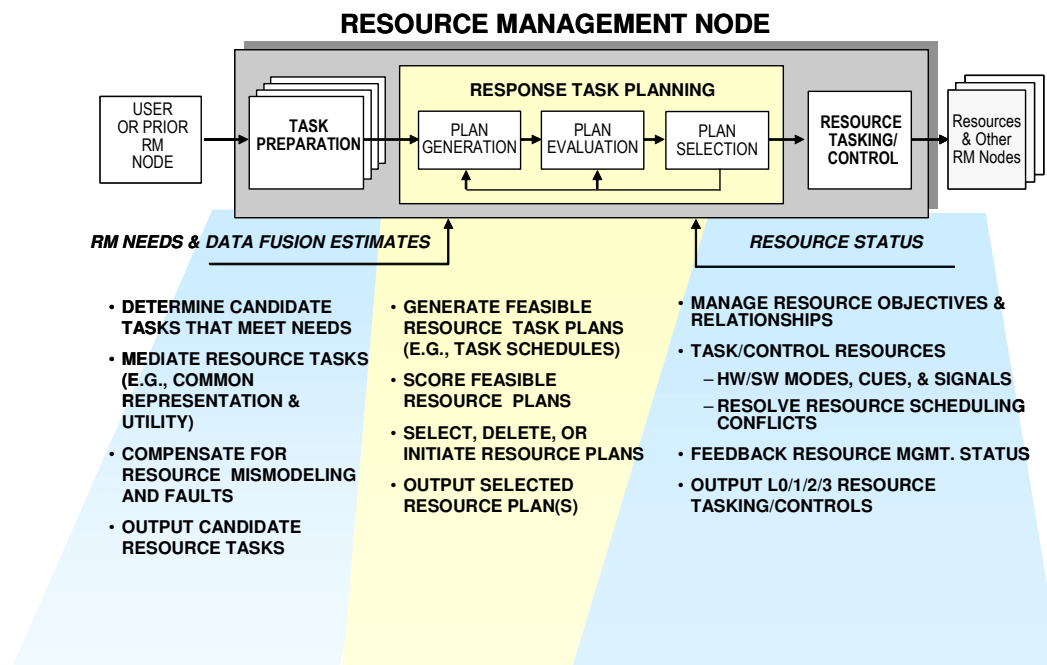
Figure 13: The Dual Node Network Architecture Resource Management Node Functional Partitioning

7. References

[6.2-1] John F. Tangney. Programs in Higher Levels of Information Fusion. *IEEE Aerospace and Electronics Systems Society Magazine,* pages 21-25, Nov. 2003.

[6.2-2] James Llinas. Information Fusion for Natural and Man-Made Disasters. In *Proc. Fifth Int. Conf. Information Fusion,* pages 570-574, Annapolis, MD, USA, 8 July–11 July 2002. Int. Soc. Information Fusion, Sunnyvale, CA 2002.

[6.2-3] U.S. Air Force Scientific Advisory Board Committee on Building the Joint Battlespace Infosphere. Building the Joint Battlespace Infosphere. Report SAB-TR-99-02. Washington D.C., 7 December 2000. U.S. Air Force Scientific Advisory Board, 2002.

[6.2-4] Ikuo Takeuchi, Shigeru Kakumoto and Yozo Goto. Towards and Integrated Earthquake Disaster Simulation System. Proc. First Int. Workshop on Synthetic Simulation and Robotics to Mitigate Earthquake Disasters. Padova, IT, 5 July-8 July 2002, Universita di Roma 2002.

[6.2-5] John Salerno and Paul Bello. Information fusion. Information Institute General Workshop II. Rome NY 10 June-11 June 2003.

[6.2-6] History of ICS. *Incident Command System National Training Curriculum Report*. Washington DC, October 1994. National Wildfire Coordinating Group, 1994.

[6.2-7] Gray Davis and Dallas Jones. *SEMS guidance for special districts*. July 1999, Fresno CA. Standardized Emergency System Advisory Board.

[6.2-8]   Hiroaki Kitano et. al. *Robo-Cup Rescue: search and rescue in large scale disasters as a domain for autonomous agents research*. IEEE Int. Conf. on Sys. Man and Cyb., pages 739-743, Tokyo, October 1999.

[6.2-9] Michael E. Durkin. Fatalities, nonfatal injuries, and medical aspects of the Northridge earthquake. Mary C. Woods and W. Ray Sieple (eds), The Northridge CA earthquake of 17 Jan 1994. California Dept. of Conservation Div. of Mines and Geol. Spec. Pub. 116. pages 247-254, 1994.

[6.2-10] Earl Aurelius, ed. *The January 17 1994 Northridge CA earthquake*. Houston TX 1994. Tech rept of ABS Consulting Inc., 1994.

[6.2-11] Qiang Gong, Arun Jotshi and Rajan Batta. Dispatching/routing of emergency vehicles in a disaster environment using data fusion concepts. *submitted to the Seventh Int. Conf. Information Fusion* Stockholm SW 28 June–1 July 2004.

[6.2-12] D. Lambert. Grand Challenges of Information Fusion. *Proc Sixth Int. Conf. Information Fusion,* pages 570-574, Cairns, Australia, pages 213-220, 8 July–10 July 2003. Int. Soc. Information Fusion, Sunnyvale, CA, 2004.

[6.2-13] A. Rasmussen, J. Pejtersen, L. Goodstein. *Cognitive Systems Engineering*. Wiley, New York, 1994.

 [6.2-14] Wayne Johnson, Ian D. Hall. From Kinematics to Symbolics for Situation and Threat Assessment. *Proc. Conf. on Information, Decision and Control*. Adelaide, Australia, 8 Feb.-10 Feb. 1999. Defence Science and Technology Organization, 1999.

[6.2-15] Eric Little. Foundations of Threat Ontology (ThrO) for Data Fusion Applications. *Center for Multisource Information Fusion Technical Report*, Buffalo NY, 2003.

[6.2-16] Finn. V. Jensen. *An introduction to Bayesian Networks*, Springer, New York, 1996.

[6.2-17] Eveline. M. Helsper, Linda. C. van der Gaag. Building Bayesian networks through ontologies. F. van Harmelen (ed.). *Proc. 15th Euro. Conf. on Art. Int.* pages 680-684, Amsterdam, the Netherlands, IOS Press, 2002.

[6.2-18] Thuong Doan, Peter Haddawy, TienNguyen, and Deva Seetharam. A Hybrid Bayesian Network Modeling Environment. In *The Nat. Comp. Sci. and Eng. Conf. NCSEC* 1999.

[6.2-19] Peter Gärdenfors. *Belief Revision*. Cambridge University Press, Cambridge, U.K., 1992.

[6.2-20] Craig. Boutilier, Nir. Friedman, Joeseph. Halpern.. Belief Revision With Unreliable Observations. *Proc. of the Fifteenth Nat. Conf. on Art. Int.* pages 127--134, Madison, WI, 26 July-30 July, 1998. American Assoc. for Art. Int. Menlo Park CA USA, 1998.

[6.2-21] Didier Dubois and Henri Prade. Introduction: Revision, Updating, And Combining Knowledge. *Handbook of defeasible reasoning and uncertainty management systems Vol. 3.* Kluwer Academic Publishers, London, U.K., 1998.

[6.2-22] Isabelle Bloch, Anthony Hunter, Introduction, Fusion: General Concepts and Characteristics, International Journal Of Intelligent Systems, Vol. 16, 1107–1134, 2001 John Wiley & Sons, Inc, 2001

[6.2-23] Aldo Dragoni. Belief Revision: From Theory To Practice. *The Knowledge Engineering Review*. Vol. 12(2), Cambridge University Press, 1997.

[6.2-24] Salem Benferhat, S., Didier Dubios and Henri Prade. Kalman-like Filtering in a Possibilistic Setting. in the Proc. of the Euro. Conf. on Art. Int. 2000. Berlin GE, 20 Aug.-25 Aug. 2000. European Coordinating Committee for Art. Int., West Lothian, U.K., 2000.

[6.2-25] James Llinas, Christopher Bowman, Galina Rogova, Alan Steinberg, Edward Waltz, and Frank White, Revisions to the JDL Data Fusion Model II, submitted to *the. Seventh Int. Conf. Information Fusion* Stockholm SW 28 June–1 July 2004. Int. Soc. Information Fusion, Sunnyvale, CA., 2004.

[6.3-1] E. Waltz, J. Llinas, Multisensor Data Fusion, Artech House, Norwood MA, 1990.

[6.3-2] K. Vicente, Cognitive Work Analysis, Lawrence Erlbaum Associates, Mahwah NJ, 1999.

[6.3-3] A. Rasmussen, J. Pejtersen, L. Goodstein. (1994) Cognitive Systems Engineering. Wiley, New York.

[6.3-4] J. K. Uhlmann, Efficient Data Association with Multivariate Gaussian Distributed States, US Patent No. 6,239,740, 2001, 2001.

[6.3-5] D.L. Hall, J. Llinas, Handbook of Multisensor Data Fusion, CRC Press, New York, 2001.

[6.3-6] M.R. Endsley, Automation and Situation Awareness, in Automation and Human Performance: Theory and Practice, Lawrence Erlbaum Associates, Mahwah NJ, 1996.

[6.3-7] C.M. Burns et al, Boundary Purpose and Values in Work-Domain Models, MILCOM 2002, pp. 448-454.

[6.3-8] Donald E. Brown, Justin R. Stile et al, Using Simulation to Produce a Data Fusion Decision Support Tool for the Assessment of Manmade and Natural Disasters, 21st Int. Conf. on Modeling, 2003.

 [6.3-9] Anne-Claire Boury-Brisset, "Ontology-based Approach for Information Fusion", CMIF Workshop II on Ontology and Information Fusion, September 2003, Buffalo, NY (see www.infofusion.buffalo.edu).

[6.3-10] Campbell, K. (1990) Abstract Particulars, Cambridge, MA: Basil Blackwell.

[6.3-11] Mark T. Elmore Thomas E. Potok and Frederick T. Sheldon. "Dynamic Data Fusion Using An Ontology-Based Software Agent System" Proceedings of the 7th World Multiconference on Systemics, Cybernetics and Informatics, 2003.

[6.3-12] Gomez-Perez, A., "Some Ideas and Examples to Evaluate Ontologies", Proceedings of the Eleventh IEEE Conference on Artificial Intelligence Applications. Editorial IEEE Computer Society Press. 1995

[6.3-13] P. Grenon (2003) "Knowledge Management from the Ontological Standpoint," in Proceedings of WM 2003 Workshop on Knowledge Management and Philosophy, April, Luzern Switzerland.

[6.3-14] P. Grenon (2003) "Spatiotemporality in Basic Formal Ontology: SNAP and SPAN, Upper-Level Ontology and Framework for Formalization, IFOMIS Technical Report Series, (http://ifomis.de).

[6.3-15] P. Grenon & B. Smith. (2003) "SNAP and SPAN: Towards Dynamic Spatial Ontology," Spatial Cognition and Computation (forthcoming).

[6.3-16] T. R. Gruber, (1993) "A Translation Approach to Portable Ontologies," Knowledge Acquisition, 5: 199-220.

[6.3-17] E. Husserl. (1900-01) Logische Untersuchungen, 2 Bde, Husserliana, Band XIX, Den Haag: Martinus Nijoff, 1985 ed.

[6.3-18] Johansson, I. (1989) Ontological Investigations: An Inquiry into the Categories of Nature, Man and Society, New York, Routledge.

[6.3-19] Kogut, P., Crane, S., Hart, L., Dutra, M., Baclawski, K., Kokar, M., and Smith, J. (2002) UML for ontology development. The Knowledge Engineering Review, 17, 1:61–64.

[6.3-20] M. M. Kokar, J. A. Tomasik, and J. Weyman. Formalizing Classes of Information Fusion Systems. Information Fusion: An International Journal on Multi-Sensor, Multi-Source Information Fusion, Vol. 5(3), pp. 189-202, 2004.

[6.3-21] M. M. Kokar. Situation awareness: issues and challenges. In Proceedings of the Seventh International Conference on Information Fusion, pages 533–534, 2004.

[6.3-22] M. M. Kokar and J. Wang. (2002) Using ontologies for recognition: An example. In Proceedings of the Fifth International Conference on Information Fusion, pp. 1324 – 1343.

[6.3-23] Kokar, M., "Choices in Ontological Languages and Implications for Inferencing", Presentation at Center for Multisource Information Fusion 2004 Workshop III on Critical Issues in Information Fusion, September 2004

[6.3-24] Little, E. (2003) "A Proposed Methodology for Application-Based Formal Ontologies," Proceedings of the Workshop on Reference Ontologies vs. Application Ontologies, 15-18 Sept., University of Hamburg, CEUR-WS.org.

[6.3-25] Little, E., Rogova, G., Boury-Brisset, A.C., (2005) "Theoretical Foundations of Threat Ontology (ThrO) for Data Fusion Applications", TR-2005 -269, Nov. 2005.

[6.3-26] Little, E. & Rogova, G. (2005) "Ontology Meta-Model for Building A Situational Picture of Catastrophic Events," in Proceedings of the FUSION 2005-8th International Conference on Multisource Information Fusion, July 25-29, Philadelphia, PA.

[6.3-27] C. J. Matheus, M. M. Kokar, and K. Baclawski. (2003) "A core ontology for situation awareness". In Proceedings of the Sixth International Conference on Information Fusion, pages 545 –552.

[6.3-28] Chris Nowak. (2003). "On ontologies for high-level information fusion". In Proceedings of the Sixth International Conference on Information Fusion. Cairns, Australia.

[6.3-29] Nowak, C. & Lambert, D. (2005) "The Semantic Challenge for Situation Assessment" in Proceedings of the FUSION 2005-8th International Conference on Multisource Information Fusion, July 25-29, Philadelphia, PA.

[6.3-30] A. Rasmussen, J. Pejtersen, L. Goodstein. (1994) Cognitive Systems Engineering. Wiley, New York.

[6.3-31] Rescher, N. (1955) "Axioms for the Part Relation," Philosophical Studies, 6, 8-11.

[6.3-32] Simons, P. (1987) Parts: A Study in Ontology, Oxford: Oxford Univ. Press.

[6.3-33] Smith, B. (forthcoming) "Against Fantology," in J. Marek and E. M. Reicher (eds.), Experience and Analysis, Vienna: öbv&hpt.

[6.3-34] Smith, B and Grenon, P., (2004) The Cornucopia of Formal Relations, in Dialectica 58: 279-296.

[6.3-35] Smith, B. (1996) "Mereotopology: A Theory of Parts and Boundaries," Data and Knowledge Engineering, 20 (1996), 287–303.

[6.3-36] Steinberg, A. (2005) "An Approach to Threat Assessment," ," in Proceedings of the FUSION 2005-8th International Conference on Multisource Information Fusion, July 25-29, Philadelphia, PA.

[6.4-1] Health Forum. AHA hospital statistics, 2001 edition, American Hospital Association, Chicago, IL.

[6.4-2] Health Forum. AHA guide to the healthcare field, 2001-2002 edition, American Hospital Association, Chicago, IL.

[6.4-3] Health Forum. AHA hospital statistics, 2003-2004 edition, American Hospital Association, Chicago, IL.

[6.4-4] S. Aroni and M. Durkin. Injuries and occupant behavior in earthquakes. In *Proceedings of the Joint US-Romanian Seminar on Earthquakes and Energy*, Washington (DC): Architectural Research Centers Consortium, 1985:3-40.

[6.4-5] K. M. Bretthauer and M. J. Cote. A model for planning resource requirements in health care organizations. *Decision Sciences*, 1998; 29(1): 243-270.

[6.4-6] H. H. Chang. Determinants of hospital efficiency: the case of central government-owned hospitals in Taiwan. *International Journal of Management Science*, 1998; 26(2): 307-317.

[6.4-7] D. H. Cheu. *Northridge earthquake, January 17, 1994: the hospital response*. California Seismic Safety Commission, Sacramento, 1994.

[6.4-8] J. K. Cochran and L. Lin. Compound dynamic event metamodels for electronic assembly line systems. *IIE Transactions*, 1993; 25(2): 12-25.

 [6.4-9] D. M. Cohan. *A hospital patient care flow model of stochastic structure*. PhD dissertation, Pennsylvania State University, University Park, PA, 1972.

[6.4-10] M. J. Cote. Patient flow and resource utilization in an outpatient clinic. *Socio-Economic Planning Sciences*, 1999; 33(3): 231-245. [11] M. E. Durkin. *Fatalities, nonfatal injuries and medical aspects of the Northridge earthquake, Northridge, California earthquake of 17 January 1994*. Edited by Woods MC, Seiple WR. California Department of Conservation, Division of Mines and Geology, Sacramento, 1995: 247-254.

[6.4-12] ECMC 2001. *OR efficiency improvement study*. Report by The Center for Excellence in Global Enterprises Management - Department of Industrial Engineering, State University of New York at Buffalo, Buffalo, NY, Dec 2001.

[6.4-13] FEMA-366, HAZUS99 estimated annualized earthquake loss for the U.S., February 2001; http://www.fema.gov/hazus/dl_eqpub.shtm.

[6.4-14] B. F. Giraldo, P. Martinez and P. Caminal. Parametric modeling of the hospital activity applied to the simulation of patients in waiting-list. *Proceedings of the 22nd Annual EMBS International Conference*, Chicago IL, 2000; 3: 2334-2336.

[6.4-15] A. K. Henderson, S. R. Lillibridge, C. Salinas, R. W. Graves, P. B. Roth and E. K. Noji. Disaster medical assistance terms: Providing health care to a community struck by hurricane Iniki. *Annals of Emergency Medicine*, 1994; 23(4): 726-730.

[6.4-16] R. R. Jean. *Staff allocation and cost analysis: Application of a hospital patient flow model*. M.S. thesis, Virginia Polytechnic Institute and State University, Blacksburg, VA, 1974.

[6.4-17] J. B. Jun, S. H. Jacobson and J. R. Swisher. Application of discrete-event simulation in health care clinics: A survey. *Journal of the Operational Research Society*, 1999; 50(2): 109-123.

[6.4-18] E. P. C. Kao and S. L. Chang. Modeling time-dependent arrivals to service systems: A case in using a piecewise polynomial rate function in a non-homogeneous Poisson process. *Management Science*, 1988; 34(11): 1367-1379.

[6.4-19] M. Lagergren. ASIM-a system for monitoring and evaluation of the care of the elderly and disabled in a municipality. *Health Services Research*, 1993; 28(1): 27-44.

[6.4-20] J. C. Lowery. Multi-hospital validation of critical care simulation model. *Proceedings of the 1993 Winter Simulation Conference*, Los Angeles, CA, USA, 1993: 1207-1215.

[6.4-21] M. Mahue. Methodologies for comparing injury data - II: Impact of Northridge injuries on emergency departments in Los Angeles County. *Second National Workshop on Modelling*

Earthquake Casualties for Planning and Response, Jesuit Retreat House, Los Altos, California, February 5-7, 1996.

[6.4-22] Mercy Hospital, August 2002. *Operating room efficiency improvement study*. Report by The Center for Excellence in Global Enterprises Management - Department of Industrial Engineering, State University of New York at Buffalo, Buffalo, NY, August 2002.

[6.4-23] R. A. Olson and D. E. Alexander. *Summary of proceedings, Second National Workshop on Modeling Earthquake Casualties for Planning and Response*, Jesuit Retreat House, Los Altos, California, February 5-7, 1996.

[6.4-24] Personal Communication with Michael Ford, ER Manager, Mercy Hospital of Buffalo, NY, Nov 2002.

[6.4-25] G. Royston, A. Dost, J. Townshend and H. Turner. Using system dynamics to help develop and implement policies and programmes in health care in England. *System Dynamics Review*, 1999; 15(3): 293-313.

[6.4-26] C. Salinas and J. Kurata. The effects of Northridge earthquake on the pattern of emergency department care. *American Journal of Emergency Medicine*, 1998; 16(3): 254-256.

[6.4-27] J. R. Swisher, S. H. Jacobson, J. B. Jun and O. Balci. Modeling and analyzing a physician clinic environment using discrete-event (visual) simulation. *Computers and Operations Research*, 2001; 28(2): 105-125.

[6.4-28] B. Walker and T. Haslett. System dynamics and action research in aged care. *Australian Health Review: a Publication of the Australian Hospital Association*, 2001; 24(1): 183-191.

[6.4-29] M. L. Weng and A. A. Houshmand. Healthcare Simulation: A case study at a local clinic. *Winter Simulation Conference Proceedings 1999*, Phoenix, AZ, 1999; 2: 1577-1584.

[6.4-30] E. Wolstenholme. A patient flow perspective of U.K. health services: Exploring the case for new "Intermediate Care" initiatives. *System Dynamics Review*, 1999;

15(3): 253-271.

[6.4-31] A. H. Zon and G. J. Kommer. Patient flows and optimal health-care resource allocation at the macro-level: A dynamic linear programming approach. *Health Care Management Science*, 1999; 2(2): 87-96.

[6.4-32] R. Sibbel and C. Urban. *Agent-Based modeling and simulation for hospital management*. Kluwer Academic Publishers, Boston, 2001.

[6.4-33] A. R. Mahachek. An introduction to patient flow simulation for health-care managers. *Journal of the Society for Health Systems*, 1992; 3(3): 73-81.

[6.4-34] C. R. Standridge. A tutorial on simulation in health care: applications and issues. *Proceedings of the 1999 Winter Simulation Conference*, Phoenix, AZ, 1999, 1:49-55.

[6.4-35] Pan American Health Organization (PAHO). *Establishing a mass casualty management system,* Washington, D.C., USA, 1995.

[6.4-36] G. E. Nacey. *Maximizing hospital capacity, increasing patient through-put by improving the patient flow continuum*. Tele-Tracking Technologies, Inc, Pittsburg, PA, 2004.

[6.4-37] W. McClure. *Reducing excess hospital capacity*. U.S. Department of Commerce, 1976.

[6.4-38] HEW, Bureau of Health Planning. *Health care facilities existing and needed; Hill-Burton State Plan Data as of January 1975*, Washington, DC, 1977.

[6.4-39] M. Roemer and M. Shain. *Hospital utilization under insurance*. American Hospital Association, Chicago, IL, 1959.

[6.4-40] J. C. Bailey. *Long term bed need projections*. Indiana State Department of Health Information Services Commission, Health Planning Division, Indianapolis, IN, 1994.

[6.4-41] P. Trye, N. Murray, I. Wolstencroft and A. Stewart. Health service capacity modeling. *Australian health review: a publication of the Australian Hospital Association*, 2002; 25(4): 159-168.

[6.4-42] A. M. Mouza. Estimation of the total number of hospital admissions and bed requirements for 2011: the case for Greece. *Health services management research*, 2002; 15(3): 186-192.

[6.4-43] J. de Boer, B. Brismar, R. Eldar and W. H. Rutherford. The medical severity index of disasters. *The Journal of Emergency Medicine*, 1989; 7(3): 269-273.

[6.4-44] P. R. Harper. A Framework for operational modelling of hospital resources. *Health Care Management Science*, 2002; 5(3): 165-173.

[6.4-45] J. A. Rodger, D. J. Paper and P. C. Pendharkar. An empirical study for measuring operating room quality performance attributes. *The Journal of High Technology Management Research*, 1998; 9(1): 131-156.

[6.4-46] J. Cohen. *Statistical power analysis for the behavioral sciences*, 2nd ed., Lawrence

Erlbaum Associates, City??Hillsdale, NJ, 1988.

 [6.4-47] September 11, 2001 attacks; http://en.wikipedia.org/wiki/September_11,_2001_attacks.

[6.4-48] Some of the deadliest natural disasters since 1900. Associated Press, New York,

[6.5-1] Zhan F. and Noon, C. Shortest Path Algorithms: An Evaluation using Real Road Networks, *Transportation Science* 14:211-242, 1998.

[6.5-2] Chou, Y.,Romeijn, H., Smith R. Approximating Shortest Paths in Large Scale Networks with an Application to Intelligent Transportation Systems, *INFORMS Journal on Computing* 10: 163-179, 1998.

[6.5-3] Yi, P., George S., Paul, J. and Lin, L. Hospital capacity planning for emergency management in disaster mitigation. To appear in *Socio-Economic and Planning Sciences*, 2007.

[6.5-4] Yen, J. Finding the k shortest loop-less paths in a network. *Management Science* 17: 712-716, 1971.

[6.5-5] Johnson, P.E., Joy, D.S., Clarke, D.B., Jacobi, J.M. HIGHWAY 3.01, An enhanced highway routing model: program, description, methodology, and revised user manual. Technical Report, ORNL/TM- 12124, Oak Ridge National Lab, Oak Ridge, Tennessee, 1992.

[6.5-6] Lombard, K., Church, R.L. The gateway shortest path problem: generating alternative routes for a corridor location problem. *Geographical Systems* 1: 25-45, 1993.

[6.5-7] Kuby, M., Zhongyi, X., Xiaodong, X. A minimax method for finding the k best differentiated paths. *Geographical Analysis* 29: 298-313, 1997.

[6.5-8] Erkut, E. The discrete p-dispersion problem. European Journal of Operational Research 46: 48-60, 1990.

[6.5-9] Erkut, E., Iksal, Y., Yenicerioglu, O. A comparison of p-dispersion heuristics. Computers and Operational Research 21: 1103-1113, 1994.

[6.5-10] Tyagarajan, K., Batta, R., Karwan, M.H., Szczerba, R. Routing aircraft to minimize the chance of detection during mission ingress. Under review by Military Operations Research.

[6.5-11] http://www.people.hofstra.edu/geotrans/eng/ch5en/meth5en/ch5m2en.html

[6.5-12] Myles, A. Publication 2321: Extension Service of Mississippi State University, cooperating with U.S. Department of Agriculture. Published in furtherance of Acts of Congress, May 8 and June 30, 1914. JOE H. MCGILBERRY, Director (500-07-03)

[6.6-1] D. Ozisik and N. Kerle , Post-earthquake damage assessment using satellite and airborne data in the case of the 1999 Kocaeli earthquake, Turkey. *Proc. of the XXth ISPRS congress: Geo-imagery bridging continents*, Istanbul, Turkey, 2004, 686-691.

[6.6-2] J. Llinas, Information Fusion for Natural and Man-Made Disasters, *Proc. 5th International Conference on Information Fusion*, Annapolis, MD, USA, 2002.

[6.6-3] L. A. Treinish, Visual Data Fusion for Applications of High-Resolution Numerical Weather Prediction, *Proc. of IEEE 2000 Visualization Conference*, Salt Lake City, UT, USA, 2000 , 477-480.

[6.6-4] K. Severance, P. Brewster, B. Lazos, and D. Keefe, Wind Tunnel Data Fusion and Immersive Visualization: A Case Study, *Proc. of IEEE 2001 Visualization Conference*, San Diego, CA, USA, 2001.

[6.6-5] T.-J. Hsieh, F. Kuester, and T. C. Hutchinson, Visualization of Large-Scale Seismic Field Data, *Proc. of 2003 High Performance Computing Symposium*, Orlando, FL, USA, 2003.

[6.6-6] G. Srimathveeravalli, N. Subramanian, and T. Kesavadas, A Scenario Generation Tool for DDF Simulation Testbeds, *Proc. of 2004 Winter Simulation Conference*, Washington D.C., 2004.

[6.6-7] I. Rauschert, P. Agrawal, S. Fuhrmann, I. Brewer, R. Sharma, G. Cai, A. MacEachren, and H. Wang, Designing a Human-Centered, Multimodal GIS Interface to Support Emergency Management, *Proc. of ACM International Symposium on Advances in Geographic Information Systems*, McLean, VA, USA, 2002.

[6.6-8] S. Jain and C. McLean, A Framework for Modeling and Simulation for Emergency Response, *Proc. of 2003 Winter Simulation Conference*, New Orleans, Louisiana, USA, 2003.

[6.6-9] Y. Kim and T. Kesavadas, Automated Dynamic Symbology for Visualization of High Level Fusion, *Proc. of 7th International Conference on Information Fusion*, Stockholm, Sweden, 2004, 944-950.

[6.6-10] Northridge Earthquake, Southern California Earthquake Center. Retrieved on March 20, 2005 from http://www.data.scec.org/chrono_index/northreq.html.

[6.6-11] HAZUS, Multihazard Loss Estimation Methodology, FEMA. Retrieved on January 11, 2005 from http://www.fema.gov/hazus/.

[6.6-12] P. D. Scott and G. L. Rogova, Crisis Management in a Data Fusion Synthetic Task Environment, *Proc. of 7th International Conference on Information Fusion*, Stockholm, Sweden, 2004, 330-337.

[6.6-13] Q. Gong, A. Jotshi, and R. Batta, Dispatching/Routing of Emergency Vehicles in a Disaster Environment using Data Fusion Concepts, *Proc. of 7th International Conference on Information Fusion*, Stockholm, Sweden, 2004, 967-974.

[6.6-14] S. Xiaoxia and Z. Quihai, "The Introduction on High Level Architecture (HLA) and Run-Time Infrastructure (RTI)," *Presented at SICE Annual Conference*, Fukui, Japan, 2003.

[6.6-15] K. Watson, D. Espinosa, Z. Greenvoss, J. H. Pedersen, C. Nagel, J. D. Reid, M. Reynolds, M. Skinner, and E. White, *Beginning Visual C#*. Indianapolis, IN: Wiley Publishing, Inc., 2003.

[6.6-16] USGS, U.S. Geological Survey. Retrieved on January 11, 2005 from http://www.usgs.gov.

[6.6-17] TeleAtlas. Retrieved on March 20, 2005 frm http://www.teleatlas.com.

[6.6-18] G. M. Karam, Visualization using Timelines*, Proc. of the 1994 ACM SIGSOFT international symposium on Software testing and analysis*, Seattle, Washington, USA, 1994, 125-137.

[6.6-19] E. White, C. Garrett, and S. Robinson, *GDI+ Programming: Creating Custom Controls Using C#*. Birmingham, UK: Wrox Press Ltd., 2002.

[6.6-20] A. Schilling and A. Zipf, Generation of VRML city models for focus based tour animations: integration, modeling and presentation of heterogeneous geo-data sources, *Proc. of the eighth international conference on 3D Web technology*, Saint Malo, France, 2003.

[6.6-21] C. M. Hoffman, Y. J. Kim, R. P. Winkler, J. D. Walrath, and P. J. Emmerman, Visualization for Situation Awareness, *Proc. of the 1998 workshop on New paradigms in information visualization and manipulation*, Washington D.C., United States, 1998.

[6.6-22] K. Chen and L. Liu, ClusterMap: Labeling Clusters in Large Datasets via Visualization, *Proc. of ACM Conference on Information and Knowledge Management*, Washington D.C., USA, 2004, 285-293.

[6.7- 1] M. Turoff, Past and Future Emergency Response Information Systems, Communications of the ACM, (45) 2002, pp. 29-32.

[6.7- 2] D. L. Westphal, T. R. Holt et al, Meteorological Reanalyses for the Study of Gulf War Illnesses: Khamisiyah Case Study, Weather and Forecasting (14) 1999, pp. 214-221.

[6.7- 3] N. Belkin, C. Cool et al, The effect multiple query representations on information retrieval system performance, ACM Conference on Research and Development in Information Retrieval, 1003, pp. 339-346.

[6.7-4] Z. Zhang, J. Salerno et al, Using data mining techniques for building fusion models, Data Mining and Knowledge Discovery: Theory, Tools, and Technology V. Edited by Belur V. Dasarathy. Proceedings of the SPIE (5098) 2003, pp. 174-184.

[6.7- 5] J. Kohlas, R. Haenni et al, Assumption Based Reasoning and Probabilistic Argumentation Systems, Tech. Rept 96-07, U. Fribourg, Theoretical Computer Science, 1996.

[6.7-6] D. B. Turner, Workbook of Atmospheric Dispersion Estimates, Lewis Publishers, Washington D.C., 1994.

[6.9-1] M. L. Hinman, Some Computational Approaches for Situation Assessment and Impact Assessment, Fusion 2002, Annapolis MD, July 8-11, 2002, pp 687-693

[6.9-2] State of California Emergency Plan, Governor's office of emergency services, 1998.

[6.9-3] J. Llinas. *Information Fusion for Natural and Man-Made Disasters*. In *Proc. Fifth Int. Conf. Information Fusion,* pages 570-574, Annapolis, MD, USA, 8 July–11 July 2002.

[6.9-4] P Scott, G. Rogova, *Crisis Management in a Data Fusion Synthetic Task Environment*, in: *Proc. of the FUSION'2004-7thConference on Multisource Information Fusion*, 2004.

[6.9-6] P. Thagard, C. P. Shelley, Abductive reasoning: Logic, visual thinking, and coherence, In M.-L. Dalla Chiara et al. (Eds.), *Logic and scientific methods*. Dordrecht: Kluwer, 413-427,1997.

AFOSR F49620-01-1-0371

[6.9-.7] G. Harman, The Inference to the Best Explanation, *Philosophical Review* 64, 88-95, 1965.

[6.9-.8] A. Bisantz, G. Rogova, E. Little, On the Integration of Cognitive Work Analysis within Information Fusion Development Methodology, in *Proc. of the Human Factors and Ergonomics Society Annual Meeting*, New Orleans, 2004

[6.9-9] A. Rasmussen, J. Pejtersen, L. Goodstein. *Cognitive Systems Engineering*. Wiley, New York, 1994.

[6.9-10] E. Little, G. Rogova, Ontology Meta-Model For Building A Situational Picture Of Catastrophic Events, in: *Proc. of the FUSION'2005-8th Conference on Multisource Information Fusion*, 2005.

[6.9-11] A. E. Nicholson and J. M. Brady, *Dynamic belief networks for discrete monitoring*. IEEE Transactions on Systems, Man, and Cybernetics 24(11), 1994

[6.9-12] J. Llinas, C. Bowman, G. Rogova, A. Steinberg, E. Waltz, and F. White, Revisions to the JDL Data Fusion Model II, in: Proc. of the FUSION'2004-7th Conference on Multisource Information Fusion, 2004.

[6.9-13] G. Rogova, P. Scott, C. Lollett, Distributed Fusion: Learning in multi-agent systems for time critical decision making, in Data Fusion for Situation Monitoring, Incident Detection, Alert and Response Management, E. Shahbazian, G. Rogova, P. Valen (eds), FOI Press, 123-152, 2005.

[6.9-14] G. Shafer, *A Mathematical Theory of Evidence*, Princeton University Press, 1976

[6.9-15] N. Friedman, J.Y Halpern, Modeling Beliefs in Dynamic systems, Part I, Foundations, *Artificial Intelligence*, 95:257--316, 1997.

[6.9-16] Y. Shi, Y. Song and A. Zhang, *A shrinking-based approach for multi-dimensional data analysis*, Proceedings of the 29th Very Large Data Bases Conference, Berlin Germany, 2003.

[6.9-17]   Winter, Distances for uncertain topological relations, ESF-NSF Summer institute for geographic information, Berlin, 1996.

[6.9-18] J. Josephson, On the Logical Form of Abduction, *AAAI Spring Symposium Series: Automated abduction*, pages 140-144, 1990.

[6.9-19] R. Haenni, J. Kohlas, N. Lehmann. Probabilistic Argumentation Systems, in: J. Kohlas, S. Moral (eds). *Handbook of Defeasible Reasoning and Uncertainty Management Systems,* Vol. 5, Kluwer, 2001.

[6.9-20]   P. Smets and R. Kennes, The transferable belief model, *Artificial Intelligence*, 66, 1994, 191-243

[6.9-21] G. Rogova, Higher Level Fusion: Issues and Design Approaches, in: *Data Fusion Technologies for Harbor Protection,* E. Shahbazian, M. DeWeert, G. Rogova, (eds), Springer, in print.

[6.9-22] E. Little, G. Rogova, Formal ontology and Higher Level Fusion, under review, *Information fusion*, Elsevier.

[6.9-23] G. Rogova, P. Scott, C. Lollett, R. Mudiyanur, Reasoning about situations in the early post-disaster response environment, in: *Proc. of the FUSION'2006-9th Conference on Multisource Information Fusion*, 2006.

 [6.9-24] G. Rogova, P. Scott, C. Lollett, Higher level fusion for post-disaster casualty mitigation operations, in: *Proc. of the FUSION'2005-8th Conference on Multisource Information Fusion*, 2005.

[6.9-25] E. Little, G. Rogova, Ontology Meta-Model For Building A Situational Picture Of Catastrophic Events, in: *Proc. of the FUSION'2005-8th Conference on Multisource Information Fusion*, 2005.

[6.9-26] P. Scott, G. Rogova, Crisis Management in a Data Fusion Synthetic Task Environment, in: *Proc. of the FUSION'2004-7thConference on Multisource Information Fusion*, 2004.

[6.9-27] A. Bisantz, G. Rogova, E. Little, On the Integration of Cognitive Work Analysis within Information Fusion Development Methodology, in *Proc. of the Human Factors and Ergonomics Society Annual Meeting*, New Orleans, 2004.

[6.10-1] Ah-Dhaher AHG and Mackesy D, Multi-sensor data fusion architecture, Proc IEEE HAVE 2003.

[6.10-2] Akita RM, User based data fusion approaches, Proc ISIF 2002

[6.10-3] Allouche MK, Getting over the edge between blackboard and multi-agent systems, Tech Rept 2004-09-16, Lockheed Martin Canada, 2004.

[6.10-4] Aude EPL et al, CONTROLAB MUFA: a multi-level fusion architecture for intelligent navigation of a telerobot, Proc IEEE Conf on Robotics and Auto, 1999.

[6.10-5] Brenner W et al, Intelligent SW agents, Springer (a book) 1998.

[6.10-6] Broder S et al, Robotic heterogeneous multi-sensor fusion with spatial and temporal alignment, Proc Conf Dec & Contr, 1991.

[6.10-7] Carvalho H, A general data fusion architecture, Proc ISIF 2003.

[6.10-8] Chaudhuri SP and Das S, Neural networks for data fusion, 1990

[6.10-9] Clark V, Information fusion acrhitectures for next generation avionics systems, IEEE NAECON 1996

[6.10-10] Dasarathy B, Sensor fusion potential exploitation – innovative architectures and illustrative applications, Proc IEEE 1997.

[6.10-11] Engelmore R and Morgan T, Blackboard systems, Addison Wesley (a book) 1988.

[6.10-12] Fountain G and Drager S, High performance real-time fusion architecture, Proc ISIF 2002.

[6.10-13] Gad A and Farooq M, data fusion architecture for maritime surveillance, Proc ISIF 2002.

[6.10-14] Gatepaille S et al, Data fusion multi-agent framework, SPIE 4051, 2000.

[6.10-15] Gorodetski V et al, Multi-agent fusion systems: design and implementation issues, ISIF 2003.

[6.10-16] Henrich W et al, Data fusion for the new German F124 frigate: concepts and architecture, Proc ISIF 2003.

[6.10-18] M. Boman, M. and Van de Velde, W. Multi-Agent Rationality: Proceedings of the 8th European Workshop on Modelling Autonomous Agents in a Multi-Agent World, MAAMAW'97, Ronneby, Sweden, May 13-16, 1997.

[6.10-19] Bratman.M.E., Intentions, Plans, and Practical Reason. Harvard University Press: Cambridge, MA, 1987.

[6.10-20] Crick, F. Visual perception: rivalry and consciousness. Nature 379:485-486, 1996

[6.10-21] DeLoach, S.A., Matson, E.T., Li, Y.. Exploiting Agent Oriented Software Engineering in the Design of a Cooperative Robotics Search and Rescue System. The International Journal of Pattern Recognition and Artificial Intelligence, 17 (5), pp. 817-835, 2003.

[6.10-22] Hollywood, J., Snyder, D., McKay, K. N., and Boon, J.E. Out of the Ordinary: Finding Hidden Threats by Analyzing Unusual Behavior, Rand pub, ISBN: 0-8330-3520-7, 2004.

[6.10-23] Gazi, V., Moore, M.L., Passino, K.M., Shackleford, W.P., Proctor, F., Albus, J.S. The RCS Handbook: Tools for Real Time Control Systems Software Development, Wiley, 2001.

[6.10-24] Giorgini, P., Kolp, M., Mylopoulos, J., and Pistore, M. The Tropos Methodology: an overview. In F. Bergenti, M.-P. Gleizes and F. Zambonelli (Eds) Methodologies And Software Engineering For Agent Systems, Kluwer Academic Publishing, 2004.

[6.10-25] Healey, C. G., Booth, K. S., and Enns, J. T. Harnessing preattentive processes for multivariate data visualization. Proceedings Graphics Interface '93 (Toronto, Canada, 1993), pp. 107–117, 1993.

[6.10-26] Kinny, D. and M. Georgeff., M. Commitment and effectiveness of situated agents. In Proceedings of the Twelfth International Joint Conference on Artificial Intelligence (IJCAI-91), pages 82–88, Sydney, Australia, 1991.

[6.10-27] Muller, J.P. The Design of Intelligent Agents (LNAI Volume 1177). Springer-Verlag: Berlin,

[6.10-28] Rao, A.S. and Georgeff, M. Decision procedures of BDI logics. Journal of Logic and

[6.10-29] Richards, C. Certain To Win: The Strategy Of John Boyd, Applied To Business, Xlibris Corporation publisher, 2004.

[6.11-1] Sanjay Rawat, James Llinas, Christopher Bowman. Design of a Performance Evaluation Methodology for Data-Fusion Based Multi-Target Tracking Systems, Proc. Of the SPIE (5099), pp. 139-151, 2003.

[6.11-2] Erik Blasch, Fusion Metrics for Dynamic Situation Analysis, Proc. of the SPIE (5429) pp. 428-438, 2004.

[6.12-1] Bowman, C. L., "The Dual Node Network (Dual Node Network) Data Fusion & Resource Management (DF&RM) Architecture" AIAA Intelligent Systems Conference, Chicago, September 20-22, 2004

[6.12-2] Steinberg, A, Bowman, C. "Rethinking the JDL Data Fusion Levels", NSSDF Conference, JHAPL, June, 04

[6.12-3] Steinberg, Bowman, and White F., "Revisions to the JDL Model", Joint NATO/IRIS Conference Proceedings, Quebec, October, 1998.

[6.12-4] Bowman C. L., "The Data Fusion & Resource Management Dual Node Network (DF&RM Dual Node Network) Architecture: Application to Hercules MSF Distributed Architecture Problems" Hercules Blue Team Advanced Concepts Panel Presentation, June, 2000

[6.12-5] Bowman C. L., and A. S. Steinberg, Data Fusion Handbook: Chapter 18: Data Fusion Systems Engineering and Chapter 3: Revisions to the JDL Data Fusion Model, to appear in '01.

[6.12-6] Bowman, "Data Fusion" International Seminar, H. Silver and Associates, September, 1997.

[6.12-7] Steinberg and C. L. Bowman, " Development and Application of Data Fusion Engineering Guidelines" Proceedings of the National Symposium on Sensor and Data Fusion, MIT Lincoln Lab, Lexington, MA, April '97.

[6.12-8] Bowman, "Affordable Information Fusion Via an Open, Layered, Paradigm-Based Architecture", Proceedings of 9th NSSF, Monterey, CA, March '96.

[6.12-9] Llinas, B. Neuenfeldt, L. McConnell, D. Bohney, C. Bowman, D. Hall, et al, "Studies and Analyses Within Project Correlation: An In-Depth Assessment of Correlation Problems and Solution Techniques", Proceedings of 9th NSSF, Monterey, CA, March '96.

[6.12-10] Bowman and C. Morefield, "Multi-sensor Fusion of Target Attributes and Kinematic Reports", 3rd ONR/MIT Conference on C31, June 1980.

[6.12-11] Bowman "The Data Fusion Tree Paradigm and It's Dual" Proceedings of 7th National Symposium on Sensor Fusion, invited paper, Sandia Labs, NM, March '94

Appendix A: DIRE Source Code

Source code listings for selected elements of the DIRE simulation test bed are given in Appendix A. These listings include header .h and C++ .cpp files in two categories: Appendix A.1 contains the core source files required to instantiate the system and are required across all federates. A.2 contains the additional module-specific functional sources for one selected module, the Dispatch/Router Federate. Listing the complete source code here would occupy perhaps an additional thousand pages, thus a single representative federate has been selected. The goal of offering these listings is to illustrate how the interactive core of DIRE is configured, and how each federate's task-specific code is wrapped to interface with the federation and run properly within it. The code has been set in small type to conserve space. Electronic copies of the complete DIRE source code library is available from the authors upon request.

A.1 Common code

A.1.1 logParms.h

```
#ifndef h_PARAMETERSTOINDEPENDENTMESSAGELOGGINGTHREAD_001
#define h_PARAMETERSTOINDEPENDENTMESSAGELOGGINGTHREAD_001
#include <string>
#include <cassert>
#include "tsQueue.h"
// define a carrier-object to transmit the addresses of both
//     the the tsQueue and the filename that have been created.
// if the filename * == NULL, display will be to the CRT only
struct logParms
{
        tsQueue          *InteractionHighway;
        std::string          logfileName;
        logParms(tsQueue *queue, const std::string &name)
        {
                assert(queue != NULL);
                assert(name.size() > 0);
                InteractionHighway = queue;
                logfileName = name;
        }
        logParms(tsQueue *queue)
        {
                assert(queue != NULL);
                InteractionHighway = queue;
                logfileName = "";
        }
        ~logParms(void)
        {
                delete InteractionHighway;
        }
};
```

```
#endif
```

A.1.2 timeStamp.h

```cpp
#ifndef h_TIMESTAMPGENERATOR_0001
#define h_TIMESTAMPGENERATOR_0001
/////////////////////////////
// The debugger can't handle symbols more than 255 characters long.
// STL often creates symbols longer than that.
// When symbols are longer than 255 characters, the warning is issued.
#pragma warning(disable:4786)
/////////////////////////////
#include <string>
#include <ctime>
class timeStamp
{
public:
        timeStamp(void)
        {     }
        ~timeStamp(void)
        {     }
        std::string stamp(void)
        {
                time_t clock;
                struct tm *TimeDateStruct;
                char *TimeDateString;
                char dbNameString[26];
                time(&clock);
                TimeDateStruct = localtime(&clock);
                TimeDateString = asctime(TimeDateStruct);
                strncpy(dbNameString+0, TimeDateString+8, 2);
                strncpy(dbNameString+2, TimeDateString+4, 3);
                strncpy(dbNameString+5, TimeDateString+22, 2);
                *(dbNameString+7) = '_';
                strncpy(dbNameString+8, TimeDateString+11, 2);
                strncpy(dbNameString+10, TimeDateString+14, 2);
                strncpy(dbNameString+12, TimeDateString+17, 2);
                *(dbNameString+14) = '\0';
                return(dbNameString);
        }
};
#endif
```

A.1.3 tsQueue.h

```cpp
#ifndef h_CRITICALREGIONINFO_0001
#define h_CRITICALREGIONINFO_0001
#include <vector>
#include <string>
#include <queue>
//#include "typedefs.h"
class tsQueue : public std::queue<std::string> // Threadsafe access to data
{
public:
    tsQueue( )                                              // Constructor
        {
```

```cpp
            // Create the mutex which serializes all access to our data
            m_Mutex = CreateMutex( NULL, false, "SerializeAccess" );
            if ( m_Mutex == NULL )
            {
                                                            // This will crash the program - what
else to do?
                    exit(9);
            }
    }
    virtual ~tsQueue( )                                    // Destructor
    {
            CloseHandle(m_Mutex);                 // Clean up
    }

    void insert(const std::string & s)        // Add a member at the end
    {
            getAccessToData();                              // Check for non-destructive access
            this->push(s);                                  // Add data-element to vector
            m_TotalMemberCount += 1;              // add to total count
            if (m_HighWaterMark < this->queueSize())
            {
                    m_HighWaterMark = this->queueSize();
            }
            releaseAccessToData();                // Give up control of data
    }
    std::string extract(void)                 // Pull a member from the front
    {
            std::string retValue;                 // What we'll return to caller
            getAccessToData();                              // Check for non-destructive access
            if (!this->empty())                             // make sure something is there
            {
                    retValue = this->front();     // Grab requested element
                    this->pop();                            // remove element from queue
            }
            releaseAccessToData();                // Give up control of data
            return retValue;                                // Send removed value to caller
    }
    unsigned long queueSize(void)             // How many members in queue
    {
            unsigned long retValue;                       // What we'll return to caller
            getAccessToData();                              // Check for non-destructive access
            retValue = this->size();              // Get number of elements in vector
            releaseAccessToData();                // Give up control of data
            return retValue;                              // Send count to caller
    }
    unsigned long getHighWaterMark(void)
    {
            return m_HighWaterMark;
    }
    unsigned long getTotalMemberCount(void)
    {
            return m_TotalMemberCount;
    }
```

```cpp
private:
    HANDLE                  m_Mutex;             // Serialize access to this data
        unsigned long  m_HighWaterMark;      // instantaneous largest number of members
        unsigned long  m_TotalMemberCount;   // total number of insertions
    void getAccessToData(void)                        // Grab sole control
        {
                                                      // Request access to data
            WaitForSingleObject(m_Mutex, INFINITE);
        }
    void releaseAccessToData(void)           // Release sole control
        {
            ReleaseMutex(m_Mutex);                    // Release access to data
        }

        tsQueue(const tsQueue &dS)                    // copy constructor
        {
        // private member: cannot be copied
    }
        tsQueue &operator=(const tsQueue &dS)// assignment
        {
        // private member: cannot be assigned
    }
};
#endif
```

A.1.4 localFederate.h

```cpp
#ifndef h_LOCALFEDERATE_0001
#define h_LOCALFEDERATE_0001
#if defined(_MSC_VER) // if we're using Microsoft VC6
#define RTI_USES_STD_FSTREAM
#endif // defined(_MSC_VER)
#include <windows.h>
#include <RTI.hh>
#include "AllInteractions.h"  // All interaction classes are contained here
#include "interaction.h"
#include <iostream>         // for printout
#include <fstream>
#include <vector>
#include <string>
#include <map>
#include "baseTypes.hh"
#include <fedtime.hh>
#include "tsQueue.h"
#include "logParms.h"
#include <cassert>
//------------------------------------------------------------------
// type definitions
//------------------------------------------------------------------
typedef std::vector<std::string>      stringArray;
typedef stringArray::iterator         pString;
typedef std::pair<unsigned long, unsigned long>     twoCounts;
typedef std::pair<std::string, twoCounts>           labeledTwoCounts;
typedef std::map<std::string, twoCounts>            historyMap;
```

```
typedef historyMap::iterator                              pTwoCount;
void SyncReady(const unsigned long &duration);
void MessageLogger(LPVOID param);
//--------------------------------------------------------------------
//
// CLASS:
//      fedModel
//
// PURPOSE:
//      This class is the definition of the local federate simulation.
//
//--------------------------------------------------------------------
class fedModel
{
public:
        virtual ~fedModel()                                  // default dtor
        {        }

                                                             // our calls to the RTI
        void setAmbAddress(RTI::RTIambassador* rtiAmb)
        {
            m_rtiAmb = rtiAmb;           // initialize the pointer
        }
        void receiveInteraction(
                    RTI::InteractionClassHandle theInteraction, // supplied C1
                    const RTI::ParameterHandleValuePairSet &theParameters,  // supplied C4
                    const RTI::FedTime &theTime,                             // supplied C4
                    const char *theTag,                                      // supplied C4
                    RTI::EventRetractionHandle theHandle);     // supplied C1
        void Update( RTI::FedTime& newTime );
    //--------------------------------------------------------------
    // Accessor Methods
    //--------------------------------------------------------------
    RTI::FedTime const &getCurrentTime()
    {
            return m_CurrentTime;
    }
    RTI::FedTime const &getPreviousTime()
    {
            return m_PreviousTime;
    }
    RTI::FedTime const &getCurrentTimePlusLookahead()
    {
         m_TimePlusLookahead = m_CurrentTime;
      m_TimePlusLookahead += m_lookahead;
         return m_TimePlusLookahead;
    }
    const std::string getFederateName()
    {
            return m_FederateName;
    }
    RTI::FedTime &getLookahead()
    {
            return m_lookahead;
```

```cpp
    }
    unsigned long const &getUpdateTimeStep(void)
    {
            return m_UpdateTimeStep;
    }
    unsigned long &getRunDuration(void)
    {
            return m_RunDuration;
    }
    bool getPauseStatus(void)
    {
            return m_StartupPause;
    }
    std::string getInitParm(const std::string &name)
    {
            return m_MessageRGtoALL02.getValue(name);
    }
    twoCounts getReceiptCounters(std::string &IntName)
    {
            return (*(m_ReceiptCounters.find(IntName))).second;
    }
    twoCounts getSentCounters(std::string &IntName)
    {
            return (*(m_SentCounters.find(IntName))).second;
    }
    //------------------------------------------------------------------
    // Mutator Methods
    //------------------------------------------------------------------
    void  setCurrentTime( RTI::FedTime const & time )
    {
            m_CurrentTime = time;
    }
    void setPreviousTime( RTI::FedTime const & time )
    {
            m_PreviousTime = time;
    }
    void setUpdateTimeStep(const unsigned long &step)
    {
            m_UpdateTimeStep = step;
    }
    void setLookahead( RTI::FedTime& time )
    {
            m_lookahead = time;
    }
    void setFederateName(const std::string &name)
    {
            m_FederateName = name;
    }
private:
      unsigned long         m_UpdateTimeStep;
      std::string                 m_FederateName;        // Name of this federate
      RTI::RTIambassador *m_rtiAmb;          // Pointer to RTIambassador
      RTI::FedTime        &m_CurrentTime;    // Time of current update
```

```cpp
    RTI::FedTime      &m_PreviousTime;   // Time of previous update
    RTI::FedTime       &m_TimePlusLookahead;
    RTIfedTime          m_lookahead;     // Minimum time for action
    char                    m_SeparaterChar;
    unsigned long       m_RunDuration;
    bool                    m_StartupPause;       // true -> duration read on-line
    logParms                *m_MessageLoggerThreadParms;
    tsQueue                 *m_MessageLoggerQueue;
    DWORD                   m_MessageLoggerThreadID;
    HANDLE                  m_MessageLoggerThreadHandle;
    std::string             m_MessageLoggerFileName;
stringArray                 m_IntName;
stringArray                 m_IntParm;
stringArray                 m_CasualtyReport;

RGtoALL01              m_MessageRGtoALL01;
RGtoALL02              m_MessageRGtoALL02;
RTI::InteractionClassHandle    m_RGtoALL01_TypeId;
RTI::ParameterHandle           m_RGtoALL01_Parms_TypeId;
RTI::InteractionClassHandle    m_RGtoALL02_TypeId;
RTI::ParameterHandle           m_RGtoALL02_Parms_TypeId;
std::map<std::string, RTI::InteractionClassHandle>  m_TypeId;
std::map<std::string, RTI::ParameterHandle>         m_Parms_TypeId;
    stringArray             m_PubNames;          // names of publishable interactions
    stringArray             m_SubNames;          // names of subscribed-to interactions
    // These containers are for the purpose of counting interactions In and Out.
    // The counts are made available to the local federate for accounting purposes.
    //      The key-type of the map is a string (containing the interaction name).
    //      The value-type is a pair of unsigned long values;
    // pair.first is the count of receipts or sends of the named interaction
    //      during the most recent inter-federateUpdate interval.
    //      It is cleared to zero at the end of a federateUpdate (in preparation
    //      for the ensuing inter-federateUpdate interval).
    // pair.second is the total count of receipts or sends of the named interaction
    //      since the start of the simulation.
    //      It is cleared to zero at the beginning of a federateUpdate (in preparation
    //      for counting the sends during the federateUpdate).
    historyMap              m_ReceiptCounters;
historyMap              m_SentCounters;

    // Type counts are used to supply values for SrcDstCount (in msgHeader.h)
    // values are initialized to 0 in fedModel::Init
    // values are assigned to SrcDstCount in fedModel::sendMsgs
    std::map<std::string, unsigned long> m_IntCount;
    //------------------------------------------------------------------
    // function prototypes
    //------------------------------------------------------------------
    stringArray getPublicationNames(void);
    stringArray getSubscriptionNames(void);
    void fedUpdate(const unsigned long &n);
    void fedMessage(const std::string &id, std::string &data);

public:
```

```cpp
//------------------------------------------------------------------
//
// METHOD:
//      void fedModel(void)
//
// PURPOSE:
//      This method is the constructor for an object of the
//         fedModel class.
//
// RETURN VALUES:
//      None.
//
//------------------------------------------------------------------
fedModel( void )                                         // Constructor
        : m_CurrentTime(*(RTI::FedTimeFactory::makeZero()))
        , m_PreviousTime(*(RTI::FedTimeFactory::makeZero()))
        , m_TimePlusLookahead(*(RTI::FedTimeFactory::makeZero()))
        , m_StartupPause(false)
        , m_lookahead(1)                                 // minimum time in this federate
        , m_UpdateTimeStep(60)
        , m_SeparaterChar(',')
{
        logMessage("Local ctor startup.");
        m_IntName.push_back("RGtoDF01");    // Casualty Observation
        m_IntName.push_back("RGtoDF02");    // Medical Facility Damage
        m_IntName.push_back("RGtoDF03");    // Roadway Damage
        m_IntName.push_back("RGtoDF04");    // Casualty Pickup
        m_IntName.push_back("RGtoDF05");    // Casualty Delivery
        m_IntName.push_back("RGtoDF06");    // Police Location
        m_IntName.push_back("RGtoDF07");    // Ambulance Location
        m_IntName.push_back("RGtoDF08");    // Medical Facility Capacity
        m_IntName.push_back("RGtoDF09");    // Casualty Treatment Delay
        m_IntName.push_back("RGtoDF10");    // Ambulance Idle
        m_IntName.push_back("RGtoDF11");    // Ambulance Stuck
        m_IntName.push_back("RGtoDF12");    // Travel Delay
        m_IntName.push_back("RGtoDF13");    // Cluster Ident
        m_IntName.push_back("DFtoED01");    // Casualty observation
        m_IntName.push_back("DFtoED02");    // Medical facility damage
        m_IntName.push_back("DFtoED03");    // Roadway damage
        m_IntName.push_back("DFtoED04");    // Casualty pickup
        m_IntName.push_back("DFtoED05");    // Casualty delivery
        m_IntName.push_back("DFtoED06");    // Police location
        m_IntName.push_back("DFtoED07");    // Ambulance location
        m_IntName.push_back("DFtoED08");    // Medical facility capacity
        m_IntName.push_back("DFtoED09");    // Casualty Treatment Delay
        m_IntName.push_back("DFtoED10");    // Ambulance Idle
        m_IntName.push_back("DFtoED11");    // Ambulance Stuck
        m_IntName.push_back("DFtoED12");    // Travel Delay
        m_IntName.push_back("DFtoED13");    // Cluster Ident
        m_IntName.push_back("EDtoDP01");    // Casualty Observation
        m_IntName.push_back("EDtoDP04");    // Roadway Damage
        m_IntName.push_back("EDtoDP05");    // Medical Facility Capacity
        m_IntName.push_back("EDtoDP06");    // Travel Delay
```

```cpp
        m_IntName.push_back("EDtoDP07");      // Casualty Treatment Delay
        m_IntName.push_back("EDtoDP08");      // Ambulance Idle
        m_IntName.push_back("EDtoDP09");      // Ambulance Stuck
        m_IntName.push_back("EDtoDP10");      // Cluster Ident
        m_IntName.push_back("EDtoMF01");      // Medical Facility Damage
        m_IntName.push_back("EDtoMF02");      // Casualty Delivery
        m_IntName.push_back("MFtoRG01");      // Medical Facility Capacity
        m_IntName.push_back("MFtoRG02");      // Casualty Treatment Delay
        m_IntName.push_back("DPtoRG01");      // Ambulance Route
        m_IntName.push_back("DFtoL201");      // Casualty Observation
        m_IntName.push_back("DFtoL202");      // Casualty Pickup
        m_IntName.push_back("L2toRG01");      // Cluster Ident
        m_IntName.push_back("RGtoMF01");      // Hospital Location
        m_IntName.push_back("EDtoVZ01");      // Casualty Observation
        m_IntName.push_back("EDtoVZ02");      // Ambulance Location
        m_IntName.push_back("EDtoVZ03");      // Medical Facility Capacity
        m_IntName.push_back("EDtoVZ04");      // Cluster Ident
        m_IntName.push_back("EDtoVZ05");      // Medical Facility Damage
        m_IntName.push_back("EDtoVZ06");      // Roadway Damage
        m_IntName.push_back("EDtoVZ07");      // Police Location
        m_IntName.push_back("RGtoDP01");      // Hospital Location
        m_IntName.push_back("RGtoVZ01");      // Hospital Location
        m_IntName.push_back("RGtoL201");      // Casualty Estimates
        m_IntName.push_back("L2toED01");      // Viz Cluster Ident
        m_IntName.push_back("EDtoVZ08");      // Cluster Ident
        m_IntName.push_back("RGtoL202");      // Hospital Location

        std::string temp;
        for (pString i = m_IntName.begin(); i != m_IntName.end(); ++i)
        {
                temp = *i;
                temp += "Parms";
                m_IntParm.push_back(temp);
        }
        std::pair<std::string, RTI::InteractionClassHandle> it;
        std::pair<std::string, RTI::ParameterHandle>        ip;
        for (pString iCount = m_IntName.begin(); iCount != m_IntName.end(); ++iCount)
        {
                it.first = *iCount;
                it.second = m_rtiAmb->getInteractionClassHandle((*iCount).c_str());
                m_TypeId.insert(it);
                ip.first = *iCount;
                ip.second = m_rtiAmb->getParameterHandle((*iCount).c_str(), m_TypeId[*iCount]);
                m_Parms_TypeId.insert(ip);
        }
        for (pString n = m_IntName.begin(); n != m_IntName.end(); ++n)
        {
                m_IntCount.insert(std::pair<std::string, unsigned long>(*n, 0) );
        }
        logMessage("Local ctor complete.");
}
//------------------------------------------------------------------
//
```

```
// METHOD:
//     void Init(void)
//
// PURPOSE:
//     This method is reserved for the initialization of any
//         federate-specific variables placed into localFederate.h
//         by the user/federate-builder.
//
// RETURN VALUES:
//     None.
//
//----------------------------------------------------------------
void Init(void)
{
}
//----------------------------------------------------------------
//
// METHOD:
//     void Terminate(void)
//
// PURPOSE:
//     This method is reserved for end-of-run processing.
//             This function is called by modelMain and should NOT
//             be called by the localFederate.
//         This is where the modeler will clean up any allocated
//             data and prepare for termination of the localFederate.
//         The proper way for the local federate to terminate is
//             to throw an exception.
//
// RETURN VALUES:
//     None.
//
//----------------------------------------------------------------
void fedModel::Terminate(void)
{
}// terminate
//----------------------------------------------------------------
//
// METHOD:
//     void PublishAndSubscribe(void)
//
// PURPOSE:
//     This method conveys to the federation the names of the
//             interactions which this federate will receive.
//
// RETURN VALUES:
//     None.
//
//----------------------------------------------------------------
void PublishAndSubscribe(void)
{
        logMessage("Publish and Subscribe Starting.");
        m_SubNames = getSubscriptionNames();
```

```
            m_PubNames = getPublicationNames();
            try
            {
                if ( m_rtiAmb )
                {
                    m_RGtoALL01_TypeId = m_rtiAmb->getInteractionClassHandle("RGtoALL01");
                    m_RGtoALL01_Parms_TypeId    =    m_rtiAmb->getParameterHandle("RGtoALL01Parms",
m_RGtoALL01_TypeId);
                    m_rtiAmb->subscribeInteractionClass( m_RGtoALL01_TypeId );
                    m_RGtoALL02_TypeId = m_rtiAmb->getInteractionClassHandle("RGtoALL02");
                    m_RGtoALL02_Parms_TypeId    =    m_rtiAmb->getParameterHandle("RGtoALL02Parms",
m_RGtoALL02_TypeId);
                    m_rtiAmb->subscribeInteractionClass( m_RGtoALL02_TypeId );
                    for (pString pubName = m_PubNames.begin(); pubName != m_PubNames.end(); ++pubName)
                    {
                        m_rtiAmb->publishInteractionClass(m_TypeId[*pubName]);
                    }
                    for (pString subName = m_PubNames.begin(); subName != m_PubNames.end(); ++subName)
                    {
                        m_rtiAmb->subscribeInteractionClass(m_TypeId[*subName]);
                    }
                }
            }
            catch(RTI::InteractionClassNotDefined &e)
            {
                std::string out = "Interaction Class Not Defined: ";
                out += e._reason;
                logMessage(out);
                throw;
            }
            catch(RTI::RTIinternalError &e)
            {
                std::string out = "RTI Internal Error: ";
                out += e._reason;
                logMessage(out);
                throw;
            }
            logMessage("Publish and Subscribe Ending.");
        }// PublishAndSubscribe
private:
        //------------------------------------------------------------
        //
        // METHOD:
        //      bool parse(stringArray &tokens,
        //                                       char *rawData)
        //
        // PURPOSE:
        //              Dissect the rawData char-array into text tokens separated by
        //              the separaterChar character.
        //
        // RETURN VALUES:
        //      void
        //
```

AFOSR F49620-01-1-0371

```cpp
//-------------------------------------------------------------
void parse(stringArray &tokens, char *rawData)
{
        tokens.clear();                                         // empty the container
        std::string catcher;                           // catch chars of token
        unsigned int tokenCounter = 0;              // count tokens returned
        int index = 0;                                      // count chars of input area
        while (rawData[index] != '\0')              // skim the input area
        {
                if (rawData[index] == m_SeparaterChar)
                {
                        tokens.push_back(catcher);    // save the token
                        catcher = "";                               // clear the character-catcher
                        if (rawData[index+1]==' ')    // exclude leading blank-chars
                        {
                                while (rawData[++index]==' ');
                                index -= 1;
                        }
                }
                else
                {
                        if ('"' != rawData[index])    // exclude VB quote-chars
                        {
                                catcher += rawData[index];// catch another character
                        }
                }
                index += 1;                                     // move to next character
        }
        if (catcher.size() > 0)
        {
                tokens.push_back(catcher);          // save terminal token
        }
}


//-------------------------------------------------------------
//
// METHOD:
//          void sendMsgs(parmType *parmlist)
//                   where parmlist is an interaction such as RGtoDF01
//
// PURPOSE:
//          Send sendInteraction to all subscribing federates.
//
// RETURN VALUES:
//     void
//
//-------------------------------------------------------------
template<class parmType>
void sendMsgs(parmType *parmlist)
{
        std::string sourcedest = parmlist->getIntName();
        std::string msg = "Sending interaction - sourcedest = ";
        msg += sourcedest;
```

AFOSR F49620-01-1-0371

```
        logMessage(msg);
        interaction<parmType> *Interaction = new interaction<parmType>(*parmlist);
        RTI::ParameterHandleValuePairSet *pParams = RTI::ParameterSetFactory::create(1);
        RTI::InteractionClassHandle TypeID;
        try
        {
                TypeID = m_rtiAmb->getInteractionClassHandle(sourcedest.c_str());
        }
        catch(RTI::NameNotFound e)
        {
                throw "Name Not Found";
        }
        catch(RTI::RTIinternalError e)
        {
                throw "RTI Internal Error";
        }
        std::string IntParm = sourcedest;
        IntParm += "Parms";
        RTI::ParameterHandle Parms_TypeID;
        try
        {
                Parms_TypeID = m_rtiAmb->getParameterHandle(IntParm.c_str(), TypeID);
        }
        catch(RTI::InteractionClassNotDefined e)
        {
                throw "Interaction Class Not Defined";
        }
        catch(RTI::NameNotFound e)
        {
                throw "Name Not Found";
        }
        catch(RTI::RTIinternalError e)
        {
                throw "RTI Internal Error";
        }
        Interaction->setSrcDstCount(++(m_IntCount[sourcedest.c_str()]));
        try
        {
                pParams->add(Parms_TypeID, (char *)((Interaction->GetASCII()).c_str()),
                                    ((strlen(((Interaction->GetASCII()).c_str()))+1) * sizeof(char)));
        }
        catch(RTI::ValueLengthExceeded e)
        {
                throw "Value Length Exceeded";
        }
        catch(RTI::ValueCountExceeded e)
        {
                throw "Value Count Exceeded";
        }
//      std::string msg = "--> Attaching parms ";
//      msg += Parms_TypeID;
//      msg += " to interaction";
//      logMessage(msg);
```

```
                      msg = "----> Parms text: ";
                      msg += (char *)((Interaction->GetASCII()).c_str());
                      logMessage(msg);
//       std::string msg = "------> Now send ";
//       msg += TypeID;
//       logMessage(msg);
//       std::string msg = "------> Timestamp = ";
//       msg += this->GetLastTimePlusLookahead();
//       msg += " to interaction";
//       logMessage(msg);
                      // make sure that timestamp isn't in the past
                      char dummy[65];
                      char *timeStamp = _ltoa(parmlist->getTime(), dummy, 10);
                      RTI::FedTime &outTime = (*(RTI::FedTimeFactory::decode(timeStamp)));
                      if (outTime < this->GetLastTimePlusLookahead())
                      {
                              outTime = this->GetLastTimePlusLookahead();
                      }
                      try
                      {
                              (void)m_rtiAmb->sendInteraction(TypeID, *pParams, outTime, NULL);
                      }
                      catch(RTI::InteractionClassNotDefined e)
                      {
                              throw "Interaction Class Not Defined";
                      }
                      catch(RTI::InteractionClassNotPublished e)
                      {
                              throw "Interaction Class Not Published";
                      }
                      catch(RTI::InteractionParameterNotDefined e)
                      {
                              throw "Interaction Parameter Not Defined";
                      }
                      catch(RTI::RTIinternalError e)
                      {
                              throw "RTI Internal Error";
                      }
//       logMessage("--------> Interaction Sent");
                      logSend(parmlist);
                      delete pParams;
                      delete Interaction;
              } // sendMsgs
public:
        //----------------------------------------------------------------
        //
        // METHOD:
        //      void logMessage(const std::string &txtMsg)
        //
        // PURPOSE:
        //              Receives a text-message and sends it to a background thread
        //                      that displays it on the console screen.
        //
```

```cpp
        // RETURN VALUES:
        //     None.
        //
        //----------------------------------------------------------------
        void fedModel::logMessage(const std::string &txtMsg)
        {
                if (m_MessageLoggerQueue != NULL)
                {
                        m_MessageLoggerQueue->insert(txtMsg);
                }
                else
                {
                        std::cout << txtMsg << std::endl;
                }
        }
private:
        //----------------------------------------------------------------
        //
        // METHOD:
        //     void logReceipt(msgHeader &msg)
        //
        // PURPOSE:
        //             Logs the receipt of an interaction and sends the log to
        //                  a background thread   that displays it on the console screen.
        //
        // RETURN VALUES:
        //     None.
        //
        //----------------------------------------------------------------
        void fedModel::logReceipt(msgHeader *msg)
        {
               char dummy[65];
               std::string out =  "   Received ";
               out += (msg->getSourceDestination()).c_str();
               out += ", timestamp = ";
               out += _ltoa(msg->getTime(), dummy, 10);
               logMessage(out);
        }
        //----------------------------------------------------------------
        //
        // METHOD:
        //     void logSend(msgHeader *msg)
        //
        // PURPOSE:
        //             Logs the sending of an interaction and sends the log to
        //                  a background thread   that displays it on the console screen.
        //
        // RETURN VALUES:
        //     None.
        //
        //----------------------------------------------------------------
        void fedModel::logSend(msgHeader *msg)
        {
```

-A-15-

```
            RTI::FedTime *DisplayTime = RTI::FedTimeFactory::makeZero();
            *DisplayTime = this->getCurrentTimePlusLookahead();
            char Now[255];
            DisplayTime->getPrintableString(Now);
            std::string out = "   Sending ";
            out += (msg->getSourceDestination()).c_str();
            out += " for delivery at ";
            out += Now;
            logMessage(out);
            delete DisplayTime;
        }
};


#endif
```

A.1.5. **ModelMain.cpp**

```
/////////////////////////////
// The debugger can't handle symbols more than 255 characters long.
// STL often creates symbols longer than that.
// When symbols are longer than 255 characters, the warning is issued.
#pragma warning(disable:4786)
/////////////////////////////
//#include <windows.h>
#include "localFederate.h"
#include "HwFederateAmbassador.hh"
#include "timeStamp.h"
#include <RTI.hh>
#include <fedtime.hh>
#include <sys/timeb.h>        // for "struct _timeb"
#include <ctime>
#include <iostream>                  // for printout
#include <cassert>
using namespace std;
// Global Variables - used here and in HwFederateAmbassador.cpp
//                  - (NOT an example to be emulated)
RTI::Boolean  timeAdvGrant = RTI::RTI_FALSE;
RTI::Boolean  TimeRegulation = RTI::RTI_FALSE;
RTI::Boolean  TimeConstrained = RTI::RTI_FALSE;
RTI::FedTime &grantTime = (*(RTI::FedTimeFactory::makeZero()));
static RTIfedTime endTime;                              // run-termination time
static bool WaitingForStart;
fedModel *localFederate = NULL;                    // pointer to our federate
timeStamp *clicker = NULL;                          // point to useful support-object
int hw_main(int argc, char *argv[])          // "main" is at the end of this module
{
        string out;                                          // used for generating console messages
        WaitingForStart = true;                          // don't begin until "start" signal
        //----------------------------------------------------
        // localFederate construction
        //----------------------------------------------------
        localFederate = new fedModel();              // create our federate-object
        const string federationName = "IFD";// Name of the Federation
        string localName = localFederate->getFederateName();// Name of this federate
        try
```

```
        {
            //-------------------------------------------------------
            // Create RTI objects
            //
            // The federate communicates to the RTI through the RTIambassador
            // object and the RTI communicates back to the federate through
            // the FederateAmbassador object.
            //-------------------------------------------------------
            RTI::RTIambassador      rtiAmbassador;      // libRTI provided
            HwFederateAmbassador    fedAmbassador;      // defined by the federate
            RTI::FederateHandle     federateId; // used for self-reference
            RTI::Boolean Joined = RTI::RTI_FALSE;
            int numTries  = 0;
            //-------------------------------------------------------
            // Here we loop around the joinFederationExecution call
            // until we try too many times or the Join is successful.
            //
            // The federate that successfully CREATES the federation
            // execution will get to the join call before the
            // FedExec is initialized and will be unsuccessful at
            // JOIN call.  In this loop we catch the
            // FederationExecutionDoesNotExist exception to
            // determine that the FedExec is not initialized and to
            // keep trying. If the JOIN call does not throw an
            // exception then we set Joined to TRUE and that will
            // cause us to exit the loop and proceed in the execution.
            // The loop repeats only if a FederationExecutionDoesNotExist
            // exception was thrown.  Since the RTI 1.3 specification
            // has the inherent race condition that another process
            // could have destroyed the federation after this process
            // calls create, we need to create the federation each
            // time through this loop.
            //-------------------------------------------------------
            while( !Joined && (numTries++ < 20) )
            {
                try
                {
                    out = "Creating federation execution for ";
                    out += federationName;
                    localFederate->logMessage(out);
                    string fedFileName = localName;
                    fedFileName += ".fed";
                    rtiAmbassador.createFederationExecution(                    federationName.c_str(),
fedFileName.c_str() );
                    localFederate->logMessage("Successful federation creation!");
                }
                catch ( RTI::FederationExecutionAlreadyExists& e )
                {
                    out = "Note: Federation ";
                    out += federationName;
                    out += " execution already exists.";
                    out += e._reason;
                    localFederate->logMessage(out);
```

-A-17-

```
            }
            catch ( RTI::Exception& e )
            {
                    out = "RTI Error: ";
                    out += e._reason;
                    localFederate->logMessage(out);
                    throw 101;                               // don't continue local federate
            }
            try
            {
                    out = localName.c_str();
                    out += " joining federation execution.";
                    localFederate->logMessage(out);
                                                             // Join the named federation
execution
                    federateId    =    rtiAmbassador.joinFederationExecution(    localName.c_str(),
federationName.c_str(), &fedAmbassador);
                    Joined = RTI::RTI_TRUE;           // mark attempt successful
            }
            catch (RTI::FederateAlreadyExecutionMember& e)
            {
                    out = "Error: ";
                    out += localName;
                    out += " already exists in Federation ";
                    out += federationName;
                    out += ".";
                    localFederate->logMessage(out);
                    out = e._reason;
                    localFederate->logMessage(out);
                    throw 201;                               // don't continue local federate
            }
            catch (RTI::FederationExecutionDoesNotExist&)
            {
                    out = "Error: ";
                    out += federationName ;
                    out += "  - Federation Execution does not exist.";
                    localFederate->logMessage(out);
                    rtiAmbassador.tick(2.0, 2.0);
            }
            catch ( RTI::Exception& e )
            {
                    out = "RTI Error: ";
                    out += e._reason;
                    localFederate->logMessage(out);
                    throw 202;                               // don't continue local federate
            }
        } // end while
        // at this point we have joined the federation, one way or the other
        out = localName;                                     // name of this federate
        out += " joined successfully to federation:";
        localFederate->logMessage(out);
        localFederate->setAmbAddress( &rtiAmbassador );
        //----------------------------------------------------
```

```
// localFederate initialization
//-------------------------------------------------------
try
{
        localFederate->Init( );                         // perform federate initialization
}
catch(string e)
{
        localFederate->logMessage(e);
        throw 301;                                      // don't continue local federate
}
catch(...)
{
        localFederate->logMessage("Federate cannot proceed due to initialization error.");
        throw 302;                                      // don't continue local federate
}
//-------------------------------------------------------
// Publication/Subscription
//
// Declare my interests to the RTI for the object and
// interaction data types I want to receive.
//-------------------------------------------------------
try
{
        localFederate->PublishAndSubscribe();
}
catch(...)
{
        localFederate->logMessage("Federate cannot proceed due to Publish/Subscribe error.");
        throw 401;                                      // don't continue local federate
}
const RTIfedTime timeStep(localFederate->getUpdateTimeStep());
//-------------------------------------------------------
// local time advances only at the discretion of the RTI
// timeAdvGrant will remain false until a requested advance
// is granted
//-------------------------------------------------------
timeAdvGrant = RTI::RTI_FALSE;
TimeConstrained = RTI::RTI_FALSE;
//-------------------------------------------------------
// Set the Time Management parameters
//
// This version of EstimateDirector operates as a time-stepped
// simulation.  This means that it should be constrained
// and regulating.
//-------------------------------------------------------
try
{
        //-------------------------------------------------------
        // Turn on constrained status so that regulating
        // federates will control our advancement in time.
        //-------------------------------------------------------
        localFederate->logMessage("Enabling time-constrained.");
```

AFOSR F49620-01-1-0371

```
                    rtiAmbassador.enableTimeConstrained();
                    timeAdvGrant = RTI::RTI_FALSE;
                    TimeConstrained = RTI::RTI_FALSE;
                    //------------------------------------------------------
                    // Tick the RTI until we get the timeConstrainedEnabled
                    // callback with my current time. timeConstrainedEnabled()
                    // will set both timeAdvGrant and TimeConstrained to true
                    //------------------------------------------------------
                    while ( !TimeConstrained )
                    {
                            rtiAmbassador.tick(0.01, 1.0);
                    }
                    localFederate->logMessage("Time-constrained is enabled.");
            }
            catch ( RTI::Exception& e )
            {
                    out = "Error:";
                    out += e._reason;
                    localFederate->logMessage(out);
                    throw 501;                                      // don't continue local federate
            }
            try
            {
                    out = "Enabling time regulation with lookahead = ";
                    char        *PrintableLookahead      =        new        char        [(localFederate-
>getLookahead()).getPrintableLength()];
                    (localFederate->getLookahead()).getPrintableString(PrintableLookahead);
                    out += PrintableLookahead;
                    delete [] PrintableLookahead;
                    localFederate->logMessage(out);
                    //------------------------------------------------------
                    // Turn on regulating status so that constrained
                    // federates will be controlled by our time.
                    //
                    // If we are regulating and our local federate interactions
                    // are specified with timestamp in the EstimateDirector.fed
                    // file we will send Time Stamp Ordered (TSO) messages.
                    //------------------------------------------------------
                    out = "Attempt to enableTimeRegulation starting at time = ";
                    char *PrintableRegulation = new char [grantTime.getPrintableLength()];
                    grantTime.getPrintableString(PrintableRegulation);
                    out += PrintableRegulation;
                    delete [] PrintableRegulation;
                    localFederate->logMessage(out);
                    rtiAmbassador.enableTimeRegulation( grantTime, localFederate->getLookahead());
                    //------------------------------------------------------
                    // enableTimeRegulation is an implicit timeAdvanceRequest
                    // so set timeAdvGrant to TRUE since we will get a
                    // timeRegulationEnabled which is an implicit
                    // timeAdvanceGrant
                    //------------------------------------------------------
                    timeAdvGrant = RTI::RTI_FALSE;
                    TimeRegulation = RTI::RTI_FALSE;
```

```cpp
        //-------------------------------------------------------
        // Tick the RTI until we get the timeRegulationEnabled
        // callback with my current time.
        // timeRegulationEnabled will set
        //  timeAdvGrant = true      and
        //  TimeRegulation = true
        //-------------------------------------------------------
        while ( !TimeRegulation )
        {
                rtiAmbassador.tick(0.01, 1.0);
        }
}
catch ( RTI::Exception& e )
{
        out = "Error:";
        out += e._reason;
        localFederate->logMessage(out);
        throw 601;                                              // don't continue local federate
}
//-------------------------------------------------------
// Event Loop
// ----------
//
// 1.) Calculate current state of local federate.
// 2.) Ask for a time advance.
// 3.) Tick the RTI waiting for the grant and process all
//     RTI initiated services.
// 4.) Repeat.
//-------------------------------------------------------
RTIfedTime requestTime(0);            // time of first update
string out;
char dummy[65];
clicker = new timeStamp();
unsigned long counter = 0;            // count the number of our updates
endTime = grantTime;
while ( grantTime <= endTime )
{
        out = "Event Loop Iteration #: ";
        out += _ltoa(counter, dummy, 10);
        out += " begin at time ";
        out += clicker->stamp();
        localFederate->logMessage(out);              // log the time
        //-------------------------------------------------------
        // Update current state of this federate at the local time
        //-------------------------------------------------------
        char Now[255];
        grantTime.getPrintableString(Now);
        out = "(Line-";
        out += _itoa(__LINE__, dummy, 10);
        out += ")";
        out += "Begin local federate Update at time = ";
        out += Now;
        localFederate->logMessage(out);
```

```
                counter += 1;                          // count the iterations
                out = "Event Loop Iteration #: ";
                out += _ltoa(counter, dummy, 10);
                out += " start at time ";
                out += clicker->stamp();
                localFederate->logMessage(out);            // log the time
                localFederate->Update( grantTime );   // Perform the update
                out = "Update end at time ";
                out += clicker->stamp();
                localFederate->logMessage(out);            // log the time

                //------------------------------------------------------
                // Ask for a time advance to the next event.
                //------------------------------------------------------
                requestTime = timeStep.getTime();
                requestTime += grantTime;// request the next tick
                out = "(Line-";
                out += _itoa(__LINE__, dummy, 10);
                out += ")";
                out += "Request time advance to ";
                out += _gcvt(((double)requestTime.getTime()), 16, dummy);
                localFederate->logMessage(out);
                timeAdvGrant = RTI::RTI_FALSE;
                try
                {
                        rtiAmbassador.nextEventRequest( requestTime );
//cout << "Waiting for startup-signal.";
                        while (WaitingForStart)
                        {
                                Sleep(2000);
                                rtiAmbassador.tick(0.01, 1.0);
//cout << ".";
                        }
//cout << endl;
                }
                catch ( RTI::Exception& e )
                {
                        out = "Error:";
                        out += e._reason;
                        localFederate->logMessage(out);
                        throw 701;                              // don't continue local federate
                }
                //------------------------------------------------------
                // Tick the RTI waiting for the time-grant and process all
                //    RTI initiated interactions.
                //------------------------------------------------------

                while( timeAdvGrant != RTI::RTI_TRUE )
                {
                        //------------------------------------------------------
                        // Tick will turn control over to the RTI so that it can
                        // process an event.  This will cause an invocation of one
                        // of the federateAmbassadorServices methods.
```

-A-22-

```
                    //
                    // NOTE:
                    // Be sure not to invoke the RTIambassadorServices from the
                    // federateAmbassadorServices; otherwise, a ConcurrentAccess
                    // exception will be thrown.
                    //------------------------------------------------------
                    rtiAmbassador.tick(0.01, 1.0);
            }//end while
            // we have been granted an advance to the next requested time
            {
                    // this following code is special-purpose and enables the
                    //     test-scripter to request an orderly federate termination.
                    string QuitFlag = "quitflag";
                    ifstream *AbortFederate = new ifstream(QuitFlag.c_str(), ios::in);
                    if (AbortFederate->is_open())                 // if file exists
                    {
                            AbortFederate->close();                               // close the file
                            delete AbortFederate;                     // clean up
                            AbortFederate = NULL;                     // clear the pointer
                            system("DEL quitflag");                        // clean up the drive
                            break;                                                   //      leave
execute-loop
                    }
            }
    }//end while
    try
    {
            rtiAmbassador.disableTimeConstrained();
            TimeConstrained = RTI::RTI_FALSE;
    }
    catch(RTI::Exception& e)
    {
            out = "Error:";
            out += e._reason;
            localFederate->logMessage(out);
    }
    try
    {
            rtiAmbassador.disableTimeRegulation();
            TimeRegulation = RTI::RTI_FALSE;
    }
    catch(RTI::Exception& e)
    {
            out = "Error:";
            out += e._reason;
            localFederate->logMessage(out);
    }
    //------------------------------------------------------
    // Resign from the federation execution to remove this
    // federate from participation.  The flag provided
    // will instruct the RTI to call deleteObjectInstance
    // for all objects this federate has privilegeToDelete
    // for (which by default is all objects that this federate
```

-A-23-

```
        // registered) and to release ownership of any attributes
        // that this federate owns but does not own the
        // privilegeToDelete for.
        //-------------------------------------------------------
        try
        {
                localFederate->logMessage("Resign-federation execution called");

                rtiAmbassador.resignFederationExecution(RTI::DELETE_OBJECTS_AND_RELEASE_ATTRIBUTES);
                localFederate->logMessage("Successful resign-federation execution called.");
        }
        catch ( RTI::Exception& e )
        {
                out = "Error:";
                out += e._reason;
                localFederate->logMessage(out);
        }
        //-------------------------------------------------------
        // Destroy the federation execution in case we are the
        // last federate. This will not do anything bad if there
        // other federates joined.  The RTI will throw us an
        // exception telling us that other federates are joined
        // and we can just ignore that.
        //-------------------------------------------------------
        try
        {
                localFederate->logMessage("Destroy federation execution called");
                rtiAmbassador.destroyFederationExecution(federationName.c_str());
                localFederate->logMessage("Successful destroy federation execution called.");
        }
        catch ( RTI::FederatesCurrentlyJoined &e)
        {
                localFederate->logMessage(e._reason);
        }
        catch ( RTI::FederationExecutionDoesNotExist &e)
        {
                localFederate->logMessage(e._reason);
        }
        catch ( RTI::Exception& e )
        {
                out = "Error:";
                out += e._reason;
                localFederate->logMessage(out);
        }
}// end try
catch (RTI::ConcurrentAccessAttempted& e)
{
        out = "Error: Concurrent access to the RTI was attemted.\n";
        out += "        Exception caught in main() - PROGRAM EXITING.\n";
        out += "\n";
        out += "Note:  Concurrent access will result from invoking\n";
        out += "        RTIambassadorServices within the scope of\n";
        out += "        federateAmbassadorService invocations.\n";
```

AFOSR F49620-01-1-0371

```
                localFederate->logMessage(out);
                out = "Error:";
                out += e._reason;
                localFederate->logMessage(out);
                throw 901;                                              // don't continue local federate
        }
        catch ( RTI::Exception& e )
        {
                out = "Error:";
                out += e._reason;
                localFederate->logMessage(out);
                throw 902;                                              // don't continue local federate
        }
        catch(string e)
        {
                localFederate->logMessage(e);
                throw 903;                                              // don't continue local federate
        }
        out = "Exiting ";
        out += localName;
        out += ".";
        localFederate->logMessage(out);
        localFederate->Terminate();                          // do user-defined cleanup
        delete clicker;
        delete localFederate;
        localFederate = NULL;
        return 0;
}
void SyncReady(const unsigned long &duration)
{
        endTime = duration;   // value received (in interaction) at localFederate
        WaitingForStart = false;
        char dummy[65];
        string out = "Synchronized execution-start beginning.";
        out += "\n";
        out += "Run duration = ";
        out += _ltoa(duration, dummy, 10);
        localFederate->logMessage(out);
}
int main(int argc, char** argv)
{
        int retcode = 0;
        try
        {
                hw_main(argc, argv);
        }
        catch(int abortCode)                                    // get here if local federate aborted
        {
                retcode = abortCode;
                char dummy[65];
                if (localFederate != NULL)
                {
                        string out = "Execution terminated; Code = ";
```

```
                                out += _itoa(abortCode, dummy, 10);
                                localFederate->logMessage(out);
                                localFederate->Terminate();            // do user-defined cleanup
                                delete clicker;
                                delete localFederate;
                                localFederate = NULL;
                        }
            }
            // get here if local federate terminated normally
            return retcode;                                            // either way, quit
}
```

6. `localFederate.cpp`
```
/////////////////////////////
// The debugger can't handle symbols more than 255 characters long.
// STL often creates symbols longer than that.
// When symbols are longer than 255 characters, the warning is issued.
#pragma warning(disable:4786)
/////////////////////////////
#include "localFederate.h"
//#include <winsock2.h>
#include <iostream>             // for printout
#include <string>               // for interactions
#include <vector>               // for interactions
#include <climits>              // for INT_MAX
#include <stdlib.h>             // for _itoa
#include <stddef.h>
#include <stdio.h>
#include <cassert>
using namespace std;
//------------------------------------------------------------------
//
// METHOD:
//      void fedModel::Update( RTIfedTime& newTime )
//
// PURPOSE:
//      Update the state of the federate based on the new time
//      value.  The deltaTime is calculated based on the last
//      time the federate was updated and the newTime passed in.
//
// RETURN VALUES:
//      None.
//
//------------------------------------------------------------------
void fedModel::Update( RTI::FedTime& newTime )
{
        char PrintableTime[255];
        newTime.getPrintableString(PrintableTime);
        string msg = "Federate update begun at ";
        msg += PrintableTime;
        logMessage(msg);

                                                        // Save time of previous update
        this->setPreviousTime( this->getCurrentTime() );
```

-A-26-

```
        this->setCurrentTime(newTime);                  // Establish time now
        RTIfedTime now = getCurrentTime();
        // Prepare for this update by clearing the count
        //      of interactions sent during the previous update
        //      (total counts remain undisturbed)
        for (pTwoCount sent = m_SentCounters.begin(); sent != m_SentCounters.end(); ++sent)
        {
                (sent->second).first = 0;                       // clear the past interval's count
        }

        // here we will make the Update available to the local federate
        unsigned long federateTime = now.getTime();
        this->fedUpdate(federateTime);
        // Complete this update by clearing the count
        //      of interactions received since the previous update
        //      (total counts remain undisturbed)
        for (pTwoCount received = m_ReceiptCounters.begin(); received != m_ReceiptCounters.end(); ++received)
        {
                (received->second).first = 0;            // clear the past interval's count
        }
        logMessage("Update is Complete.");
    return;                         // return to modelMain
}
//---------------------------------------------------------------
//
// METHOD:
//      void fedModel::receiveInteraction(
//                              RTI::InteractionClassHandle theInteraction,
//                              const RTI::ParameterHandleValuePairSet &theParameters,
//                              const RTI::FedTime &theTime,
//                              const char *theTag,
//                              RTI::EventRetractionHandle theHandle)
//
// PURPOSE:
//      Update the state of the federate based on the contents of the
//          interaction. The timestamp on the interaction is guaranteed
//          to be >= the most recent invocation of Update (above)
//          and <= the next invocation of Update.
//
// RETURN VALUES:
//      None.
//
//---------------------------------------------------------------
void fedModel::receiveInteraction(
                            RTI::InteractionClassHandle theInteraction,
                            const RTI::ParameterHandleValuePairSet &theParameters,
                            const RTI::FedTime &theTime,
                            const char *theTag,
                            RTI::EventRetractionHandle theHandle)
{
        char Now[255];
        // The following message produces a record of the interaction-receipt
        RTI::FedTime *pTime = RTI::FedTimeFactory::makeZero();
```

```cpp
        (*pTime) = theTime;
    pTime->getPrintableString(Now);
    delete pTime;
    string out = "Interaction received at time ";
    out += Now;
    logMessage(out);
    // Get parameter string from theParameters
    unsigned long valueLength = 0;
    char *rawData;
    rawData = new char [theParameters.getValueLength(0)];
    theParameters.getValue(0, rawData, valueLength);
    string xmitData = rawData;                      // Convert the char[] to a std::string
    delete [] rawData;                                      // clean up after ourselves

for (pString iCount = m_IntName.begin(); iCount != m_IntName.end(); ++iCount)
{
        if (theInteraction == m_TypeId[*iCount])
        {
                if (theParameters.getHandle(0) != m_Parms_TypeId[*iCount])
                {
                            logMessage("Interaction / parameter mismatch.");
                            return;
                }
                else
                {
                    string IntName = xmitData.substr(0, xmitData.find("~"));
                    pTwoCount pIntCounters = m_ReceiptCounters.find(IntName);
                    if (pIntCounters != m_ReceiptCounters.end())
                    {
                            (pIntCounters->second).first += 1;   // increment total count
                            (pIntCounters->second).second += 1;   // increment count this interval
                    }
                    else
                    {
                            twoCounts start(1,1);
                            labeledTwoCounts tagged(IntName, start);
                            m_ReceiptCounters.insert(tagged);
                    }
                    // At this point we must make the received interaction
                    //     available to the local federate.
                    for (pString subName = m_PubNames.begin(); subName != m_PubNames.end(); ++subName)
                    {
                            if (*subName == *iCount)
                            {
                                    fedMessage(*iCount, xmitData);
                                    return;
                            }
                    }
                }
        }
    }
if (theInteraction == m_RGtoALL01_TypeId)
{
```

```
        if (theParameters.getHandle(0) != m_RGtoALL01_Parms_TypeId)
        {
                logMessage("Interaction / parameter mismatch.");
                return;
        }
        else
        {
                interaction<RGtoALL01> *FederationStartup = new interaction<RGtoALL01>(xmitData);
                m_MessageRGtoALL01 = FederationStartup->GetContents();
                m_RunDuration = m_MessageRGtoALL01.getDuration();
                SyncReady(m_RunDuration);
                char dummy[65];
                string out = "LOG:Startup msg received - duration = ";
                out += _itoa(m_MessageRGtoALL01.getDuration(), dummy, 10);
                logMessage(out);
                delete FederationStartup;
                return;
        }
    }

    if (theInteraction == m_RGtoALL02_TypeId)
    {
        if (theParameters.getHandle(0) != m_RGtoALL02_Parms_TypeId)
        {
                logMessage("Interaction / parameter mismatch.");
                return;
        }
        else
        {
                interaction<RGtoALL02> *FederationInitialization = new interaction<RGtoALL02>(xmitData);
                m_MessageRGtoALL02 = FederationInitialization->GetContents();
                // here is where we can access any or all of the ini-file parms
                m_MessageLoggerFileName = m_MessageRGtoALL02.getValue("ScreenLog");
                delete FederationInitialization;
                // create a queue to communicate received interactions
                //     to the background thread that saves them
                m_MessageLoggerQueue = new tsQueue();
                if ( (m_MessageLoggerFileName == "")
                        || (m_MessageLoggerFileName == "NULL") )   // if none was supplied
                {
                        m_MessageLoggerThreadParms = new logParms(m_MessageLoggerQueue);
                }
                else
                {
                                                                 // truncate previous
version - if any
                        std::ofstream                        logFile(m_MessageLoggerFileName.c_str(),
std::ios::out|std::ios::trunc);
                        logFile.close();
                        m_MessageLoggerThreadParms       =       new       logParms(m_MessageLoggerQueue,
m_MessageLoggerFileName);
                }
                // start the background message-logging thread
```

AFOSR F49620-01-1-0371

```
                        m_MessageLoggerThreadHandle = CreateThread(  NULL,                          //     No     security
attributes
                                                                     0,
        // Default stack size

        (LPTHREAD_START_ROUTINE)MessageLogger,

        (LPVOID)(m_MessageLoggerThreadParms),
                                                                     0,
        // Create running

        &m_MessageLoggerThreadID   );
                        if ( INVALID_HANDLE_VALUE == m_MessageLoggerThreadHandle )
                        {
                                logMessage("Background processing could not be activated.");
                                string out = " Extended Err Info = ";
                                out += GetLastError();
                                logMessage(out);
                                exit(1);
                        }
                        else
                        {
                        if              (               !              SetThreadPriority(m_MessageLoggerThreadHandle,
THREAD_PRIORITY_ABOVE_NORMAL) )
                                {
                                        logMessage("Background thread priority could not be set.");
                                        string out = " Extended Err Info = ";
                                        out += GetLastError();
                                        logMessage(out);
                                        exit(1);
                                }
                        }
                }
                return;
    }
    out = this->getFederateName();
    out += ": Unknown interaction received.";
    logMessage(out);
}

//----------------------------------------------------------------
//
// METHOD:
//     void MessageLogging(LPVOID param)
//                 where: param = ptr->struct carrier
//                                                  {
//                                                       tsQueue            *queue;
//                                                       std::string        *name;
//                                                  };
//
//
// PURPOSE:
//         Represents a background thread that runs continuously to
```

```
//                        output logged messages to the display screen.
//              The parameters are:
//                                      tsQueue *              - the queue of text msgs
//                                      std::string  * - the name of a log-file or NULL
//                                                      (NULL => display to crt only)
//
// RETURN VALUES:
//      None.
//
//-------------------------------------------------------------
void MessageLogger(LPVOID param)
{
        logParms *LogParms = (logParms *)param;
        assert(LogParms != NULL);
        std::string xmitData;
        std::ofstream logFile;
        tsQueue *IntSupply = LogParms->InteractionHighway;
        assert(IntSupply != NULL);
        std::string filename = LogParms->logfileName;
        while (1==1)                                            // run for ANY length
        {
                while (IntSupply->queueSize() == 0)  // if there are no interactions waiting to be logged
                {
                        Sleep(1000);                           // wait for 1 sec
                }
                assert(IntSupply->queueSize() > 0);  // some interactions are waiting to be logged
                if (filename != "")                            // if a logfile name has been provided
                {
                        logFile.open(filename.c_str(), std::ios::app);
                        assert(logFile.is_open());             // file is successfully open
                }
                while (IntSupply->queueSize() > 0)    // do while any data remains
                {
                        xmitData = IntSupply->extract();// get the waiting text message
                        assert(xmitData.size() >= 0); // it *could* be 0-length
                        if (!logFile.is_open())                        // if a logfile has not been requested
                        {
                                std::cout << xmitData << std::endl;
                        }
                        else                                            // a logfile has been requested
                        {
                                logFile << xmitData << std::endl;
                        }
                }
                if (logFile.is_open())                         // if a logfile has been requested
                {
                        logFile.flush();                               // empty any holding-buffers
                        logFile.close();                               // close the file for safety
                        assert(!logFile.is_open());        // file is closed
                }
        }
}
```

-A-31-

A.1.7. `fedSpecCode.cpp`

```cpp
////////////////////////////////
// The debugger can't handle symbols more than 255 characters long.
// STL often creates symbols longer than that.
// When symbols are longer than 255 characters, the warning is issued.
#pragma warning(disable:4786)
////////////////////////////////
//**********************************************************************
//
// The code in this module will be written by the writer of
//      the federate simulation. The function names are to be
//      used as follows:
//
//      getPublicationNames()
//                      This function returns a stringArray of interaction
//                      names which this federate wishes to publish.
//
//      getSubscriptionNames()
//                      This function returns a stringArray of interaction
//                      names to which this federate wishes to subscribe.
//
//      fedMessage(string, string)
//                      This function provides the federate an opportunity to
//                      process the receipt of an interfederate interaction.
//                      Arg-1 contains the name of the interaction (Ex: RGtoDF01)
//                      Arg-2 contains the contents of the interaction string.
//
//      fedUpdate(long)
//                      This function provides the federate an opportunity to
//                      process a regularly scheduled update. The update
//                      time interval is set by a call to setUpdateTimeStep.
//                      Arg-1 contains the current time (at the update).
//
//**********************************************************************

#include "localFederate.h"
#include <iostream>          // for printout
using namespace std;
stringArray fedModel::getPublicationNames(void)
{
        stringArray ret;
        ret.push_back("");
        ret.push_back("");
        ret.push_back("");
        return ret;
}
stringArray fedModel::getSubscriptionNames(void)
{
        stringArray ret;
        ret.push_back("RGtoDF01");
        ret.push_back("RGtoDF02");
        ret.push_back("RGtoDF03");
        return ret;
```

```
}
// This function processes the receipt of an interfederate interaction.
void fedModel::fedMessage(const std::string &id, std::string &data)
{
        if (id == "RGtoDF01")
        {
                RGtoDF01 Msg = (new interaction<RGtoDF01>(data))->GetContents();
                // process a receipt of this message
        }
        if (id == "RGtoDF02")
        {
                RGtoDF02 Msg = (new interaction<RGtoDF02>(data))->GetContents();
                // process a receipt of this message
        }
        return;
}
void fedModel::fedUpdate(const unsigned long &n)
{
        std::cout << n << std::endl;
        // process a regularly scheduled update of the federate state
}
```

A.2 Dispatch/Routher Federate Code

A.2.1 Ambulance_Idle.h
```
#ifndef AMBULANCEIDLEDEFINITION_01
#define AMBULANCEIDLEDEFINITION_01
struct Ambulance_Idle
{
        unsigned long  time;
        unsigned long  AmbulanceID;
        double               X;
        double               Y;
        long                 Nearest_Node;
        long int             Link_ID;          // Link Id that is impassable
        unsigned int   Onboard_2;
        unsigned int   Onboard_3;
        unsigned int   Idle;

        Ambulance_Idle(void)
        {
                time = 0;
                AmbulanceID = 0;
                X = 0.0;
                Y = 0.0;
                Nearest_Node = 0;
                Link_ID=0;
                Onboard_2 = 0;
                Onboard_3 = 0;
                Idle=0;
        }
};
#endif
```

A.2.2 AmbulanceStuck.h

```cpp
#ifndef AMBULANCESTUCKDEFINITION_01
#define AMBULANCESTUCKDEFINITION_01
struct Ambulance_Stuck
{
        unsigned long  time;
        unsigned long  AmbulanceID;
        double                 X;
        double                 Y;
        long                   Nearest_Node;
        long int               Link_ID;          // Link Id that is impassable
        int                        Onboard_2;
        int                        Onboard_3;
        Ambulance_Stuck(void)
        {
                time = 0;
                AmbulanceID = 0;
                X = 0.0;
                Y = 0.0;
                Nearest_Node = 0;
                Link_ID=0;
                Onboard_2=0;
                Onboard_3=0;
        }
};
#endif
```

A.2.3 Casualty_Delivery.h

```cpp
#ifndef CASUALTYDELIVERYDEFINITION_01
#define CASUALTYDELIVERYDEFINITION_01
struct Casualty_Delivery
{
        unsigned long  time;
        //Changed on 01/19/05 -- Rashmi
        unsigned long  TrackID;
        unsigned long  Hospital;
        unsigned int   Severity;
        //Changed on 01/19/05 -- Rashmi
        Casualty_Delivery(void)
        {
                time = 0;
                TrackID = 0;
                Hospital = 0;
                Severity = 0;
        }
};
#endif
```

A.2.4 Casualty_Observation.h

```cpp
#ifndef CASUALTYOBSERVATIONDEFINITION_01
#define CASUALTYOBSERVATIONDEFINITION_01
#include <string>
```

-A-34-

AFOSR F49620-01-1-0371

```cpp
#include <cstdlib>
struct Casualty_Observation
{
        unsigned long  time;
        unsigned long  TrackID;
        double                X;
        double                Y;
        long                  Nearest_Node;
        unsigned int   Severity;
        float                 Sev_Prob_Vect[4];
        int                         pick;
        int                         ignore;
        Casualty_Observation(void)
        {
                time = 0;
                TrackID = 0;
                X = 0.0;
                Y = 0.0;
                Nearest_Node = 0;
                Severity = 0;
                Sev_Prob_Vect[0] = 0.0;
                Sev_Prob_Vect[1] = 0.0;
                Sev_Prob_Vect[2] = 0.0;
                Sev_Prob_Vect[3] = 0.0;
                pick=0;
                ignore=0;
        }
        std::string reportSelf(void)
        {
                char dummy[65];
                std::string ret = "\nCasualty_Observation";
                ret += "\n    time = ";
                ret +=              _ltoa(time, dummy, 10);
                ret += "\n    TrackID = ";
                ret +=              _ltoa(TrackID, dummy, 10);
                ret += "\n    X = ";
                ret +=              _gcvt(X, 9, dummy);
                ret += "\n    Y = ";
                ret +=              _gcvt(Y, 9, dummy);
                ret += "\n   Nearest_Node = ";
                ret +=              _ltoa(Nearest_Node, dummy, 10);
                ret += "\n    Severity = ";
                ret +=              _itoa(Severity, dummy, 10);
                ret += "\n    Sev_Prob_Vect[0] = ";
                ret +=              _gcvt(Sev_Prob_Vect[0], 7, dummy);
                ret += "\n    Sev_Prob_Vect[1] = ";
                ret +=              _gcvt(Sev_Prob_Vect[1], 7, dummy);
                ret += "\n    Sev_Prob_Vect[2] = ";
                ret +=              _gcvt(Sev_Prob_Vect[2], 7, dummy);
                ret += "\n    Sev_Prob_Vect[3] = ";
                ret +=              _gcvt(Sev_Prob_Vect[3], 7, dummy);
                ret += "\n    pick = ";
                ret +=              _itoa(pick, dummy, 10);
```

```
                ret += "\n    ignore = ";
                ret +=                    _itoa(ignore, dummy, 10);
                return ret;
        }
};
#endif
```

A.2.5 Casualty_Pickup.h

```
#ifndef CASUALTYPICKUPDEFINITION_01
#define CASUALTYPICKUPDEFINITION_01
struct  Casualty_Pickup
{
        unsigned long  time;
        //Changed on 01/19/05 -- Rashmi
        unsigned long  TrackID;
        unsigned long  Nearest_Node;
        unsigned int   Severity;
        //Changed on 01/19/05 -- Rashmi
        Casualty_Pickup(void)
        {
                time = 0;
                TrackID = 0;
                Nearest_Node = 0;
                Severity = 0;
        }
};
#endif
```

A.2.6 Cluster_Cell.h

```
#ifndef CLUSTERCELLDEFINITION_01
#define CLUSTERCELLDEFINITION_01
struct Cluster_Cell
{
        unsigned int   type;// Boundary cell or not
        double              Cell_X;                  // Centroid
        double              Cell_Y;
        unsigned int   Sev2_Count;          // Severity 2 = Medium Priority
        unsigned int   Sev3_Count;          // Severity 3 = High Priority
        int                     ignore;

        Cluster_Cell(void)
        {
                type=0;
                Cell_X = 0.0;
                Cell_Y = 0.0;
                Sev2_Count = 0;
                Sev3_Count = 0;
                ignore=0;


        }
};
#endif
```

A.2.7 Cluster_Identify.h

```
#ifndef CLUSTERIDENTIFICATIONDEFINITION_01
#define CLUSTERIDENTIFICATIONDEFINITION_01
```

```cpp
#include <vector>
#include "Cluster_Cell.h"

struct Cluster_Identify
{
        unsigned long  time;
        unsigned long  ClusterID;
        unsigned int   Side;                    // Size of cell side
        unsigned int   Count;
        std::vector<Cluster_Cell> Cells;
        Cluster_Identify(void)
        {
                time = 0;
                ClusterID = 0;
                Side = 0;
                Count = 0;
        }
        void addCell(const Cluster_Cell &c)
        {
                Cells.push_back(c);                     // makes a copy of c inside Cells
        }
        Cluster_Cell getCell(const int &c)
        {
                return Cells[c];                        // returns a copy
        }
//      Cluster_Cell subCell(const int &i)
        void subCell(const int &i)
        {
                if(Cells[i].Sev3_Count<3)
                {
                        Cells[i].Sev3_Count=0;
                }
                else
                {
                Cells[i].Sev3_Count = Cells[i].Sev3_Count - 3;
                }
//      return Cells[c];                        // returns a copy
        }
        void ignore(const int &i)
        {
                Cells[i].ignore = 1;
        }

        void Change_ignore_Flag(const int &i)
        {
                Cells[i].ignore = 0;
        }
        unsigned int getCount(void)
        {
                return Cells.size();
        }
};
```

A.2.8 Hospital_Capacity.h

```cpp
#ifndef HOSPITALCAPACITYDEFINITION_01
#define HOSPITALCAPACITYDEFINITION_01
struct  Hospital_Capacity
{
        unsigned long  HospitalID;
        unsigned int   Severity_1;
        unsigned int   Severity_2;
        unsigned int   Severity_3;
        double                X;
        double                Y;
        long                  Nearest_Node;
        float                 Delay_1;
        float          Delay_2;
        float                 Delay_3;
        int                         ignore;
        Hospital_Capacity(void)
        {
                HospitalID = 0;
                Severity_2 = 0;
                Severity_3 = 0;
                X=0.0;
                Y=0.0;
                Nearest_Node=0;
                Delay_1 = 0.0;
                Delay_2 = 0.0;
                Delay_3 = 0.0;
                ignore = 0;
        }
};
#endif
```

A.2.9 Hospital_Delay.h

```cpp
#ifndef HOSPITALDELAYDEFINITION_01
#define HOSPITALDELAYDEFINITION_01
struct  Hospital_Delay
{
        unsigned long  time;
        unsigned long  HospitalID;
        float                 Delay_1;
        float          Delay_2;
        float                 Delay_3;
        Hospital_Delay(void)
        {
                time = 0;
                HospitalID = 0;
                Delay_1 = 0.0;
                Delay_2 = 0.0;
                Delay_3 = 0.0;
        }
};
#endif
#endif
//#include <stdio.h>
```

AFOSR F49620-01-1-0371

```cpp
#include <stdlib.h>
#include <conio.h>
#include <ctype.h>
#include <iostream>
#include <fstream>
#include <math.h>
#include <malloc.h>
//#include <iomanip.h>
//#include <string.h>
#include <conio.h>
#include <time.h>
using namespace std;
#define MAX 50000
const double INF=9999999.0;
const double ISPEED=65.0;
const double USPEED=55.0;
const double SSPEED=40.0;
const double RSPEED=30.0;
const double XSPEED=20.0;

const char OBJ[5][40]={"Distance\0","Travel Time\0"};
struct NODE
{
        int                     NODEID                  ;
        int                     FEATUREID               ;
        long int        NODE_ID                 ;
        int                     KEY                                     ;
        int                     REGION                  ;
        char            DESCRIPT [150]  ;
};
struct LINK
{
        int                     LINKID                  ;
        int                     FEATUREID               ;
        long int        ANODE                   ;
        long int        BNODE                   ;
        long double     MILE                    ;
        int                     FCLASS                  ;
        float           DELAY                   ;
        double          SPEED                   ;
        long int        LINK_ID                 ;
        char            DESCRIPT [150]  ;
};
struct PathNode
{
        int                             nodix;
        int                             preix;
        int                             lnkix;
        int                             stats;
        double                  label;
        long double             time;
        long double             realtime;
        PathNode        *next;
```

```
        PathNode        *prev;
};
struct FinalPath
{
        int                     nodix;
        int                     lnkix;
        long double     label;
        long double     dist;
        long double     time;
        FinalPath       *next;
};
struct PathList
{
        int                     nodcnt;
        double          length;
        int                     *nodix;
        int                     *lnkix;
        double          *label;
        double          *dist;
        double          *time;
        PathList        *next;
        int                     flag;
        int                     pathno;
};
class INTEGRATION
{
        public:
                INTEGRATION();
                ~INTEGRATION();
                void            Reset();
                ofstream        ofpath;
                NODE            *node;long int NoNd;
                LINK            *link;long int NoLn;
                int                     *adjn;long int NoAj;
                int                     *ndix;
                int                     *lnix;
                int                     Final_Path[700];//***********************changed
                int                     Final_Path1[700];
                long double     Final_Path2[700];
                double          Final_Path3[700];
                int                     Node_Count;
                void            ReadNode(char fnode1[100]);
                void            ReadLink(char flink1[100]);
                void            ReadMtrx(char fadjn1[100],char fadjx1[100]);
                void            DeletePathNode(PathNode*);
                void            DeleteFinalPath(FinalPath*);
                void            DeletePathList(PathList*);
                void            DijkstraShortestPath(int, int, int);
                void            YenShortestPath(int, int, int, int);
                void            DisplayNodes(PathList*, int No, int opt, char*);  //opt=0 without WS, 1 with WS
                void            DisplayDetail(PathList*, int No, int opt, char*);
                void            DisplayBrief(PathList*, int No, int opt, char*);
```

```
            void          DisplayDetailSorted(PathList*, int No, int opt, char*);   //opt=0 without WS, 1
with WS
            void          DisplayBriefSorted(PathList*, int No, int opt, char*);
//          int*          SortPathList(PathList*, int);
            double        tstart;
            void          SetStartTime(double);
            int                   Origin;
            int                   Destin;
            //Section for Dijkstra's shortest path algorithm
            PathNode      *Head;
            PathNode      *Last;
            int                   size;
            FinalPath     *Final;
            PathList       *SP;
            int                   NodeCnt;
            double        FinLabel;
            void          SetPair(int, int);
            void          ResetSpath();


            double        GetFinalDistance();


            void          Dijkstra(int, int, int);
            void          AddList(PathNode*, int, int, int);
            void          UpdateLabel(PathNode*, PathNode*, int, int);
            void          DisplayPath(int, int);
            PathNode*     MinLabel();
            PathNode*     ExistList(int);
            void          BuildPath(int, int);
            void          FindLabel();
            void          AddPathSP();
            void          NodeClean();
            void          PathClean();
            //Section for Yen's k-shortest path algorithm
            PathList       *A;int NoA;
            PathList       *B;int NoB;
            int                   KPATH;
            int                   Iorigin;
            long double    *t1;
            long          *t2;


            FinalPath     *HELP;
            FinalPath     *ROOT;
            FinalPath     *SPUR;
            void          Dijkstra(int, int, int, int);
            void          AddList(PathNode*, int, int, int, int);
            void          UpdateLabel(PathNode*, PathNode*, int, int, int);
            void          ResetKpath();
            void          SetPair(int, int, int);
            void          AddPathA(FinalPath*);
            void          AddPathB(FinalPath*);
            void          GetPath();
            int                   GetLength();
            void          FindRoot(int);
```

```
        void            Compare(int,int);
        void            FindSpur(int);
        void            Combine(int);
        int                     IfSame(int, PathList*);
        int                     IfLoop();
        void            Replace();
        void            Retrieve();
        void            BackUp(int);
        void            Restore(int);
        int                     RootPenalty(int);
};


#ifndef h_LOCALFEDERATE_0001
#define h_LOCALFEDERATE_0001
#if defined(_MSC_VER) // if we're using Microsoft VC6
#define RTI_USES_STD_FSTREAM
#endif // defined(_MSC_VER)
#define _DIAGNOSTIC_PRINTOUT_
//#define _TimeProfileOnly_PRINTOUT_
#include <windows.h>
#include <RTI.hh>
#include "AllInteractions.h"  // All interaction classes are contained here
#include <vector>
#include <string>
#include <map>
#include "baseTypes.hh"
#include <fedtime.hh>
#include "logParms.h"
#include "tsQueue.h"
#include <assert.h>
//Added on 01/19/05 -- Rashmi
#include "Casualty_Observation.h"
#include "Casualty_Pickup.h"
#include "Casualty_Delivery.h"
#include "Road_Damage.h"
#include "Hospital_Capacity.h"
#include "Road_Delay.h"
#include "Hospital_Delay.h"
#include "Ambulance_Idle.h"
#include "Ambulance_Stuck.h"
#include "Cluster_Identify.h"
#include <vector>
#include "hosploc.h"
//Added on 01/19/05 -- Rashmi
typedef std::vector<std::string> stringVector;
typedef stringVector::iterator ipStringVector;
void MessageLogger(LPVOID param);
// function prototypes - definition in ModelMain
void SyncReady(const long &duration);
void AbortRun(void);


//-----------------------------------------------------------------
//
```

```
// CLASS:
//      fedModel
//
// PURPOSE:
//      The purpose of this class is to
//
//---------------------------------------------------------------
class fedModel
{
public:
    fedModel(void);
    virtual ~fedModel(void)
    {         }
    void setAmbAddress(RTI::RTIambassador* rtiAmb)
    {
                m_rtiAmb = rtiAmb;              // initialize the pointer
    }
    void Init(void);
    void Terminate(void);
    void receiveInteraction(
                            RTI::InteractionClassHandle theInteraction,
                            const RTI::ParameterHandleValuePairSet &theParameters,
                            const RTI::FedTime &theTime,
                            const char *theTag,
                            RTI::EventRetractionHandle theHandle);
    void Update( RTI::FedTime& newTime );
    //---------------------------------------------------------------
    // Accessor Methods
    //---------------------------------------------------------------
    RTI::FedTime const &GetLastTime()
    { return m_lastTime; };
    RTI::FedTime const &GetLastTimePlusLookahead()
    {
        m_TimePlusLookahead = m_lastTime;
      m_TimePlusLookahead += m_lookahead;
        return m_TimePlusLookahead;
    };
    const char *GetName()
    { return m_Name; };
    RTI::FedTime const &GetLookahead()
    { return m_lookahead;};
    double const &getUpdateTimeStep(void)
    { return m_UpdateTimeStep; }
    //---------------------------------------------------------------
    // Mutator Methods
    //---------------------------------------------------------------
    void  SetLastTime( RTI::FedTime const & time )
    { m_lastTime = time;};
    void SetLookahead( RTI::FedTime& time )
    { m_lookahead = time;};
        //---------------------------------------------------------------
        //
        // METHOD:
```

```
//      void PublishAndSubscribe(void)
//
// PURPOSE:
//      This method conveys to the federation the names of the
//              interactions which this federate will receive.
//
// RETURN VALUES:
//      None.
//
//-----------------------------------------------------------------
void PublishAndSubscribe(void)
{
        std::cout << "Publish and Subscribe Starting." << std::endl;
        try
        {
                if ( m_rtiAmb )
                {
                        m_RGtoALL01_TypeId = m_rtiAmb->getInteractionClassHandle("RGtoALL01");
                        m_RGtoALL01_Parms_TypeId     =     m_rtiAmb->getParameterHandle("RGtoALL01Parms",
m_RGtoALL01_TypeId);
                        m_rtiAmb->subscribeInteractionClass( m_RGtoALL01_TypeId );
                        m_RGtoALL02_TypeId = m_rtiAmb->getInteractionClassHandle("RGtoALL02");
                        m_RGtoALL02_Parms_TypeId     =     m_rtiAmb->getParameterHandle("RGtoALL02Parms",
m_RGtoALL02_TypeId);
                        m_rtiAmb->subscribeInteractionClass( m_RGtoALL02_TypeId );
                        m_rtiAmb->publishInteractionClass(m_TypeId["DPtoRG01"]);
                        m_rtiAmb->subscribeInteractionClass(m_TypeId["EDtoDP01"]);
                        //m_rtiAmb->subscribeInteractionClass(m_TypeId["EDtoDP02"]);
                        //m_rtiAmb->subscribeInteractionClass(m_TypeId["EDtoDP03"]);
                        m_rtiAmb->subscribeInteractionClass(m_TypeId["EDtoDP04"]);
                        m_rtiAmb->subscribeInteractionClass(m_TypeId["EDtoDP05"]);
                        m_rtiAmb->subscribeInteractionClass(m_TypeId["EDtoDP06"]);
                        m_rtiAmb->subscribeInteractionClass(m_TypeId["EDtoDP07"]);
                        m_rtiAmb->subscribeInteractionClass(m_TypeId["EDtoDP08"]);
                        m_rtiAmb->subscribeInteractionClass(m_TypeId["EDtoDP09"]);
                        m_rtiAmb->subscribeInteractionClass(m_TypeId["EDtoDP10"]);
                        m_rtiAmb->subscribeInteractionClass(m_TypeId["RGtoDP01"]);
                }
        }
        catch(RTI::InteractionClassNotDefined e)
        {
                throw "Interaction Class Not Defined";
        }
        catch(RTI::RTIinternalError e)
        {
                throw "RTI Internal Error";
        }
        std::cout << "Publish and Subscribe Ending." << std::endl;
}// PublishAndSubscribe
//-----------------------------------------------------------------
//
// METHOD:
//          void sendMsgs(parmType *parmlist)
```

```
//                      where parmlist is an interaction such as RGtoDF01
//
// PURPOSE:
//           Send sendInteraction to all subscribing federates.
//
// RETURN VALUES:
//     void
//
//----------------------------------------------------------------
template<class parmType>
void sendMsgs(parmType *parmlist)
{
        std::string sourcedest = parmlist->getIntName();
        interaction<parmType> *Interaction = new interaction<parmType>(*parmlist);
        RTI::ParameterHandleValuePairSet *pParams = RTI::ParameterSetFactory::create(1);
        RTI::InteractionClassHandle TypeID;
        try
        {
                TypeID = m_rtiAmb->getInteractionClassHandle(sourcedest.c_str());
        }
        catch(RTI::NameNotFound e)
        {
                throw "Name Not Found";
        }
        catch(RTI::RTIinternalError e)
        {
                throw "RTI Internal Error";
        }
        std::string IntParm = sourcedest;
        IntParm += "Parms";
        RTI::ParameterHandle Parms_TypeID;
        try
        {
                Parms_TypeID = m_rtiAmb->getParameterHandle(IntParm.c_str(), TypeID);
        }
        catch(RTI::InteractionClassNotDefined e)
        {
                throw "Interaction Class Not Defined";
        }
        catch(RTI::NameNotFound e)
        {
                throw "Name Not Found";
        }
        catch(RTI::RTIinternalError e)
        {
                throw "RTI Internal Error";
        }
        Interaction->setSrcDstCount(++(m_IntCount[sourcedest.c_str()]));
        try
        {
                pParams->add(Parms_TypeID, (char *)((Interaction->GetASCII()).c_str()),
                                ((strlen(((Interaction->GetASCII()).c_str()))+1) * sizeof(char)));
        }
```

```
        catch(RTI::ValueLengthExceeded e)
        {
                throw "Value Length Exceeded";
        }
        catch(RTI::ValueCountExceeded e)
        {
                throw "Value Count Exceeded";
        }
        // make sure that timestamp isn't in the past
        char dummy[65];
        char *timeStamp = _ltoa(parmlist->getTime(), dummy, 10);
        RTI::FedTime &outTime = (*(RTI::FedTimeFactory::decode(timeStamp)));
        if (outTime < this->GetLastTimePlusLookahead())
        {
                outTime = this->GetLastTimePlusLookahead();
        }
        try
        {
                (void)m_rtiAmb->sendInteraction(TypeID, *pParams, outTime, NULL);
        }
        catch(RTI::InteractionClassNotDefined e)
        {
                throw "Interaction Class Not Defined";
        }
        catch(RTI::InteractionClassNotPublished e)
        {
                throw "Interaction Class Not Published";
        }
        catch(RTI::InteractionParameterNotDefined e)
        {
                throw "Interaction Parameter Not Defined";
        }
        catch(RTI::RTIinternalError e)
        {
                throw "RTI Internal Error";
        }
        logSend(parmlist);
        delete pParams;
        delete Interaction;
} // sendMsgs
        //-----------------------------------------------------------------
private:
   double                      m_UpdateTimeStep;
   char                        *m_Name;                      // Name of this federate
   RTI::RTIambassador *m_rtiAmb;                // Pointer to RTIambassador
   RTI::FedTime       &m_lastTime;              // Time of previous update
   long                        m_LastUpdate;        // Time of previous update
   long                        m_PreviousUpdate;// saved time of previous update
   RTI::FedTime       &m_TimePlusLookahead;
   RTIfedTime          m_lookahead;             // Minimum time for action
   stringVector               m_IntName;
   stringVector               m_IntParm;
   RTI::InteractionClassHandle        m_RGtoALL01_TypeId;
```

```cpp
    RTI::ParameterHandle                    m_RGtoALL01_Parms_TypeId;
    RTI::InteractionClassHandle      m_RGtoALL02_TypeId;
    RTI::ParameterHandle                    m_RGtoALL02_Parms_TypeId;
    std::map<std::string, RTI::InteractionClassHandle>  m_TypeId;
    std::map<std::string, RTI::ParameterHandle>         m_Parms_TypeId;
    unsigned long                           m_ReceiptCount[INTCOUNT];

        // Type counts are used to supply values for SrcDstCount (in msgHeader.h)
        // values are initialized to 0 in fedModel::Init
        // values are assigned to SrcDstCount in fedModel::sendMsgs
        std::map<std::string, unsigned long> m_IntCount;
        RGtoALL01                   m_MessageRGtoALL01;
        RGtoALL02                   m_MessageRGtoALL02;
public:
    void logMessage(const std::string &txtMsg);
    void logTime(const std::string &txtMsg);
    void logReceipt(msgHeader &msg);
    void logSend(msgHeader *msg);
    void logCounterOut(const std::string &cntName, const long &cnt);
    void logCounterIn(const std::string &cntName, const long &cnt);


        char    separaterChar;
public:
        void parse(stringVector &tokens, char *rawData);
        unsigned long Dispatch_Hospital(Ambulance_Idle* Amb,int& ORIGIN_Hosp,int& DESTINATION_Hosp);
        unsigned long Dispatch_Casualty(Ambulance_Idle* Amb, int Step, int& ORIGIN_Amb,int& DESTINATION_Cas);
        unsigned  long  Dispatch_Cluster(Ambulance_Idle*  Amb,int&  ORIGIN_Amb,int&  DESTINATION_Cell,double&
Dispatch_Cell_X,double& Dispatch_Cell_Y);
        void Find_Nearest_Node(double x,double y,int& Nearest_Node_ID);
        std::string                 m_MessageLoggerFileName;
        logParms              *m_MessageLoggerThreadParms;
        tsQueue               *m_MessageLoggerQueue;
        DWORD                     m_MessageLoggerThreadID;
        HANDLE                    m_MessageLoggerThreadHandle;
        //Added on 01/19/05 -- Rashmi
        std::vector< Casualty_Observation* > v_co;
        std::vector< Casualty_Pickup* > v_cp;
        std::vector< Casualty_Delivery* > v_cd;
        std::vector< Road_Damage* > v_rd;
        std::vector< Hospital_Capacity* > v_hc;
        std::vector< Road_Delay* > v_rdy;
        std::vector< Hospital_Delay* > v_hd;
        typedef std::vector< Ambulance_Idle* > AmbQueue;
        typedef AmbQueue::iterator                  itAmbQueue;
        AmbQueue                v_ai;
        std::vector< Ambulance_Stuck* > v_as;
        std::vector< Cluster_Identify* > v_ci;
        std::vector< hospLoc > v_hl;
        //Added on 01/19/05 -- Rashmi

};
#endif
#ifndef h_PARAMETERSTOINDEPENDENTMESSAGELOGGINGTHREAD_001
```

```
#define h_PARAMETERSTOINDEPENDENTMESSAGELOGGINGTHREAD_001
#include <string>
#include <cassert>
#include "tsQueue.h"
// define a carrier-object to transmit the addresses of both
//     the the tsQueue and the filename that have been created.
// if the filename * == NULL, display will be to the CRT only
struct logParms
{
        tsQueue            *InteractionHighway;
        std::string         logfileName;
        logParms(tsQueue *queue, const std::string &name)
        {
                assert(queue != NULL);
                assert(name.size() > 0);
                InteractionHighway = queue;
                logfileName = name;
        }
        logParms(tsQueue *queue)
        {
                assert(queue != NULL);
                InteractionHighway = queue;
                logfileName = "";
        }
        ~logParms(void)
        {
                delete InteractionHighway;
        }
};
#endif
#pragma warning(disable:4786)
#include "Integration.h"
#include "localFederate.h"
#include <cassert>
void          origin_destin(FILE *fp_origin_destin, char file_origin_destin[200]);
void          CallYen(INTEGRATION* Beyza,int org,int dst);
class MainClass
{
public:
        MainClass(void *arg)
        {
                localFederate = (fedModel *)arg;
                i = 0;
                m = 0;
        }
fedModel *localFederate;                         // pointer to our federate
int                 Exit_Entry[50];
char          fnode[100];
char          flink[100];
char          fadjn[100];
char          fadjx[100];
int                 i;
int                 m;
```

```
int                  number_entry_exit;
int                  kpath,opt;
int                  exit_points;
int                  entry_points;
double        Shortest_Path;
int                  final_exit;
int                  final_entry;
long          display_nodes_origin[1000];
long          display_links_origin[1000];
long          display_nodes_interm[1000];
long          display_links_interm[1000];
long          display_nodes_destin[1000];
long          display_links_destin[1000];
int                  Cnt_Exit;
int                  Cnt_Interm;
int                  Cnt_Entry;
struct Final_output_origin
{
       int                  Final_Nodes_origin[500];
       int                  Final_Links_origin[500];
       long double    Final_Dist_origin[500];
       double         Final_Time_origin[500];
       double         Path_Distance_origin_exit;
       int                  count;
} origin_final_output[50];
struct Final_output_destin
{
       int                  Final_Nodes_destin[500];
       int                  Final_Links_destin[500];
       long double    Final_Dist_destin[500];
       double         Final_Time_destin[500];
       double         Path_Distance_destin_entry;
       int                  count;
} destin_final_output[50];
struct Final_output_interm
{
       int                  Final_Nodes_interm[500]       ;
       int                  Final_Links_interm[500]       ;
       long double    Final_Dist_interm[500] ;
       double         Final_Time_interm[500] ;
       double         Path_Distance_interm   ;
       int                  count                              ;
} interm_final_output[50][50];
struct original_data
{
       long    from_id[MAX]   ;
       long    to_id[MAX]              ;
       int            region[MAX]            ;
} input_nodes;
struct nodes
{
       int            sno[7500]             ;
       int            node_id[7500]  ;
```

```
        long    node[7500]                 ;
        int             key[7500]                   ;
        long    regn[7500]              ;
} convert_node;
struct links
{
        int                     sno[7500]                       ;
        int                     linkid[7500]    ;
        int                     anode[7500]                     ;
        int                     bnode[7500]                     ;
        long double     mile[7500]          ;
        int                     fclass[7500]    ;
        float           delay[7500]             ;
        float           speed[7500]             ;
        long int        link_id[7500]   ;
}convert_link;
void main_method(int origin, int destination)
{
        string out;
        char dummy[65];
FILE *fp;
FILE *fp_origin;
FILE *fp_destin;
FILE *fp_macro;
FILE *fp_origin_links;
FILE *fp_interm_links;
FILE *fp_destin_links;
int             j = 0;
int             x = 0;
int             y = 0;
int             n = 0;
int             origin_actual=0;
int             destin_actual=0;
int             Exit_origin[50];
int             Entry_destin[50];
char    file_origin[500];
char    file_destin[500];
char    file_macro[500];
char    file_origin_links[500];
char    file_interm_links[500];
char    file_destin_links[500];
int             origin_region=0;
int             destin_region=0;
int             Macro_Enter[50];
int             Macro_Exit[50];
int             Enter[50];
int             Exit[50];

        Cnt_Exit=0;
        Cnt_Interm=0;
        Cnt_Entry=0;
        kpath=1;
        opt=1;
```

```
// FINDING THE ORIGIN & DESTINATION REGION
        fp = fopen("original.txt", "r");
        if (fp == NULL)
        {
                cout<<"Could not open file for reading"<<endl;
        }

        while (feof(fp) == 0)
        {
                fscanf(fp,          "%ld\t%ld\t%d\n",          &input_nodes.from_id[i],          &input_nodes.to_id[i],
&input_nodes.region[i]);
                if ((input_nodes.from_id[i]==origin) || (input_nodes.to_id[i]==origin))
                {
                        origin_region=input_nodes.region[i];
                }
                if ((input_nodes.from_id[i]==destination) || (input_nodes.to_id[i]==destination))
                {
                        destin_region=input_nodes.region[i];
                }
                i++;
        }
        fclose(fp);
//      cout<<"Origin region is: "<<origin_region<<endl;
//      cout<<"Destination region is: "<<destin_region<<endl;
        // *************If the Origin or Destination node is not listed in the database ==>> IGNORE ROUTING CALL
AND MOVE TO NEXT THING ************* //
        if(origin_region!=0 && destin_region!=0)
        {
// FINDING THE ORIGINAL NODE ID FOR ORIGIN
                sprintf(file_origin,"node%d.txt",origin_region);
                fp_origin=fopen(file_origin,"r");
                if (fp_origin == NULL)
                {
                        cout<<"Could not open file for reading"<<endl;
                }
                i=0;
                m=0;
                while (feof(fp_origin) == 0)
                {
                        fscanf(fp_origin,               "%d\t%d\t%ld\t%d\t%ld\n",               &convert_node.sno[i],
&convert_node.node_id[i], &convert_node.node[i],&convert_node.key[i],&convert_node.regn[i]);
                        if(convert_node.node[i]==origin)
                        {
                                origin_actual = convert_node.node_id[i];
                                break;
                        }
                        i++;
                }
                fclose(fp_origin);
        // FINDING THE ORIGINAL NODE ID FOR DESTINATION
                sprintf(file_destin,"node%d.txt",destin_region);
                fp_destin=fopen(file_destin,"r");
```

-A-51-

AFOSR F49620-01-1-0371

```
                if (fp_destin == NULL)
                {
                        cout<<"Could not open file for reading"<<endl;
                }
                i=0;
                m=0;
                while (feof(fp_destin) == 0)
                {
                        fscanf(fp_destin,"%d\t%d\t%ld\t%d\t%ld\n",                         &convert_node.sno[i],
&convert_node.node_id[i], &convert_node.node[i],&convert_node.key[i],&convert_node.regn[i]);
                        if(convert_node.node[i]==destination)
                        {
                                destin_actual = convert_node.node_id[i];
                                break;
                        }
                        i++;
                }
                fclose(fp_destin);
        //*******************************************************
        //      CASE 1: BOTH ORIGIN AND DESTINATION ARE IN SAME REGION
                if(origin_region==destin_region)
                {
out = "CASE 1: BOTH ORIGIN AND DESTINATION ARE IN SAME REGION";
localFederate->logMessage(out);
                        sprintf(fnode,"node%d.txt",origin_region);
                        sprintf(flink,"link%d.txt",origin_region);
                        sprintf(fadjn,"adjacency%d.txt",origin_region);
                        sprintf(fadjx,"index%d.txt",origin_region);
                        INTEGRATION *Beyza =new INTEGRATION;
                        Beyza->ReadNode(fnode);
                        Beyza->ReadLink(flink);
                        Beyza->ReadMtrx(fadjn,fadjx);
                        CallYen(Beyza,origin_actual,destin_actual);
                        n=0;
                        origin_final_output[n].Path_Distance_origin_exit = Beyza->GetFinalDistance()/1600;
                        for(i=0;i<Beyza->Node_Count;i++)
                        {
                                origin_final_output[n].Final_Nodes_origin[i]=Beyza->Final_Path[i];
                                origin_final_output[n].Final_Links_origin[i]=Beyza->Final_Path1[i];
                                origin_final_output[n].Final_Dist_origin[i]=Beyza->Final_Path2[i];
                                origin_final_output[n].Final_Time_origin[i]=Beyza->Final_Path3[i];
                        }
                        delete Beyza;
                        origin_final_output[n].count=i;
                        Shortest_Path=origin_final_output[n].Path_Distance_origin_exit;
//                      cout<<"Distance    from    "<<origin<<"    to    "<<destination <<"    is    " <<
origin_final_output[n].Path_Distance_origin_exit<<"   miles"<<endl;
                        cout<<"100\t"<<"100\t"<<origin_final_output[n].Path_Distance_origin_exit<<endl;
                        if(Shortest_Path > (30)||(Shortest_Path<0))
                        {
//                              cout<<"  ==>>  Inside MAIN:  THE DESTINATION IS UNREACHABLE  <<==   "<<endl;
                        }
```

AFOSR F49620-01-1-0371

```
    //DISPLAY NODES
                    else
                    {
                            fp_origin=fopen(file_origin,"r");
                            if (fp_origin == NULL)
                            {
                                    cout<<"Could not open file for reading"<<endl;
                            }
//                      cout<<endl;

                            i=0;
                            m=0;
                            while (feof(fp_origin) == 0)
                            {
                                    fscanf(fp_origin,       "%d\t%d\t%ld\t%d\t%ld\n",        &convert_node.sno[i],
&convert_node.node_id[i], &convert_node.node[i],&convert_node.key[i],&convert_node.regn[i]);
                                    i++;
                            }
                            fclose(fp_origin);
                            Cnt_Exit=origin_final_output[n].count;
//      cout<<"===>>  Count in MAIN FUNCTION FOR EXIT is:  "<< origin_final_output[n].count<<endl;
                            for (m=0;m<origin_final_output[n].count; m++)
                            {
                                    for(j=0;j<i;j++)
                                    {
                                            if(origin_final_output[n].Final_Nodes_origin[m]                       ==
convert_node.node_id[j])
                                            {
                                                    display_nodes_origin[m]=convert_node.node[j];
                                                    break;
                                            }
                                    }

                            }
                    //      DISPLAY LINKS
                            sprintf(file_origin_links,"link%d.txt",origin_region);
                            fp_origin_links=fopen(file_origin_links,"r");
                            if (fp_origin_links == NULL)
                            {
                                    cout<<"Could not open file for reading"<<endl;
                            }
                            i=0;
                            m=0;
                            while (feof(fp_origin_links) == 0)
                            {
                                    fscanf(fp_origin_links,         "%d\t%d\t%d\t%d\t%lf\t%d\t%d\t%f\t%ld\n",
&convert_link.sno[i],                                                            &convert_link.linkid[i],
&convert_link.anode[i],&convert_link.bnode[i],&convert_link.mile[i],&convert_link.fclass,&convert_link.delay[i],
&convert_link.speed[i],&convert_link.link_id[i]);
                                    i++;
                            }
                            fclose(fp_origin_links);
                            for (m=0;m<origin_final_output[n].count; m++)
```

-A-53-

```
                                    {
                                        for(j=0;j<i;j++)
                                        {
                                            if(origin_final_output[n].Final_Links_origin[m]                  ==
convert_link.linkid[j])
                                            {
                                                display_links_origin[m]=convert_link.link_id[j];
                                                break;
                                            }
                                        }
                                    }
                                // DISPLAY
        /*              for (m=0;m<origin_final_output[n].count-1; m++)
                        {
                        //      cout<<"Link\t\t"<<display_links_origin[m+1]<<endl;
                        //      cout<<"Node\t\t"<<display_nodes_origin[m+1]<<endl;
                        //      cout<<"Dist\t\t"<<origin_final_output[n].Final_Dist_origin[m+1]<<endl;
                        //      cout<<"Time\t\t"<<origin_final_output[n].Final_Time_origin[m+1]<<endl;
                        }
        */
                    } //END ELSE LOOP
            } //***************************************END IF LOOP 1

        //********************************************************
        //      CASE 2: ORIGIN IS AT THE MACRO NETWORK AND DESTINATION IS IN SOME REGION
                else if(origin_region==99 && destin_region!=99)
                {
out = "CASE 2: ORIGIN IS AT THE MACRO NETWORK AND DESTINATION IS IN SOME REGION";
localFederate->logMessage(out);
                    sprintf(fnode,"node%d.txt",destin_region);//SP FROM ENTRY POINT TO DESTINATION
                    sprintf(flink,"link%d.txt",destin_region);
                    sprintf(fadjn,"adjacency%d.txt",destin_region);
                    sprintf(fadjx,"index%d.txt",destin_region);
                    INTEGRATION *Beyza=new INTEGRATION;
                    Beyza->ReadNode(fnode);
                    Beyza->ReadLink(flink);
                    Beyza->ReadMtrx(fadjn,fadjx);
                    origin_destin(fp_destin,file_destin);
                    entry_points=number_entry_exit;
                    for(n=0;n<number_entry_exit;n++)
                    {
                        Entry_destin[n]= Exit_Entry[n];
                        CallYen(Beyza,Entry_destin[n],destin_actual);
                        destin_final_output[n].Path_Distance_destin_entry          =          Beyza-
>GetFinalDistance()/1600;
                        for(i=0;i<Beyza->Node_Count;i++)
                        {
                            destin_final_output[n].Final_Nodes_destin[i]=Beyza->Final_Path[i];
                            destin_final_output[n].Final_Links_destin[i]=Beyza->Final_Path1[i];
                            destin_final_output[n].Final_Dist_destin[i]=Beyza->Final_Path2[i];
                            destin_final_output[n].Final_Time_destin[i]=Beyza->Final_Path3[i];
                        }
                        destin_final_output[n].count=i;
```

```
                }
                delete Beyza;
//Convert Entry points for Macro Network
                fp_destin=fopen(file_destin,"r");
                if (fp_destin == NULL)
                {
                        cout<<"Could not open file for reading"<<endl;
                }
                i=0;
                while (feof(fp_destin) == 0)
                {
                        fscanf(fp_destin,          "%d\t%d\t%ld\t%d\t%ld\n",          &convert_node.sno[i],
&convert_node.node_id[i], &convert_node.node[i],&convert_node.key[i],&convert_node.regn[i]);
                        i++;
                }
                fclose(fp_destin);
                for(n=0;n<number_entry_exit;n++)
                {
                        for(j=0;j<i;j++)
                        {
                                if(Entry_destin[n]==convert_node.node_id[j])
                                {
                                        Macro_Exit[n] = convert_node.node[j];
                                        break;
                                }
                        }
                }


//      Convert back for Macro Network

                sprintf(file_macro,"node%d.txt",99);
                fp_macro=fopen(file_macro,"r");
                if (fp_macro == NULL)
                {
                        cout<<"Could not open file for reading"<<endl;
                }
                i=0;

                while (feof(fp_macro) == 0)
                {
                        fscanf(fp_macro,          "%d\t%d\t%ld\t%d\t%ld\n",          &convert_node.sno[i],
&convert_node.node_id[i], &convert_node.node[i],&convert_node.key[i],&convert_node.regn[i]);
                        i++;
                }

                fclose(fp_macro);
                entry_points=0; // ********** CHANGED 03/23/2006
                for(n=0;n<number_entry_exit;n++)
                {
                        for(j=0;j<i;j++)
                        {
                                if(convert_node.node[j]==Macro_Exit[n])
                                {
```

AFOSR F49620-01-1-0371

```
                                    Exit[entry_points] = convert_node.node_id[j];
                                    ++entry_points;  // *********** CHANGED 03/23/2006
                                    break;
                             }
                      }
               }

    // SP FROM ORIGIN TO ENTRY POINT
               sprintf(fnode,"node99.txt");
               sprintf(flink,"link99.txt");
               sprintf(fadjn,"adjacency99.txt");
               sprintf(fadjx,"index99.txt");
               Beyza =new INTEGRATION;
               Beyza->ReadNode(fnode);
               Beyza->ReadLink(flink);
               Beyza->ReadMtrx(fadjn,fadjx);
               n=0;
               exit_points=1;
               for(m=0;m<exit_points;m++) //*****************************Changed
               {
                      origin_final_output[m].Path_Distance_origin_exit = 0.0;
               }
               for(m=0;m<entry_points;m++)
               {
                      CallYen(Beyza,origin_actual,Exit[m]);
                      interm_final_output[n][m].Path_Distance_interm = Beyza->GetFinalDistance()/1600;
                      for(i=0;i<Beyza->Node_Count;i++)
                      {
                             interm_final_output[n][m].Final_Nodes_interm[i]= Beyza->Final_Path[i];
                             interm_final_output[n][m].Final_Links_interm[i]= Beyza->Final_Path1[i];
                             interm_final_output[n][m].Final_Dist_interm[i]= Beyza->Final_Path2[i];
                             interm_final_output[n][m].Final_Time_interm[i]= Beyza->Final_Path3[i];
                      }
                      interm_final_output[n][m].count=i;
               }
               delete Beyza;
               Shortest_Path = GetShortestDistance();
//             cout<<"Distance from  "<<origin<<"  to   "<<destination <<"  is  " << Shortest_Path<<"
miles"<<endl;
               cout<<"100\t"<<"100\t"<<Shortest_Path<<endl;

               if(Shortest_Path > (30)||(Shortest_Path<0))
               {
//                    cout<<"  ==>>  Inside MAIN:  THE DESTINATION IS UNREACHABLE  <<==   "<<endl;
               }
    //DISPLAY NODES FOR ORIGIN TO ENTRY
               else
               {
                      fp_macro=fopen(file_macro,"r");
                      if (fp_macro == NULL)
                      {
                             cout<<"Could not open file for reading"<<endl;
                      }
```

AFOSR F49620-01-1-0371

```cpp
//                              cout<<endl;

                                i=0;
                                m=0;
                                while (feof(fp_macro) == 0)
                                {
                                        fscanf(fp_macro,       "%d\t%d\t%ld\t%d\t%ld\n",       &convert_node.sno[i],
&convert_node.node_id[i], &convert_node.node[i],&convert_node.key[i],&convert_node.regn[i]);
                                        i++;
                                }
                                fclose(fp_macro);
                                Cnt_Interm=interm_final_output[final_exit-1][final_entry-1].count;
//      cout<<"===>>   Count  in MAIN  FUNCTION FOR CASE 2 INTERM   is:   "<<  interm_final_output[final_exit-
1][final_entry-1].count<<endl;
                                for (m=0;m<interm_final_output[final_exit-1][final_entry-1].count; m++)
                                {
                                        for(j=0;j<i;j++)
                                        {
                                                if(interm_final_output[final_exit-1][final_entry-
1].Final_Nodes_interm[m] == convert_node.node_id[j])
                                                {
                                                        display_nodes_interm[m]=convert_node.node[j];
                                                        break;
                                                }
                                        }
                                }
                //DISPLAY LINKS FOR ORIGIN TO ENTRY
                                sprintf(file_interm_links,"link99.txt");
                                fp_interm_links=fopen(file_interm_links,"r");
                                if (fp_interm_links == NULL)
                                {
                                        cout<<"Could not open file for reading"<<endl;
                                }
                                i=0;
                                m=0;
                                while (feof(fp_interm_links) == 0)
                                {
                                        fscanf(fp_interm_links,          "%d\t%d\t%d\t%d\t%lf\t%d\t%d\t%f\t%ld\n",
&convert_link.sno[i],                                                  &convert_link.linkid[i],
&convert_link.anode[i],&convert_link.bnode[i],&convert_link.mile[i],&convert_link.fclass,&convert_link.delay[i],
&convert_link.speed[i],&convert_link.link_id[i]);
                                        i++;
                                }
                                fclose(fp_interm_links);
                                for (m=0;m<interm_final_output[final_exit-1][final_entry-1].count; m++)
                                {
                                        for(j=0;j<i;j++)
                                        {
                                                if(interm_final_output[final_exit-1][final_entry-
1].Final_Links_interm[m] == convert_link.linkid[j])
                                                {
                                                        display_links_interm[m]=convert_link.link_id[j];
```

-A-57-

```
                                                 break;
                                    }
                           }
                  }
             // DISPLAY
   /*                           for (m=0;m<interm_final_output[final_exit-1][final_entry-1].count-1; m++)
                           {
                           //     cout<<"Links\t\t"<<display_links_interm[m+1]<<endl;
                           //     cout<<"Nodes\t\t"<<display_nodes_interm[m+1]<<endl;
                           //     cout<<"Dist\t\t"<<interm_final_output[final_exit-1][final_entry-
   1].Final_Dist_interm[m+1]<<endl;
                           //     cout<<"Time\t\t"<<interm_final_output[final_exit-1][final_entry-
   1].Final_Time_interm[m+1]<<endl;
                           }
       */
             //DISPLAY NODES FOR ENTRY TO DESTINATION
                           fp_destin=fopen(file_destin,"r");
                           if (fp_destin == NULL)
                           {
                                    cout<<"Could not open file for reading"<<endl;
                           }

   //                       cout<<endl;

                           i=0;
                           m=0;
                           while (feof(fp_destin) == 0)
                           {
                                    fscanf(fp_destin,    "%d\t%d\t%ld\t%d\t%ld\n",    &convert_node.sno[i],
   &convert_node.node_id[i], &convert_node.node[i],&convert_node.key[i],&convert_node.regn[i]);
                                    i++;
                           }
                           fclose(fp_destin);
                           Cnt_Entry= destin_final_output[final_entry-1].count;
   //     cout<<"===>>   Count  in MAIN  FUNCTION  FOR CASE 2 DESTIN   is:   "<< destin_final_output[final_entry-
   1].count<<endl;
                           for (m=0;m<destin_final_output[final_entry-1].count; m++)
                           {
                                    for(j=0;j<i;j++)
                                    {
                                            if(destin_final_output[final_entry-1].Final_Nodes_destin[m]        ==
   convert_node.node_id[j])
                                            {
                                                    display_nodes_destin[m]=convert_node.node[j];
                                                    break;
                                            }
                                    }
                           }

             //DISPLAY LINKS FOR ENTRY TO DESTINATION
                           sprintf(file_destin_links,"link%d.txt",destin_region);
                           fp_destin_links=fopen(file_destin_links,"r");
                           if (fp_destin_links == NULL)
```

-A-58-

```
                              {
                                    cout<<"Could not open file for reading"<<endl;
                              }
                              i=0;
                              m=0;
                              while (feof(fp_destin_links) == 0)
                              {
                                    fscanf(fp_destin_links,          "%d\t%d\t%d\t%d\t%lf\t%d\t%d\t%f\t%ld\n",
&convert_link.sno[i],                                              &convert_link.linkid[i],
&convert_link.anode[i],&convert_link.bnode[i],&convert_link.mile[i],&convert_link.fclass,&convert_link.delay[i],
&convert_link.speed[i],&convert_link.link_id[i]);
                                    i++;
                              }
                              fclose(fp_destin_links);
                              for (m=0;m<destin_final_output[final_entry-1].count; m++)
                              {
                                    for(j=0;j<i;j++)
                                    {
                                          if(destin_final_output[final_entry-1].Final_Links_destin[m]        ==
convert_link.linkid[j])
                                          {
                                                display_links_destin[m]=convert_link.link_id[j];
                                                break;
                                          }
                                    }
                              }
                        // DISPLAY
        /*                    for (m=0;m<destin_final_output[final_entry-1].count-1; m++)
                              {
                              //     cout<<"Links\t\t"<<display_links_destin[m+1]<<endl;
                              //     cout<<"Nodes\t\t"<<display_nodes_destin[m+1]<<endl;
                              //     cout<<"Dist\t\t"<<destin_final_output[final_entry-
1].Final_Dist_destin[m+1]<<endl;
                              //     cout<<"Time\t\t"<<destin_final_output[final_entry-
1].Final_Time_destin[m+1]<<endl;
                              }
        */
                        } //  END ELSE LOOP
              } ////////////////////////////////////////////////END IF LOOP2
        //*****************************************************
        //     CASE 3: DESTINATION IS AT THE MACRO NETWORK AND ORIGIN IS IN SOME REGION

              else if(origin_region!=99 && destin_region==99)
              {
out = "CASE 3: DESTINATION IS AT THE MACRO NETWORK AND ORIGIN IS IN SOME REGION";
localFederate->logMessage(out);
                    sprintf(fnode,"node%d.txt",origin_region);// FINDING THE SP FROM ORIGIN TO EXIT POINTS
                    sprintf(flink,"link%d.txt",origin_region);
                    sprintf(fadjn,"adjacency%d.txt",origin_region);
                    sprintf(fadjx,"index%d.txt",origin_region);
                    INTEGRATION *Beyza=new INTEGRATION;
                    Beyza->ReadNode(fnode);
                    Beyza->ReadLink(flink);
```

-A-59-

```
                    Beyza->ReadMtrx(fadjn,fadjx);

                    origin_destin(fp_origin,file_origin);
                    exit_points=number_entry_exit;
                    for(n=0;n<number_entry_exit;n++)
                    {
                            Exit_origin[n]= Exit_Entry[n];
                            CallYen(Beyza,origin_actual,Exit_origin[n]);
                            origin_final_output[n].Path_Distance_origin_exit = Beyza->GetFinalDistance()/1600;
                            for(i=0;i<Beyza->Node_Count;i++)
                            {
                                    origin_final_output[n].Final_Nodes_origin[i]=Beyza->Final_Path[i];
                                    origin_final_output[n].Final_Links_origin[i]=Beyza->Final_Path1[i];
                                    origin_final_output[n].Final_Dist_origin[i]=Beyza->Final_Path2[i];
                                    origin_final_output[n].Final_Time_origin[i]=Beyza->Final_Path3[i];
                            }
                            origin_final_output[n].count=i;
                    }
                    delete Beyza;
       //Convert Exit points for Macro Network
                    fp_origin=fopen(file_origin,"r");
                    if (fp_origin == NULL)
                    {
                            cout<<"Could not open file for reading"<<endl;
                    }
                    i=0;
                    while (feof(fp_origin) == 0)
                    {
                            fscanf(fp_origin,        "%d\t%d\t%ld\t%d\t%ld\n",        &convert_node.sno[i],
&convert_node.node_id[i], &convert_node.node[i],&convert_node.key[i],&convert_node.regn[i]);
                            i++;
                    }
                    fclose(fp_origin);
                    for(n=0;n<number_entry_exit;n++)
                    {
                            for(j=0;j<i;j++)
                            {
                                    if(Exit_origin[n]==convert_node.node_id[j])
                                    {
                                            Macro_Enter[n] = convert_node.node[j];
                                            break;
                                    }
                            }
                    }

       //      Convert back for Macro Network
                    sprintf(file_macro,"node%d.txt",99);
                    fp_macro=fopen(file_macro,"r");
                    if (fp_macro == NULL)
                    {
                            cout<<"Could not open file for reading"<<endl;
                    }
                    i=0;
```

-A-60-

```
                            while (feof(fp_macro) == 0)
                            {
                                    fscanf(fp_macro,          "%d\t%d\t%ld\t%d\t%ld\n",          &convert_node.sno[i],
        &convert_node.node_id[i], &convert_node.node[i],&convert_node.key[i],&convert_node.regn[i]);
                                    i++;
                            }

                            fclose(fp_macro);
                            exit_points=0; // *********** CHANGED 03/23/2006
                            for(n=0;n<number_entry_exit;n++)
                            {
                                    for(j=0;j<i;j++)
                                    {
                                            if(Macro_Enter[n]==convert_node.node[j])
                                            {
                                                    Enter[exit_points] = convert_node.node_id[j];
                                                    ++exit_points;// *********** CHANGED 03/23/2006
                                                    break;
                                            }
                                    }
                            }
        // SP FROM EXIT POINTS TO DESTINATION
                            sprintf(fnode,"node99.txt");
                            sprintf(flink,"link99.txt");
                            sprintf(fadjn,"adjacency99.txt");
                            sprintf(fadjx,"index99.txt");
                            Beyza =new INTEGRATION;
                            Beyza->ReadNode(fnode);
                            Beyza->ReadLink(flink);
                            Beyza->ReadMtrx(fadjn,fadjx);
                            m=0;
                            entry_points=1;

                            for(n=0;n<entry_points;n++)//*****************
                            {
                                    destin_final_output[n].Path_Distance_destin_entry = 0.0;
                            }
                            for(n=0;n<exit_points;n++)
                            {
                                    CallYen(Beyza,Enter[n],destin_actual);
                                    interm_final_output[n][m].Path_Distance_interm = Beyza->GetFinalDistance()/1600;
                                    for(i=0;i<Beyza->Node_Count;i++)
                                    {
                                            interm_final_output[n][m].Final_Nodes_interm[i]= Beyza->Final_Path[i];
                                            interm_final_output[n][m].Final_Links_interm[i]= Beyza->Final_Path1[i];
                                            interm_final_output[n][m].Final_Dist_interm[i]= Beyza->Final_Path2[i];
                                            interm_final_output[n][m].Final_Time_interm[i]= Beyza->Final_Path3[i];
                                    }

                                    interm_final_output[n][m].count=i;
                            }
                            delete Beyza;
                            Shortest_Path = GetShortestDistance();
```

-A-61-

AFOSR F49620-01-1-0371

```
//                  cout<<"Distance from   "<<origin<<"  to   "<<destination <<"  is   " << Shortest_Path<<"
miles"<<endl;
                    cout<<"100\t"<<"100\t"<<Shortest_Path<<endl;

                    if(Shortest_Path > (30)||(Shortest_Path<0))
                    {
//                      cout<<"  ==>>  Inside MAIN:  THE DESTINATION IS UNREACHABLE  <<==   "<<endl;
                    }

        //DISPLAY NODES FROM ORIGIN TO EXIT
                    else
                    {
                        fp_origin=fopen(file_origin,"r");
                        if (fp_origin == NULL)
                        {
                                cout<<"Could not open file for reading"<<endl;
                        }

//                      cout<<endl;

                        i=0;
                        m=0;
                        while (feof(fp_origin) == 0)
                        {
                                fscanf(fp_origin,      "%d\t%d\t%ld\t%d\t%ld\n",      &convert_node.sno[i],
&convert_node.node_id[i], &convert_node.node[i],&convert_node.key[i],&convert_node.regn[i]);
                                i++;
                        }
                        fclose(fp_origin);
                        Cnt_Exit=origin_final_output[final_exit-1].count;
//      cout<<"===>>  Count in MAIN FUNCTION FOR CASE 3 ORIGIN  is:  "<< Cnt_Exit <<endl;
                        for (m=0;m<origin_final_output[final_exit-1].count; m++)
                        {
                                for(j=0;j<i;j++)
                                {
                                        if(origin_final_output[final_exit-1].Final_Nodes_origin[m]        ==
convert_node.node_id[j])
                                        {
                                                display_nodes_origin[m]=convert_node.node[j];
                                                break;
                                        }
                                }

                        }
                //      DISPLAY LINKS FROM ORIGIN TO EXIT
                        sprintf(file_origin_links,"link%d.txt",origin_region);
                        fp_origin_links=fopen(file_origin_links,"r");
                        if (fp_origin_links == NULL)
                        {
                                cout<<"Could not open file for reading"<<endl;
                        }
                        i=0;
                        m=0;
```

-A-62-

```
                          while (feof(fp_origin_links) == 0)
                          {
                                  fscanf(fp_origin_links,           "%d\t%d\t%d\t%d\t%lf\t%d\t%d\t%f\t%ld\n",
&convert_link.sno[i],                                                       &convert_link.linkid[i],
&convert_link.anode[i],&convert_link.bnode[i],&convert_link.mile[i],&convert_link.fclass,&convert_link.delay[i],
&convert_link.speed[i],&convert_link.link_id[i]);
                                  i++;
                          }
                          fclose(fp_origin_links);
                          for (m=0;m<origin_final_output[final_exit-1].count; m++)
                          {
                                  for(j=0;j<i;j++)
                                  {
                                          if(origin_final_output[final_exit-1].Final_Links_origin[m]        ==
convert_link.linkid[j])
                                          {
                                                  display_links_origin[m]=convert_link.link_id[j];
                                                  break;
                                          }
                                  }
                          }
              // DISPLAY
        /*                    for (m=0;m<origin_final_output[final_exit-1].count-1; m++)
                          {
                  //      cout<<"Links\t\t"<<display_links_origin[m+1]<<endl;
                  //      cout<<"Nodes\t\t"<<display_nodes_origin[m+1]<<endl;
                  //      cout<<"Dist\t\t"<<origin_final_output[final_exit-
1].Final_Dist_origin[m+1]<<endl;
                  //      cout<<"Time\t\t"<<origin_final_output[final_exit-
1].Final_Time_origin[m+1]<<endl;
                          }
        */
              //DISPLAY NODES FROM EXIT TO DESTINATION
                          fp_macro=fopen(file_macro,"r");
                          if (fp_macro == NULL)
                          {
                                  cout<<"Could not open file for reading"<<endl;
                          }

//                        cout<<endl;

                          i=0;
                          m=0;
                          while (feof(fp_macro) == 0)
                          {
                                  fscanf(fp_macro,      "%d\t%d\t%ld\t%d\t%ld\n",      &convert_node.sno[i],
&convert_node.node_id[i], &convert_node.node[i],&convert_node.key[i],&convert_node.regn[i]);
                                  i++;
                          }
                          fclose(fp_macro);
                          Cnt_Interm=interm_final_output[final_exit-1][final_entry-1].count;
//      cout<<"===>>   Count in MAIN FUNCTION FOR CASE 3 INTERM   is:   "<< interm_final_output[final_exit-
1][final_entry-1].count <<endl;
```

-A-63-

AFOSR F49620-01-1-0371

```
                                    for (m=0;m<interm_final_output[final_exit-1][final_entry-1].count; m++)
                                    {
                                            for(j=0;j<i;j++)
                                            {
                                                    if(interm_final_output[final_exit-1][final_entry-
1].Final_Nodes_interm[m] == convert_node.node_id[j])
                                                    {
                                                            display_nodes_interm[m]=convert_node.node[j];
                                                            break;
                                                    }
                                            }
                                    }
                  //DISPLAY LINKS FROM EXIT TO DESTINATION
                                    sprintf(file_interm_links,"link99.txt");
                                    fp_interm_links=fopen(file_interm_links,"r");
                                    if (fp_interm_links == NULL)
                                    {
                                            cout<<"Could not open file for reading"<<endl;
                                    }
                                    i=0;
                                    m=0;
                                    while (feof(fp_interm_links) == 0)
                                    {
                                            fscanf(fp_interm_links,          "%d\t%d\t%d\t%d\t%lf\t%d\t%d\t%f\t%ld\n",
&convert_link.sno[i],                                                         &convert_link.linkid[i],
&convert_link.anode[i],&convert_link.bnode[i],&convert_link.mile[i],&convert_link.fclass,&convert_link.delay[i],
&convert_link.speed[i],&convert_link.link_id[i]);
                                            i++;
                                    }
                                    fclose(fp_interm_links);
                                    for (m=0;m<interm_final_output[final_exit-1][final_entry-1].count; m++)
                                    {
                                            for(j=0;j<i;j++)
                                            {
                                                    if(interm_final_output[final_exit-1][final_entry-
1].Final_Links_interm[m] == convert_link.linkid[j])
                                                    {
                                                            display_links_interm[m]=convert_link.link_id[j];
                                                            break;
                                                    }
                                            }
                                    }
                   // DISPLAY
     /*                        for (m=0;m<interm_final_output[final_exit-1][final_entry-1].count-1; m++)
                                    {
                           //       cout<<"Links\t\t"<<display_links_interm[m+1]<<endl;
                           //       cout<<"Nodes\t\t"<<display_nodes_interm[m+1]<<endl;
                           //       cout<<"Dist\t\t"<<interm_final_output[final_exit-1][final_entry-
1].Final_Dist_interm[m+1]<<endl;
                           //       cout<<"Time\t\t"<<interm_final_output[final_exit-1][final_entry-
1].Final_Time_interm[m+1]<<endl;
                                    }
     */
```

```
                               } // END ELSE LOOP
                    } ////////////////////////////////////// END IF LOOP3
         //*******************************************************
         //      CASE 4: ORIGIN AND DESTINATION ARE IN DIFFERENT REGIONS
                    else if (origin_region!= destin_region)
                    {
out = "CASE 4: ORIGIN AND DESTINATION ARE IN DIFFERENT REGIONS";
localFederate->logMessage(out);
out = "Origin region = ";
out += _itoa(origin_region, dummy, 10);
out += " and Destination region = ";
out += _itoa(destin_region, dummy, 10);
localFederate->logMessage(out);
                         sprintf(fnode,"node%d.txt",origin_region);
                         // FINDING THE SP FROM ORIGIN TO EXIT POINTS
out = "FINDING THE SP FROM ORIGIN TO EXIT POINTS";
localFederate->logMessage(out);
                         sprintf(flink,"link%d.txt",origin_region);
                         sprintf(fadjn,"adjacency%d.txt",origin_region);
                         sprintf(fadjx,"index%d.txt",origin_region);
out = "Link file name = ";
out += flink;
localFederate->logMessage(out);
out = "Adjacency file name = ";
out += fadjn;
localFederate->logMessage(out);
out = "Index file name = ";
out += fadjx;
localFederate->logMessage(out);
                         INTEGRATION *Beyza=new INTEGRATION;
                         assert(Beyza != NULL);
                         Beyza->ReadNode(fnode);
                         Beyza->ReadLink(flink);
                         Beyza->ReadMtrx(fadjn,fadjx);

                         origin_destin(fp_origin,file_origin);
                         exit_points=number_entry_exit;                 // number_entry_exit set in origin_destin
                         assert(exit_points < 50);                      // larger will overflow array Exit_origin
                         for(n=0;n<number_entry_exit;n++)
                         {
                              Exit_origin[n]= Exit_Entry[n];
                              CallYen(Beyza,origin_actual,Exit_origin[n]);
                              origin_final_output[n].Path_Distance_origin_exit = Beyza->GetFinalDistance()/1600;
                              assert(Beyza->Node_Count < 500);// larger will overflow origin_final_output member
arrays
                              for(i=0;i<Beyza->Node_Count;i++)
                              {
                                   origin_final_output[n].Final_Nodes_origin[i]=Beyza->Final_Path[i];
                                   origin_final_output[n].Final_Links_origin[i]=Beyza->Final_Path1[i];
                                   origin_final_output[n].Final_Dist_origin[i]=Beyza->Final_Path2[i];
                                   origin_final_output[n].Final_Time_origin[i]=Beyza->Final_Path3[i];
                              }
                              origin_final_output[n].count=i;
```

-A-65-

```
                        }
                        delete Beyza;
        //Convert Exit points for Macro Network
                        fp_origin=fopen(file_origin,"r");
                        if (fp_origin == NULL)
                        {
                                cout<<"Could not open file for reading"<<endl;
                        }
                        i=0;
                        while (!feof(fp_origin))
                        {
                                fscanf(fp_origin,          "%d\t%d\t%ld\t%d\t%ld\n",          &convert_node.sno[i],
&convert_node.node_id[i], &convert_node.node[i],&convert_node.key[i],&convert_node.regn[i]);
                                i++;
                        }
out = "Number of records read from ";
out += file_origin;
out += "is ";
out += _itoa(i, dummy, 10);
localFederate->logMessage(out);
                        fclose(fp_origin);
                        assert(number_entry_exit < 50);                 // larger will overflow Exit_origin
                        for(n=0;n<number_entry_exit;n++)
                        {
                                assert(i < 7500);                              // larger will overflow convert_node
member arrays
                                for(j=0;j<i;j++)
                                {
                                        if(Exit_origin[n]==convert_node.node_id[j])
                                        {
                                                Macro_Enter[n] = convert_node.node[j];
                                                break;
                                        }
                                }
                        }
        //      Convert back for Macro Network
                        sprintf(file_macro,"node%d.txt",99);
                        fp_macro=fopen(file_macro,"r");
                        if (fp_macro == NULL)
                        {
                                cout<<"Could not open file for reading"<<endl;
                        }
                        i=0;
                        while (!feof(fp_macro))
                        {
                                fscanf(fp_macro,          "%d\t%d\t%ld\t%d\t%ld\n",          &convert_node.sno[i],
&convert_node.node_id[i], &convert_node.node[i],&convert_node.key[i],&convert_node.regn[i]);
                                i++;
                        }
out = "Number of records read from ";
out += file_macro;
out += "is ";
out += _itoa(i, dummy, 10);
```

```
localFederate->logMessage(out);

                        fclose(fp_macro);
                        exit_points=0; // *********** CHANGED 03/23/2006
                        for(n=0;n<number_entry_exit;n++)
                        {
                                for(j=0;j<i;j++)
                                {
                                        if(Macro_Enter[n]==convert_node.node[j])
                                        {
                                                Enter[exit_points] = convert_node.node_id[j];
                                                ++exit_points; // *********** CHANGED 03/23/2006
                                                break;
                                        }
                                }
                        }
        // FINDING THE SP FROM ENTRY POINTS TO DESTINATION
                        sprintf(fnode,"node%d.txt",destin_region);
                        sprintf(flink,"link%d.txt",destin_region);
                        sprintf(fadjn,"adjacency%d.txt",destin_region);
                        sprintf(fadjx,"index%d.txt",destin_region);
                        Beyza =new INTEGRATION;
                        Beyza->ReadNode(fnode);
                        Beyza->ReadLink(flink);
                        Beyza->ReadMtrx(fadjn,fadjx);

                        origin_destin(fp_destin,file_destin);
                        entry_points=number_entry_exit;
                        for(n=0;n<number_entry_exit;n++)
                        {
                                Entry_destin[n]= Exit_Entry[n];
                                CallYen(Beyza,Entry_destin[n],destin_actual);
                                destin_final_output[n].Path_Distance_destin_entry             =             Beyza-
>GetFinalDistance()/1600;
                                for(i=0;i<Beyza->Node_Count;i++)
                                {
                                        destin_final_output[n].Final_Nodes_destin[i]=Beyza->Final_Path[i];
                                        destin_final_output[n].Final_Links_destin[i]=Beyza->Final_Path1[i];
                                        destin_final_output[n].Final_Dist_destin[i]=Beyza->Final_Path2[i];
                                        destin_final_output[n].Final_Time_destin[i]=Beyza->Final_Path3[i];
                                }
                                destin_final_output[n].count=i;
                        }
                        delete Beyza;
        //Convert Entry points for Macro Network
                        fp_destin=fopen(file_destin,"r");
                        if (fp_destin == NULL)
                        {
                                cout<<"Could not open file for reading"<<endl;
                        }
                        i=0;
                        while (feof(fp_destin) == 0)
                        {
```

```
                        fscanf(fp_destin,            "%d\t%d\t%ld\t%d\t%ld\n",          &convert_node.sno[i],
&convert_node.node_id[i], &convert_node.node[i],&convert_node.key[i],&convert_node.regn[i]);
                        i++;
                }
                fclose(fp_destin);
                for(n=0;n<number_entry_exit;n++)
                {
                        for(j=0;j<i;j++)
                        {
                                if(Entry_destin[n]==convert_node.node_id[j])
                                {
                                        Macro_Exit[n] = convert_node.node[j];
                                        break;
                                }
                        }
                }

    //      Convert back for Macro Network

                sprintf(file_macro,"node%d.txt",99);
                fp_macro=fopen(file_macro,"r");
                if (fp_macro == NULL)
                {
                        cout<<"Could not open file for reading"<<endl;
                }
                i=0;

                while (feof(fp_macro) == 0)
                {
                        fscanf(fp_macro,            "%d\t%d\t%ld\t%d\t%ld\n",          &convert_node.sno[i],
&convert_node.node_id[i], &convert_node.node[i],&convert_node.key[i],&convert_node.regn[i]);
                        i++;
                }

                fclose(fp_macro);
                entry_points=0;// *********** CHANGED 03/23/2006
                for(n=0;n<number_entry_exit;n++)
                {
                        for(j=0;j<i;j++)
                        {
                                if(Macro_Exit[n]==convert_node.node[j])
                                {
                                        Exit[entry_points] = convert_node.node_id[j];
                                        ++entry_points;// *********** CHANGED 03/23/2006
                                        break;
                                }
                        }
                }

    //FINDING THE SP FROM ENTRY POINTS TO EXIT POINTS
                sprintf(fnode,"node99.txt");
                sprintf(flink,"link99.txt");
                sprintf(fadjn,"adjacency99.txt");
```

-A-68-

```
                    sprintf(fadjx,"index99.txt");
                    Beyza =new INTEGRATION;
                    Beyza->ReadNode(fnode);
                    Beyza->ReadLink(flink);
                    Beyza->ReadMtrx(fadjn,fadjx);

                    for(n=0;n<exit_points;n++)
                    {
                            for(m=0;m<entry_points;m++)
                            {
                                    CallYen(Beyza,Enter[n],Exit[m]);
                                    interm_final_output[n][m].Path_Distance_interm            =            Beyza-
>GetFinalDistance()/1600;
                                    for(i=0;i<Beyza->Node_Count;i++)
                                    {
                                            interm_final_output[n][m].Final_Nodes_interm[i]=            Beyza-
>Final_Path[i];
                                            interm_final_output[n][m].Final_Links_interm[i]=            Beyza-
>Final_Path1[i];
                                            interm_final_output[n][m].Final_Dist_interm[i]=            Beyza-
>Final_Path2[i];
                                            interm_final_output[n][m].Final_Time_interm[i]=            Beyza-
>Final_Path3[i];
                                    }
                                    interm_final_output[n][m].count=i;
                            }
                    }
                    delete Beyza;
                    Shortest_Path = GetShortestDistance();
//            cout<<"Distance from  "<<origin<<"  to  "<<destination <<"  is  " << Shortest_Path<<"
miles"<<endl;
                    cout<<"100\t"<<"100\t"<<Shortest_Path<<endl;
                    if(Shortest_Path > (30)||(Shortest_Path<0))
                    {
//            cout<<"  ==>>  Inside MAIN:  THE DESTINATION IS UNREACHABLE  <<==   "<<endl;
                    }

            //DISPLAY NODES FROM ORIGIN TO EXIT
                    else
                    {
                            fp_origin=fopen(file_origin,"r");
                            if (fp_origin == NULL)
                            {
                                    cout<<"Could not open file for reading"<<endl;
                            }

//                    cout<<endl;

                            i=0;
                            m=0;
                            while (feof(fp_origin) == 0)
                            {
```

-A-69-

```
                              fscanf(fp_origin,        "%d\t%d\t%ld\t%d\t%ld\n",        &convert_node.sno[i],
&convert_node.node_id[i], &convert_node.node[i],&convert_node.key[i],&convert_node.regn[i]);
                                         i++;
                              }
                              fclose(fp_origin);
                              Cnt_Exit = origin_final_output[final_exit-1].count;
//      cout<<"===>>  Count in MAIN FUNCTION FOR CASE 4 ORIGIN is:  "<< origin_final_output[final_exit-1].count
<<endl;
                              for (m=0;m<origin_final_output[final_exit-1].count; m++)
                              {
                                         for(j=0;j<i;j++)
                                         {
                                                    if(origin_final_output[final_exit-1].Final_Nodes_origin[m]        ==
convert_node.node_id[j])
                                                    {
                                                               display_nodes_origin[m]=convert_node.node[j];
                                                               break;
                                                    }
                                         }

                              }
           //        DISPLAY LINKS FROM ORIGIN TO EXIT
                              sprintf(file_origin_links,"link%d.txt",origin_region);
                              fp_origin_links=fopen(file_origin_links,"r");
                              if (fp_origin_links == NULL)
                              {
                                         cout<<"Could not open file for reading"<<endl;
                              }
                              i=0;
                              m=0;
                              while (feof(fp_origin_links) == 0)
                              {
                                         fscanf(fp_origin_links,        "%d\t%d\t%d\t%d\t%lf\t%d\t%d\t%f\t%ld\n",
&convert_link.sno[i],                                                   &convert_link.linkid[i],
&convert_link.anode[i],&convert_link.bnode[i],&convert_link.mile[i],&convert_link.fclass,&convert_link.delay[i],
&convert_link.speed[i],&convert_link.link_id[i]);
                                         i++;
                              }
                              fclose(fp_origin_links);
                              for (m=0;m<origin_final_output[final_exit-1].count; m++)
                              {
                                         for(j=0;j<i;j++)
                                         {
                                                    if(origin_final_output[final_exit-1].Final_Links_origin[m]        ==
convert_link.linkid[j])
                                                    {
                                                               display_links_origin[m]=convert_link.link_id[j];
                                                               break;
                                                    }
                                         }
                              }
              // DISPLAY
       /*                 for (m=0;m<origin_final_output[final_exit-1].count-1; m++)
```

-A-70-

AFOSR F49620-01-1-0371

```
                                    {
                            //        cout<<"Links\t\t"<<display_links_origin[m+1]<<endl;
                            //        cout<<"Nodes\t\t"<<display_nodes_origin[m+1]<<endl;
                            //        cout<<"Dist\t\t"<<origin_final_output[final_exit-
1].Final_Dist_origin[m+1]<<endl;
                            //        cout<<"Time\t\t"<<origin_final_output[final_exit-
1].Final_Time_origin[m+1]<<endl;
                                    }
        */
              //*****************************************************
              //DISPLAY NODES FROM EXIT TO ENTRY
                            fp_macro=fopen(file_macro,"r");
                            if (fp_macro == NULL)
                            {
                                    cout<<"Could not open file for reading"<<endl;
                            }

//                            cout<<endl;

                            i=0;
                            m=0;
                            while (feof(fp_macro) == 0)
                            {
                                    fscanf(fp_macro,      "%d\t%d\t%ld\t%d\t%ld\n",      &convert_node.sno[i],
&convert_node.node_id[i], &convert_node.node[i],&convert_node.key[i],&convert_node.regn[i]);
                                    i++;
                            }
                            fclose(fp_macro);
                            Cnt_Interm = interm_final_output[final_exit-1][final_entry-1].count;
//      cout<<"===>>   Count  in  MAIN  FUNCTION  FOR  CASE 4  INTERM  is:   "<<  interm_final_output[final_exit-
1][final_entry-1].count <<endl;
                            for (m=0;m<interm_final_output[final_exit-1][final_entry-1].count; m++)
                            {
                                    for(j=0;j<i;j++)
                                    {
                                            if(interm_final_output[final_exit-1][final_entry-
1].Final_Nodes_interm[m] == convert_node.node_id[j])
                                            {
                                                    display_nodes_interm[m]=convert_node.node[j];
                                                    break;
                                            }
                                    }
                            }
              //DISPLAY LINKS FROM EXIT TO ENTRY
                            sprintf(file_interm_links,"link99.txt");
                            fp_interm_links=fopen(file_interm_links,"r");
                            if (fp_interm_links == NULL)
                            {
                                    cout<<"Could not open file for reading"<<endl;
                            }
                            i=0;
                            m=0;
                            while (feof(fp_interm_links) == 0)
```

-A-71-

```
                               {
                                       fscanf(fp_interm_links,        "%d\t%d\t%d\t%d\t%lf\t%d\t%d\t%f\t%ld\n",
&convert_link.sno[i],                                               &convert_link.linkid[i],
&convert_link.anode[i],&convert_link.bnode[i],&convert_link.mile[i],&convert_link.fclass,&convert_link.delay[i],
&convert_link.speed[i],&convert_link.link_id[i]);
                                       i++;
                               }
                               fclose(fp_interm_links);
                               for (m=0;m<interm_final_output[final_exit-1][final_entry-1].count; m++)
                               {
                                       for(j=0;j<i;j++)
                                       {
                                               if(interm_final_output[final_exit-1][final_entry-
1].Final_Links_interm[m] == convert_link.linkid[j])
                                               {
                                                       display_links_interm[m]=convert_link.link_id[j];
                                                       break;
                                               }
                                       }
                               }
                   // DISPLAY
         /*                     for (m=0;m<interm_final_output[final_exit-1][final_entry-1].count-1; m++)
                               {
                               //      cout<<"Links\t\t"<<display_links_interm[m+1]<<endl;
                               //      cout<<"Nodes\t\t"<<display_nodes_interm[m+1]<<endl;
                               //      cout<<"Dist\t\t"<<interm_final_output[final_exit-1][final_entry-
1].Final_Dist_interm[m+1]<<endl;
                               //      cout<<"Time\t\t"<<interm_final_output[final_exit-1][final_entry-
1].Final_Time_interm[m+1]<<endl;
                               }
         */
                   //**************************************************
                   //DISPLAY NODES FROM ENTRY TO DESTINATION
                               fp_destin=fopen(file_destin,"r");
                               if (fp_destin == NULL)
                               {
                                       cout<<"Could not open file for reading"<<endl;
                               }

//                             cout<<endl;

                               i=0;
                               m=0;
                               while (feof(fp_destin) == 0)
                               {
                                       fscanf(fp_destin,     "%d\t%d\t%ld\t%d\t%ld\n",      &convert_node.sno[i],
&convert_node.node_id[i], &convert_node.node[i],&convert_node.key[i],&convert_node.regn[i]);
                                       i++;
                               }
                               fclose(fp_destin);
                               Cnt_Entry = destin_final_output[final_entry-1].count;
//            cout<<"===>>  Count in MAIN FUNCTION FOR CASE 4 Entry is:   "<< destin_final_output[final_entry-
1].count <<endl;
```

```
                        for (m=0;m<destin_final_output[final_entry-1].count; m++)
                        {
                                for(j=0;j<i;j++)
                                {
                                        if(destin_final_output[final_entry-1].Final_Nodes_destin[m]      ==
convert_node.node_id[j])
                                        {
                                                display_nodes_destin[m]=convert_node.node[j];
                                                break;
                                        }
                                }
                        }

            //DISPLAY LINKS FROM ENTRY TO DESTINATION
                        sprintf(file_destin_links,"link%d.txt",destin_region);
                        fp_destin_links=fopen(file_destin_links,"r");
                        if (fp_destin_links == NULL)
                        {
                                cout<<"Could not open file for reading"<<endl;
                        }
                        i=0;
                        m=0;
                        while (feof(fp_destin_links) == 0)
                        {
                                fscanf(fp_destin_links,        "%d\t%d\t%d\t%d\t%lf\t%d\t%d\t%f\t%ld\n",
&convert_link.sno[i],                                                &convert_link.linkid[i],
&convert_link.anode[i],&convert_link.bnode[i],&convert_link.mile[i],&convert_link.fclass,&convert_link.delay[i],
&convert_link.speed[i],&convert_link.link_id[i]);
                                        i++;
                        }
                        fclose(fp_destin_links);
                        for (m=0;m<destin_final_output[final_entry-1].count; m++)
                        {
                                for(j=0;j<i;j++)
                                {
                                        if(destin_final_output[final_entry-1].Final_Links_destin[m]      ==
convert_link.linkid[j])
                                        {
                                                display_links_destin[m]=convert_link.link_id[j];
                                                break;
                                        }
                                }
                        }
            //              cout<<"Final entry value in the display is:  "<<final_entry<<endl;
            // DISPLAY
      /*                    for (m=0;m<destin_final_output[final_entry-1].count-1; m++)
                        {
            //      cout<<"Links\t\t"<<display_links_destin[m+1]<<endl;
            //      cout<<"Nodes\t\t"<<display_nodes_destin[m+1]<<endl;
            //      cout<<"Dist\t\t"<<destin_final_output[final_entry-
1].Final_Dist_destin[m+1]<<endl;
            //      cout<<"Time\t\t"<<destin_final_output[final_entry-
1].Final_Time_destin[m+1]<<endl;
```

```
                                   }
        */
                    } // END ELSE LOOP
                } ///////////////////////////////////////////////////////////END IF LOOP4
        }// LOOP TO CHECK IF NODE IS UNLISTED IN DATABASE
        else
        {
//              cout<<"Nodes are unlisted: NOT IN DATABASE"<<endl;
                Shortest_Path=90;
        }
}        /////////////////////////****************************************************************END MAIN
//********************     Applying Algorithm********************//

void CallYen(INTEGRATION* Beyza,int org,int dst)
{
        remove("Solution/KPathDistance.txt");
        remove("Solution/KPathBriefNodes.txt");
        char ofname[100];
        switch(opt)
                {
                case 1:
                        sprintf(ofname,"%s","Solution/KPathDistance.txt\0");
                        Beyza->ofpath.open(ofname,ios::app);
                        break;
                case 2:
                        sprintf(ofname,"%s","Solution/KPathTime.txt\0");
                        Beyza->ofpath.open(ofname,ios::app);
                        break;
                        default:
                        return;
                }

        Beyza->ResetSpath();
        Beyza->ResetKpath();
        Beyza->SetPair(org,dst,kpath);
        Beyza->Dijkstra(org,dst,opt);
        Beyza->BuildPath(org,dst);
        Beyza->FindLabel();
        Beyza->DisplayPath(1,opt);
        Beyza->NodeClean();
        Beyza->AddPathA(Beyza->Final);
        Beyza->PathClean();
        Beyza->ofpath.close();
        Beyza->DisplayNodes(Beyza->A, Beyza->NoA, 0,  "KPathBriefNodes.txt");
        Beyza->ResetSpath();
        Beyza->ResetKpath();
}


void origin_destin(FILE *fp_origin_destin, char file_origin_destin[200] )
{
        fp_origin_destin=fopen(file_origin_destin,"r");
        if (fp_origin_destin == NULL)
        {
```

```cpp
                cout<<"Could not open file for reading"<<endl;
        }
        i=0;
        m=0;


        while (feof(fp_origin_destin) == 0)
        {
                fscanf(fp_origin_destin,              "%d\t%d\t%ld\t%d\t%ld\n",          &convert_node.sno[i],
&convert_node.node_id[i], &convert_node.node[i],&convert_node.key[i],&convert_node.regn[i]);
                if(convert_node.regn[i]!=99 && convert_node.key[i] == 1)
                {
                        Exit_Entry[m] = convert_node.node_id[i];
                        m++;
                }
                i++;
        }
        fclose(fp_origin_destin);
        number_entry_exit=m;
}


double GetShortestDistance()
        {

        double  distance;
        int             i,j;
        double  distance_temp = 90000.0;
        for(i=0;i<exit_points;i++)
        {
                for(j=0;j<entry_points;j++)
                {
                        distance=0;
                        distance           =          origin_final_output[i].Path_Distance_origin_exit          +
interm_final_output[i][j].Path_Distance_interm+destin_final_output[j].Path_Distance_destin_entry;
                        if(distance_temp>distance)
                        {
                                distance_temp = distance;
                                final_exit = i+1;
                                final_entry = j+1;
                        }
                }
        }
        return distance_temp;
        }
};


#ifndef ROADDAMAGEDEFINITION_01
#define ROADDAMAGEDEFINITION_01
struct Road_Damage
{
        unsigned long  time;
        unsigned int   Link_ID;
        unsigned int   Severity;      // Damage Level
        unsigned int   Delay;
```

```cpp
        double Link_Delay;
        Road_Damage(void)
        {
                time = 0;
                Link_ID = 0;
                Severity = 0;
                Delay = 0;
                Link_Delay=0.0;
        }
};
#endif
#ifndef ROADDELAYDEFINITION_01
#define ROADDELAYDEFINITION_01
struct  Road_Delay
{
        unsigned long  time;
        unsigned int   Link_ID;
        unsigned int   Delay;
        Road_Delay(void)
        {
                time = 0;
                Link_ID = 0;
                Delay = 0;
        }
};
#endif
#ifndef h_TIMESTAMPGENERATOR_0001
#define h_TIMESTAMPGENERATOR_0001


/////////////////////////////
// The debugger can't handle symbols more than 255 characters long.
// STL often creates symbols longer than that.
// When symbols are longer than 255 characters, the warning is issued.
#pragma warning(disable:4786)
/////////////////////////////
#include <string>
#include <ctime>
class timeStamp
{
public:
        timeStamp(void)
        {     }
        ~timeStamp(void)
        {     }
        std::string stamp(void)
        {
                time_t clock;
                struct tm *TimeDateStruct;
                char *TimeDateString;
                char dbNameString[26];
                time(&clock);
                TimeDateStruct = localtime(&clock);
```

```
                TimeDateString = asctime(TimeDateStruct);
                strncpy(dbNameString+0, TimeDateString+8, 2);
                strncpy(dbNameString+2, TimeDateString+4, 3);
                strncpy(dbNameString+5, TimeDateString+22, 2);
                *(dbNameString+7) = '_';
                strncpy(dbNameString+8, TimeDateString+11, 2);
                strncpy(dbNameString+10, TimeDateString+14, 2);
                strncpy(dbNameString+12, TimeDateString+17, 2);
                *(dbNameString+14) = '\0';
                return(dbNameString);
        }
};
#endif
#ifndef h_CRITICALREGIONINFO_0001
#define h_CRITICALREGIONINFO_0001
#include <vector>
#include <string>
#include <queue>
//#include "typedefs.h"
class tsQueue : public std::queue<std::string> // Threadsafe access to data
{
public:
    tsQueue( )                                        // Constructor
        {
                // Create the mutex which serializes all access to our data
                m_Mutex = CreateMutex( NULL, false, "SerializeAccess" );
                if ( m_Mutex == NULL )
                {
                                                        // This will crash the program – what
else to do?
                        exit(9);
                }
        }
    virtual ~tsQueue( )                               // Destructor
        {
                CloseHandle(m_Mutex);              // Clean up
        }

    void insert(const std::string & s)       // Add a member at the end
        {
                getAccessToData();                    // Check for non-destructive access
                this->push(s);                        // Add data-element to vector
                m_TotalMemberCount += 1;          // add to total count
                if (m_HighWaterMark < this->queueSize())
                {
                        m_HighWaterMark = this->queueSize();
                }
                releaseAccessToData();             // Give up control of data
        }
    std::string extract(void)               // Pull a member from the front
        {
                std::string retValue;              // What we'll return to caller
                getAccessToData();                    // Check for non-destructive access
```

```
                if (!this->empty())                              // make sure something is there
                {
                        retValue = this->front();       // Grab requested element
                        this->pop();                             // remove element from queue
                }
                releaseAccessToData();                  // Give up control of data
                return retValue;                         // Send removed value to caller
        }
    unsigned long queueSize(void)            // How many members in queue
        {
                unsigned long retValue;                  // What we'll return to caller
                getAccessToData();                       // Check for non-destructive access
                retValue = this->size();         // Get number of elements in vector
                releaseAccessToData();           // Give up control of data
                return retValue;                         // Send count to caller
        }
        unsigned long getHighWaterMark(void)
        {
                return m_HighWaterMark;
        }
        unsigned long getTotalMemberCount(void)
        {
                return m_TotalMemberCount;
        }

private:
    HANDLE                  m_Mutex;              // Serialize access to this data
        unsigned long  m_HighWaterMark;       // instantaneous largest number of members
        unsigned long  m_TotalMemberCount;   // total number of insertions
    void getAccessToData(void)                       // Grab sole control
        {
                                                                    // Request access to data
                WaitForSingleObject(m_Mutex, INFINITE);
        }
    void releaseAccessToData(void)           // Release sole control
        {
                ReleaseMutex(m_Mutex);                   // Release access to data
        }

        tsQueue(const tsQueue &dS)                       // copy constructor
        {
         // private member: cannot be copied
    }
        tsQueue &operator=(const tsQueue &dS)// assignment
        {
         // private member: cannot be assigned
    }
};
#endif
#pragma warning(disable:4786)
#include "Integration.h"
#include <vector>
#include "Road_Damage.h"
```

```cpp
#include "Ambulance_Idle.h"
#include "localFederate.h"
INTEGRATION::INTEGRATION()
{
        NoNd=0;
        NoLn=0;
        NoAj=0;
        tstart=0;
        Head=NULL;
        size=0;
        NodeCnt=0;
        FinLabel=0;
        NoA=NoB=KPATH=0;
        A=NULL;
        B=NULL;
        SP=NULL;
        HELP=ROOT=SPUR=Final=NULL;
}
void INTEGRATION::Reset()
{
        DeletePathNode(Head);
        Head=Last=NULL;
        size=0;
        NodeCnt=0;
        FinLabel=0;
        tstart=0;
        NoA=NoB=KPATH=0;
        DeletePathList(A);
        DeletePathList(B);
        DeletePathList(SP);
        A=NULL;
        B=NULL;
        SP=NULL;
        DeleteFinalPath(HELP);
        DeleteFinalPath(ROOT);
        DeleteFinalPath(SPUR);
        DeleteFinalPath(Final);
        HELP=NULL;
        ROOT=NULL;
        SPUR=NULL;
        Final=NULL;
}


void INTEGRATION::SetStartTime(double start_time)
{
        tstart=start_time;
}
void INTEGRATION::ReadNode(char fnode1[50])
{

        int i;
        char fname[50],ln[150];
        ifstream in;
```

-A-79-

AFOSR F49620-01-1-0371

```
        sprintf(fname,fnode1);
        in.open(fname);
        while (!in.eof())
        {
                in>>ln;
                if (ln[0]>' ')
                        NoNd++;
                in.getline(ln,150);
        }
        in.close();
        node=new NODE [NoNd];
        in.open(fname);
        for (i=0;i<NoNd;i++)
        {
                in>>node[i].NODEID;
                in>>node[i].FEATUREID;
                in>>node[i].NODE_ID;
                in>>node[i].KEY;
                in>>node[i].REGION;
                in.getline(node[i].DESCRIPT,150);
        }
        in.close();
}
void INTEGRATION::ReadLink(char flink1[100])
{
        int i,j;
        char fname[50],ln[150];
        extern fedModel *localFederate;
        ifstream in;
        sprintf(fname,flink1);
        in.open(fname);
        while (!in.eof())
        {
                in>>ln;
                if (ln[0]>' ')
                        NoLn++;
                in.getline(ln,150);
        }
        in.close();
        link=new LINK [NoLn];
        int ucnt=0,scnt=0,icnt=0,rcnt=0,xcnt=0;
        ifstream in1;
        in1.open(fname);
        for (i=0;i<NoLn;i++)
        {
                in1>>link[i].LINKID;
                in1>>link[i].FEATUREID;
                in1>>link[i].ANODE;
                in1>>link[i].BNODE;
                in1>>link[i].MILE;
                in1>>link[i].FCLASS;
                in1>>link[i].DELAY;
                in1>>link[i].SPEED;
```

```
                in1>>link[i].LINK_ID;
                in1.getline(link[i].DESCRIPT,150);
                if (link[i].FCLASS==0 || link[i].FCLASS==1 )
                {
                        link[i].SPEED=ISPEED;
                        icnt++;
                }
                else if (link[i].FCLASS==2)
                {
                        link[i].SPEED=USPEED;
                        scnt++;
                }
                else  if (link[i].FCLASS==3 || link[i].FCLASS==4)
                {
                        link[i].SPEED=SSPEED;
                        ucnt++;
                }
                else if (link[i].FCLASS==5)
                {
                        link[i].SPEED=RSPEED;
                        rcnt++;
                }
                else
                {
                        link[i].SPEED=XSPEED;
                        xcnt++;
                }
        }
        in.close();
//***** See if it can be written in a better way *****//
        for (i=0;i<NoLn;i++)  // Changing length due to road damage
        {
                for (j=0;j<localFederate->v_rd.size();j++)
                {
                        if(link[i].LINK_ID==localFederate->v_rd.at(j)->Link_ID)
                        {
                                link[i].MILE=link[i].MILE*(localFederate->v_rd).at(j)->Link_Delay;
                                break;
                        }
                        else
                        {
                                link[i].MILE=link[i].MILE;
                        }
                }
        }
        for (i=0;i<NoLn;i++)// Changing length due to ambulance stuck
        {
                for (j=0;j<localFederate->v_ai.size();j++)
                {
                        if(link[i].LINK_ID==localFederate->v_ai.at(j)->Link_ID)
                        {
                                link[i].MILE=INF;
                                break;
```

AFOSR F49620-01-1-0371

```
                     }
                     else
                     {
                             link[i].MILE=link[i].MILE;
                     }
               }
         }
}
void INTEGRATION::ReadMtrx(char fadjn1[100],char fadjx1[100])
{
       int i;
       char fname[50],ln[150];
       ifstream *in;
       sprintf(fname,fadjn1);
       adjn=new int [NoNd];
       in = new ifstream(fname);
       for (i=0;i<NoNd;i++)
       {
               (*in)>>adjn[i]>>adjn[i];
       }
       in->close();
       delete in;
       sprintf(fname,fadjx1);
       in = new ifstream(fname);
       while (!in->eof())
       {
               (*in)>>ln;
               if (ln[0]>' ')
                       NoAj++;
               in->getline(ln,150);
       }
       in->close();
       delete in;
       ndix=new int [NoAj];
       lnix=new int [NoAj];
       in = new ifstream(fname);
       for (i=0;i<NoAj;i++)
       {
               (*in)>>ndix[i];
               (*in)>>lnix[i];
       }
       in->close();
       delete in;
}
void INTEGRATION::DeletePathNode(PathNode* pathnode)
{
       PathNode *dpn=pathnode;
       while (pathnode!=NULL)
       {
               pathnode=pathnode->next;
               dpn->next=dpn->prev=NULL;
               delete dpn;
               dpn=pathnode;
```

-A-82-

```
        }
}
void INTEGRATION::DeleteFinalPath(FinalPath* finalpath)
{
        FinalPath *dfp=finalpath;
        while (finalpath!=NULL)
        {
                finalpath=finalpath->next;
                dfp->next=NULL;
                delete dfp;
                dfp=finalpath;
        }
}
void INTEGRATION::DeletePathList(PathList* pathlist)
{
        PathList *dpl=pathlist;
        while (pathlist!=NULL)
        {
                pathlist=pathlist->next;
                dpl->next=NULL;
                if (dpl->nodcnt>0)
                {
                        delete [] dpl->nodix;
                        delete [] dpl->lnkix;
                        delete [] dpl->label;
                        delete [] dpl->dist;
                        delete [] dpl->time;
                }
                delete dpl;
                dpl=pathlist;
        }
}


INTEGRATION::~INTEGRATION()
{
        delete [] link;
        delete [] adjn;
        delete [] node;
        delete [] ndix;
        delete [] lnix;
        link= NULL;
        adjn=NULL;
        node=NULL;
        ndix=NULL;
        lnix=NULL;
//      cout<<"Object deleted!!!"<<endl;
}

#pragma warning(disable:4786)
#include "Integration.h"
void INTEGRATION::DisplayPath(int i, int opt)
{
//      int z;
```

```
        if (Final!=NULL)
        {
                FinalPath *current=Final;
                char disunit[20];
                switch(opt)
                {
                        case 1:
                                strcpy(disunit,"meters\0");
                                break;
                        case 2:
                                strcpy(disunit,"min\0");
                                break;
                                default:
                                return;
                }
                ofpath<<"\n Origin - Destination: "<<this->Origin<<" - "<<this->Destin;
                ofpath<<"     Objective: Minimize "<<OBJ[opt-1]<<endl;

                if (i==0)
                {
                        ofpath<<"Minimum "<<OBJ[opt-1]<<": "<<FinLabel<<" "<<disunit<<endl<<endl;
                }
                else
                {
                        ofpath<<"Path "<<i<<"     "<<OBJ[opt-1]<<": "<<FinLabel<<" "<<disunit<<endl<<endl;
                }

                ofpath<<" NODE"<<"\t"<<"  DISTANCE"<<"\t"<<" TIME"<<endl;

                i=0;
//              z=0;
                while (current!=NULL)
                {

                        ofpath<<current->nodix<<"\t"
                                <<current->dist<<"\t"
                                <<double(int(current->time*100)/100.0)<<"\t"
                                <<endl;
                        Final_Path[i]=current->nodix;
                        Final_Path1[i]=current->lnkix;
                        Final_Path2[i]=current->dist;
                        Final_Path3[i]=double(int(current->time*100)/100.0);///////
                        i++;
                        current=current->next;
                }
                ofpath<<"\n----------------------------------------------------\n";
        }
}
void INTEGRATION::DisplayNodes(PathList *PL, int No, int opt, char *cfname)
{
        char f[100];
        ofstream of;
        int i,pt;
```

```cpp
        PathList **ListArray=new PathList* [No];
        sprintf(f,"%s%s","Solution/",cfname);
        of.open(f,ios::app);
        pt=0;
        PathList *cr = PL;
        while (cr!=NULL)
        {
                ListArray[pt] = cr;
                cr=cr->next;
                pt++;
        }
        if (PL!=NULL)
        {
                of<<"\n \t\tRoutes for Path  " << this->Origin<<" to "<<this->Destin << endl;
                for (pt=No-1;pt>=0;pt--)
                {
                        of<<"\nPath "<<No-pt<<"\t";
                        Node_Count=ListArray[pt]->nodcnt;

                        for (i=0;i<ListArray[pt]->nodcnt;i++)
                        {
                                of<<ListArray[pt]->nodix[i]<<" " ;
                        }
                of<<endl;
                }
                of.close();
        }
        delete [] ListArray;
}
double INTEGRATION::GetFinalDistance()
{
        return FinLabel;
}


// DispTest.cpp : Defines the entry point for the console application.
//#include <iostream>
#pragma warning(disable:4786)
#include <string>
#include <vector>
#include "Casualty_Observation.h"
#include "Casualty_Pickup.h"
#include "Casualty_Delivery.h"
#include "Road_Damage.h"
#include "Hospital_Capacity.h"
#include "Road_Delay.h"
#include "Hospital_Delay.h"
#include "Ambulance_Idle.h"
#include "Ambulance_Stuck.h"
#include "Cluster_Identify.h"
#include "Main.h"
#include <Math.h>
//#include <iostream>
```

```
//#include <stdio.h>
//#include <stdlib.h>
#define MAX 90000
using namespace std;
class DispTest
{

public:
        DispTest()
        {
        }
        void parse_line(string s, vector<string> &words)
        {
                int pos = 0;
                int y = 0;
                for ( int ix = 0; ix < s.size(); ++ix )
                {
                        if ( s[ix]  == '~' || s[ix] == '\n' )
                        {
                                string test;
                                test = s.substr(pos, (ix-pos));
                                words.push_back(test);
                                pos = ix+1;
                        }
                }
                //cout << "Inside parseline: " << words.size() << endl;
                return;
        }
        void parse_File()
        {    // parse the file
                ifstream ist;
                ist.open("Input.txt");
                string w;
                while (ist>>w)
                {
                        vector< string > values;
                        parse_line(w, values);
                        /*for( int i=0; i<values.size(); ++i) {
                                        cout << "Inside CO" << values.at(i) << endl;
                        }*/
                        //convert each message into a struct variable
                        if(values.at(0) == "EDtoDP01")
                        {
                                //cout << "Type is " << values.size() << endl;
                                Casualty_Observation *co;
                                co = new Casualty_Observation();
                                co->time = atol(values.at(1).c_str());
                                co->TrackID = atol(values.at(2).c_str());
                                co->X = atof(values.at(3).c_str());
                                co->Y = atof(values.at(4).c_str());
                                co->Nearest_Node = atol(values.at(5).c_str());
                                co->Severity = atoi(values.at(6).c_str());
                                co->Sev_Prob_Vect[0] = atof(values.at(7).c_str());
```

-A-86-

```
        co->Sev_Prob_Vect[1] = atof(values.at(8).c_str());
        co->Sev_Prob_Vect[2] = atof(values.at(9).c_str());
        co->Sev_Prob_Vect[3] = atof(values.at(10).c_str());
        v_co.push_back(co);
        //TO_DO should reserve() be used in prior


}
else if(values.at(0) == "EDtoDP02")
{
        //cout << "Type is " << values.size() << endl;
        Casualty_Pickup *cp;
        cp = new Casualty_Pickup();
        cp->time = atol(values.at(1).c_str());
        cp->TrackID_1 = atol(values.at(2).c_str());
        cp->Nearest_Node_1 = atol(values.at(3).c_str());
        cp->Severity_1 = atoi(values.at(4).c_str());
        cp->TrackID_2 = atol(values.at(5).c_str());
        cp->Nearest_Node_2 = atol(values.at(6).c_str());
        cp->Severity_2 = atoi(values.at(7).c_str());
        cp->TrackID_3 = atol(values.at(8).c_str());
        cp->Nearest_Node_3 = atol(values.at(9).c_str());
        cp->Severity_3 = atoi(values.at(10).c_str());
        v_cp.push_back(cp);
}
else if(values[0] == "EDtoDP03")
{
        //cout << "Type is " << values.size() << endl;
        Casualty_Delivery *cd;
        cd = new Casualty_Delivery();
        cd->time = atol(values.at(1).c_str());
        cd->TrackID_1 = atol(values.at(2).c_str());
        cd->Hospital_1 = atol(values.at(3).c_str());
        cd->Severity_1 = atoi(values.at(4).c_str());
        cd->TrackID_2 = atol(values.at(5).c_str());
        cd->Hospital_2 = atol(values.at(6).c_str());
        cd->Severity_2 = atoi(values.at(7).c_str());
        cd->TrackID_3 = atol(values.at(8).c_str());
        cd->Hospital_3 = atol(values.at(9).c_str());
        cd->Severity_3 = atoi(values.at(10).c_str());
        v_cd.push_back(cd);
}
else if(values.at(0) == "EDtoDP04")
{
        //cout << "Type is " << values.size() << endl;
        Road_Damage *rd;
        rd = new Road_Damage();
        rd->time = atol(values.at(1).c_str());
        rd->Link_ID = atoi(values.at(2).c_str());
        rd->Severity = atoi(values.at(3).c_str());
        v_rd.push_back(rd);
}
else if(values.at(0) == "EDtoDP05")
{
```

```cpp
                //cout << "Type is " << values.size() << endl;
                Hospital_Capacity *hc;
                hc = new Hospital_Capacity();
                hc->time = atol(values.at(1).c_str());
                hc->HospitalID = atol(values.at(2).c_str());
                hc->Severity_2 = atoi(values.at(3).c_str());
                hc->Severity_3 = atoi(values.at(4).c_str());
                hc->X = atof(values.at(5).c_str());
                hc->Y = atof(values.at(6).c_str());
                hc->Nearest_Node = atol(values.at(7).c_str());
                v_hc.push_back(hc);
        }
        else if(values.at(0) == "EDtoDP06")
        {
                //cout << "Type is " << values.size() << endl;
                Road_Delay *rd;
                rd = new Road_Delay();
                rd->time = atol(values.at(1).c_str());
                rd->Link_ID = atoi(values.at(2).c_str());
                //TO_DO delay not found
                rd->Delay = atoi(values.at(3).c_str());
                v_rdy.push_back(rd);
        }
        else if(values.at(0) == "EDtoDP07")
        {
                //cout << "Type is " << values[0] << endl;
                Hospital_Delay *hd;
                hd = new Hospital_Delay();
                hd->time = atol(values.at(1).c_str());
                hd->HospitalID = atol(values.at(2).c_str());
                //hd->Delay_1 = atofvalues[3].c_str());
                hd->Delay_2 = atof(values.at(3).c_str());
                hd->Delay_3 = atof(values.at(4).c_str());
                v_hd.push_back(hd);
        }
        else if(values.at(0) == "EDtoDP08")
        {
                //cout << "Type is " << values.size() << endl;
                Ambulance_Idle *ai;
                ai = new Ambulance_Idle();
                ai->time = atol(values.at(1).c_str());
                ai->AmbulanceID = atol(values.at(2).c_str());
                ai->X = atof(values.at(3).c_str());
                ai->Y = atof(values.at(4).c_str() );
                ai->Nearest_Node = atol(values.at(5).c_str());
                ai->Onboard = atoi(values.at(6).c_str());
                v_ai.push_back(ai);
        }
        else if(values.at(0) == "EDtoDP09")
        {
                //cout << "Type is " << values[0] << endl;
                Ambulance_Stuck *as;
                as = new Ambulance_Stuck();
```

```
                        as->time = atol(values.at(1).c_str());
                        as->AmbulanceID = atol(values.at(2).c_str());
                        as->X = atof(values.at(3).c_str());
                        as->Y = atof(values.at(4).c_str());
                        as->Nearest_Node = atol(values.at(5).c_str());
                        as->Loaded = ((values.at(6).c_str()) == "T") ? (true) : (false);
                        v_as.push_back(as);
                }
                else if(values.at(0) == "EDtoDP10")
                {
                        //cout << "Type is " << values[0] << endl;
                        Cluster_Identify *ci;
                        ci = new Cluster_Identify();
                        ci->time = atol(values.at(4).c_str());
                        ci->ClusterID = atol(values.at(7).c_str());
                        ci->Size = atoi(values.at(8).c_str());
                        //TO-DO fill up for cluster cell.
                        /*int l = (values.size()/4)-1;
                        for (int i=0, index=4; i<values.size(); ++i, index+4) {
                        }*/
                        v_ci.push_back(ci);
                }
        }
}
vector <Casualty_Observation*> get_Casuality_Observation()
{
        return v_co;
}
vector <Casualty_Pickup*> get_Casuality_Pickup()
{
        return v_cp;
}
vector <Casualty_Delivery*> get_Casualty_Delivery()
{
        return v_cd;
}
vector <Road_Damage*> get_Road_Damage()
{
        return v_rd;
}
vector <Hospital_Capacity*> get_Hospital_Capacity()
{
        return v_hc;
}
vector <Road_Delay*> get_Road_Delay()
{
        return v_rdy;
}
vector <Hospital_Delay*> get_Hospital_Delay()
 {
        return v_hd;
}
vector <Ambulance_Idle*> get_Ambulance_Idle()
```

AFOSR F49620-01-1-0371

```
         {
                 return v_ai;
         }
         vector <Ambulance_Stuck*> get_Ambulance_Stuck()
         {
                 return v_as;
         }
         vector <Cluster_Identify*> get_Cluster_Identify()
         {
                 return v_ci;
         }
         private:
                 vector< Casualty_Observation* > v_co;
                 vector< Casualty_Pickup* > v_cp;
                 vector< Casualty_Delivery* > v_cd;
                 vector< Road_Damage* > v_rd;
                 vector< Hospital_Capacity* > v_hc;
                 vector< Road_Delay* > v_rdy;
                 vector< Hospital_Delay* > v_hd;
                 vector< Ambulance_Idle* > v_ai;
                 vector< Ambulance_Stuck* > v_as;
                 vector< Cluster_Identify* > v_ci;
};
struct Nodes                          // for finding the nearest node
{
       long          nodeid[MAX]    ;
       double        x[MAX]         ;
       double        y[MAX]         ;
}Nearest;
int main()
{
       int           Num_Cas=0;
       int           Num_Idle=0;
       int           Num_Hosp=0;
       int           i=0;
       int           j=0;
       double  Distance_Cas[50];
       int           X_Cas[20];
       int           Y_Cas[20];
       int           X_Amb[20];
       int           Y_Amb[20];
       int           Amb_Disp=0;
       int           ORIGIN=0;
       int           DESTINATION=0;
       double  z_x=0.0;// Difference in X coordinates for calculation of Nearest Node
       double  z_y=0.0;
       double  Min=99999.0;
       double  Min_Dist_Cas=99999.0;
       double  Nearest_dist[200];
       long    Nearest_Node[200];
       int           range=0;
       int           Cas_X;// For Dispatch to Hospital
       int           Cas_Y;
```

```cpp
        int         X_Hosp[20];
        int         Y_Hosp[20];
        double Min_Dist_Hosp=999999.0;
        double Distance_Hosp[50];
        int         Hosp_Disp=0;
        MainClass *mc = new MainClass();
        //testing
        ORIGIN = 96964094;
        DESTINATION = 1396514;
        mc->main_method(ORIGIN, DESTINATION);
        cout << "End of Calling the shortest path for the first time" << endl;
        int choice;
        cin >> choice;
        FILE *fp;
        DispTest dt = DispTest();
        dt.parse_File();
        Num_Cas = dt.get_Casuality_Observation().size();
//      cout<<Num_Cas<<endl;
        Num_Hosp= dt.get_Hospital_Capacity().size();
        Num_Idle = dt.get_Ambulance_Idle().size();

        vector <Casualty_Observation*> Cas_Observation=dt.get_Casuality_Observation() ;
        vector <Road_Damage*> Road_Damage=dt.get_Road_Damage();
        vector <Hospital_Capacity*> Hospital_Capacity=dt.get_Hospital_Capacity();
        vector <Road_Delay*> Road_Delay=dt.get_Road_Delay();
        vector <Hospital_Delay*> Hospital_Delay=dt.get_Hospital_Delay();
        vector <Ambulance_Idle*> Amb_Idle=dt.get_Ambulance_Idle();

//********************** Finding Nearest Node for Hospitals *************************//
        fp = fopen("Nearest_Node.txt", "r");
        for (j=0;j<MAX;j++)
        {
                Nearest.nodeid[j] = 0;
                Nearest.x[j] = 0;
                Nearest.y[j] = 0;
        }

        if (fp == NULL)
        {
                cout<<"Could not open file for reading"<<endl;
        }
        while (feof(fp) == 0)
        {
                fscanf(fp, "%ld\t%lf\t%lf\n", &Nearest.nodeid[i], &Nearest.x[i], &Nearest.y[i]);
                i++;
        }
        fclose(fp);
        cout<<"Enter the value of Range within which the Coordinates are to be found:    "<<endl;
        //cin>>range;
        range=300;
        for(int m=0;m<Num_Hosp;m++)
        {
                for(int n=0;n<i;n++)
```

```
                {
                         z_x= Hospital_Capacity.at(m)->X  -  Nearest.x[n];
                         z_y= Hospital_Capacity.at(m)->Y  -  Nearest.y[n];
                        if ((z_x>-range) && (z_x<range) && (z_y>-range) && (z_y<range))
                        {
                                Nearest_dist[m] = (pow(pow(z_x,2)+pow(z_y,2),0.5));
                                if(Min> Nearest_dist[m])
                                {
                                        Min=Nearest_dist[m];
                                        Nearest_Node[m]=Nearest.nodeid[n];
                                }
                        }
                }
                Min=99999.0;
        //      cout<<"Nearest Node for Hospital "<<m<<" is  "<< Nearest_Node[m]<<endl;
        }
//***************************************************//

//******** DISPATCHING FROM AMBULANCE LOCATION TO CASUALTY LOCATION*******//
//**** Dispatch to Severity type 3 First ****//
for(i=0;i<Num_Cas;i++)
        {
                if(Cas_Observation.at(i)->Severity==3)
                {
                        X_Cas[i]=Cas_Observation.at(i)->X; //** Get X,Y Co-ordinates
                        Y_Cas[i]=Cas_Observation.at(i)->Y;
                        for(j=0;j<Num_Idle;j++)
                        {
                                if(Amb_Idle.at(j)->Onboard==0)
                                {
                                        X_Amb[j]=Amb_Idle.at(j)->X; //** Get X,Y Co-ordinates
                                        Y_Amb[j]=Amb_Idle.at(j)->Y;
                                        Distance_Cas[j]   =   (pow(pow((Cas_Observation.at(i)->X)-(Amb_Idle.at(j)-
>X),2)+pow((Cas_Observation.at(i)->Y)-(Amb_Idle.at(j)->Y),2),0.5));
                                        cout<<"Distance_Cas "<<j <<" is  "<<Distance_Cas[j]<<endl;
                                        if(Min_Dist_Cas> Distance_Cas[j])
                                        {
                                                Min_Dist_Cas=Distance_Cas[j];
                                                Amb_Disp=Amb_Idle.at(j)->AmbulanceID;
                                        }
                                }
                        }

                        Min_Dist_Cas=999999.0;
                        cout<<"For Casualty "<<i+1<<" Dispatch Ambulance "<<Amb_Disp<<endl;
                        for(j=0;j<Num_Idle;j++)
                        {
                                if(Amb_Idle.at(j)->AmbulanceID==Amb_Disp)
                                {
                                        Amb_Idle.at(j)->Onboard=1;                      // Update  the  status  of  the
dispatched ambulance to Busy
                                        ORIGIN=Amb_Idle.at(j)->Nearest_Node;
                                        cout<<"Origin is      "<<ORIGIN<<endl;
```

```
                              }
                       }
               DESTINATION=Cas_Observation.at(i)->Nearest_Node;
               cout<<"Destination is  "<<DESTINATION<<endl<<endl;
      //*********** insert the router call function***//
               //shortest_path(ORIGIN, DESTINATION);
               Cas_X=Cas_Observation.at(i)->X;
               Cas_Y=Cas_Observation.at(i)->Y;
               //********Dispatch to Hospital*****/////////
               Min_Dist_Hosp=999999.0;
               for(int k=0;k<Num_Hosp;k++)
                       {
                               if(Hospital_Capacity.at(k)->Severity_3 > 5)
                               {
                                       X_Hosp[k]=Hospital_Capacity.at(k)->X; //** Get X,Y Co-ordinates
                                       Y_Hosp[k]=Hospital_Capacity.at(k)->Y;
                                       Distance_Hosp[k]          =          (pow(pow((Hospital_Capacity.at(k)->X)-
(Cas_X),2)+pow((Hospital_Capacity.at(k)->Y)-(Cas_Y),2),0.5));
                                       cout<<"Distance to hospital   "<<k  <<"  is  "<<Distance_Hosp[k]<<endl;
                                       if(Min_Dist_Hosp> Distance_Hosp[k])
                                               {
                                                       Min_Dist_Hosp=Distance_Hosp[k];
                                                       Hosp_Disp=Hospital_Capacity.at(k)->HospitalID;
                                                       //     cout<<"Min Distance is  "<<Min_Dist_Hosp<<endl;
                                                       //     cout<<"For  Casualty  "<<i+1<<"  Dispatch  Ambulance
"<<Amb_Disp<<endl;
                                               }

                               }// if loop
                                       //cout<<"Casualty  "<<i<<"   Dispatch to hospital  "<<Hosp_Disp<<endl;


                       } //for loop

                       for(int l=0;l<Num_Hosp;l++)
                       {
                               if(Hospital_Capacity.at(l)->HospitalID==Hosp_Disp)
                               {
                                       Hospital_Capacity.at(l)->Severity_3=Hospital_Capacity.at(l)->Severity_3   -
5;// Update the Capacity of the Hospital
                                       cout<<"Capacity  of  hospital  remaining     "<<  Hospital_Capacity.at(l)-
>Severity_3<<endl;
                                       DESTINATION=Hospital_Capacity.at(l)->Nearest_Node;
                               }
                       }

               ORIGIN = Cas_Observation.at(i)->Nearest_Node;
               cout<<"Casualty  "<<i<<"   Dispatch to hospital  "<<Hosp_Disp<<endl;
               cout<<"Destination hospital is        "<<DESTINATION<<endl;
               cout<<"Origin is Casualty Location   "<<ORIGIN<<endl<<endl;

//             Route_origin_destin(ORIGIN,DESTINATION);
                       mc->main_method(ORIGIN, DESTINATION);
```

-A-93-

```
                    } //End 1st IF LOOP
            }// 1st FOR loop
//************************************************//
//****Dispatch to Severity type 2 ****//
for(i=0;i<Num_Cas;i++)
            {
                    if(Cas_Observation.at(i)->Severity==2)
                    {
                            X_Cas[i]=Cas_Observation.at(i)->X; //** Get X,Y Co-ordinates
                            Y_Cas[i]=Cas_Observation.at(i)->Y;
                            for(j=0;j<Num_Idle;j++)
                            {
//                                  cout<<"##"<<Amb_Idle.at(j)->Onboard<<endl;
                                    if(Amb_Idle.at(j)->Onboard==0)
                                    {
                                            X_Amb[j]=Amb_Idle.at(j)->X; //** Get X,Y Co-ordinates
                                            Y_Amb[j]=Amb_Idle.at(j)->Y;
                                            Distance_Cas[j]   =   (pow(pow((Cas_Observation.at(i)->X)-(Amb_Idle.at(j)-
>X),2)+pow((Cas_Observation.at(i)->Y)-(Amb_Idle.at(j)->Y),2),0.5));
                                            cout<<"Distance_Cas "<<j <<" is  "<<Distance_Cas[j]<<endl;
                                            if(Min_Dist_Cas> Distance_Cas[j])
                                            {
                                                    Min_Dist_Cas=Distance_Cas[j];
                                                    Amb_Disp=Amb_Idle.at(j)->AmbulanceID;
//                                                  cout<<"Min Distance is  "<<Min_Dist_Cas<<endl;
//                                                  cout<<"For Casualty "<<i+1<<" Dispatch Ambulance "<<Amb_Disp<<endl;
                                            }
                                    }
                            }

                            Min_Dist_Cas=999999.0;
                            cout<<"########For Casualty "<<i+1<<" Dispatch Ambulance "<<Amb_Disp<<endl;
                            for(j=0;j<Num_Idle;j++)
                            {
                                    if(Amb_Idle.at(j)->AmbulanceID==Amb_Disp)
                                    {
//                                          cout<<"Dispatched Ambulance is        "<<Amb_Idle.at(j)->AmbulanceID<<endl;
                                            Amb_Idle.at(j)->Onboard=1;
                                            ORIGIN=Amb_Idle.at(j)->Nearest_Node;
                                            cout<<"Origin is       "<<ORIGIN<<endl;
                                    }
                            }
                    DESTINATION=Cas_Observation.at(i)->Nearest_Node;
                    cout<<"Destination is  "<<DESTINATION<<endl<<endl;
//*********** insert the router call function***//
                    //shortest_path(ORIGIN, DESTINATION);
                    mc->main_method(ORIGIN, DESTINATION);
                    Cas_X=Cas_Observation.at(i)->X;
                    Cas_Y=Cas_Observation.at(i)->Y;
                                    //********Dispatch to Hospital*****/////////
                    Min_Dist_Hosp=999999.0;
                    for(int k=0;k<10;k++)
```

```cpp
                        {
                                if(Hospital_Capacity.at(k)->Severity_2 > 5)
                                {
                                        X_Hosp[k]=Hospital_Capacity.at(k)->X; //** Get X,Y Co-ordinates
                                        Y_Hosp[k]=Hospital_Capacity.at(k)->Y;
                                        Distance_Hosp[k]          =          (pow(pow((Hospital_Capacity.at(k)->X)-
(Cas_X),2)+pow((Hospital_Capacity.at(k)->Y)-(Cas_Y),2),0.5));
                                        cout<<"Distance to hospital   "<<k  <<"  is  "<<Distance_Hosp[k]<<endl;
                                        if(Min_Dist_Hosp> Distance_Hosp[k])
                                             {
                                                     Min_Dist_Hosp=Distance_Hosp[k];
                                                     Hosp_Disp=Hospital_Capacity.at(k)->HospitalID;
                                                     //     cout<<"Min Distance is  "<<Min_Dist_Hosp<<endl;
                                                     //     cout<<"For  Casualty  "<<i+1<<"  Dispatch  Ambulance
"<<Amb_Disp<<endl;
                                             }

                                }// if loop
                                        //cout<<"Casualty  "<<i<<"   Dispatch to hospital  "<<Hosp_Disp<<endl;

                        } //for loop
                        for(int l=0;l<Num_Hosp;l++)
                        {
                                if(Hospital_Capacity.at(l)->HospitalID==Hosp_Disp)
                                {
                                        Hospital_Capacity.at(l)->Severity_2=Hospital_Capacity.at(l)->Severity_2   -
5;// Update the Capacity of the Hospital
                                        cout<<"Capacity  of  hospital  remaining    "<<  Hospital_Capacity.at(l)-
>Severity_2<<endl;

                                        DESTINATION=Hospital_Capacity.at(l)->Nearest_Node;
                                }
                        }

                                ORIGIN = Cas_Observation.at(i)->Nearest_Node;
                                cout<<"Casualty  "<<i<<"   Dispatch to hospital  "<<Hosp_Disp<<endl;
                                cout<<"Destination hospital is        "<<DESTINATION<<endl;
                                cout<<"Origin is Casualty Location   "<<ORIGIN<<endl<<endl;


//*******************************////////
//              Route_origin_destin(ORIGIN,DESTINATION);
                                mc->main_method(ORIGIN, DESTINATION);
                } //End 2nd IF LOOP
        }// 2nd FOR loop
        return 0;
}

#include "Integration.h"
void INTEGRATION::ResetKpath()
{
        NoA=NoB=KPATH=0;
        DeletePathList(A);
        DeletePathNode(Head);
```

```
        Head=Last=NULL;
        A=NULL;
        DeleteFinalPath(Final);
        Final=NULL;
}
void INTEGRATION::SetPair(int so, int si, int kp)
{
        Origin=so;
        Destin=si;
        KPATH=kp;
}


void INTEGRATION::AddPathA(FinalPath* cand)
{
        PathList *Cur=A;
        PathList *NewList=new PathList;
        NewList->nodcnt=0;
        NewList->length=0;
        FinalPath *reserve=cand;
        while (cand!=NULL)
        {
                NewList->nodcnt++;
                NewList->length=cand->label;
                cand=cand->next;
        }
        NewList->nodix=new int [NewList->nodcnt];
        NewList->lnkix=new int [NewList->nodcnt];
        NewList->label=new double [NewList->nodcnt];
        NewList->dist=new double [NewList->nodcnt];
        NewList->time=new double [NewList->nodcnt];
        cand=reserve;
        int i=0;
        while (cand!=NULL)
        {
                NewList->nodix[i]=cand->nodix;
                NewList->lnkix[i]=cand->lnkix;
                NewList->label[i]=cand->label;
                NewList->dist[i]=cand->dist;
                NewList->time[i]=cand->time;
                cand=cand->next;
                i++;
        }
        NewList->flag=0;
        NewList->next=A;
        A=NewList;
        NoA++;
}
/////////////////////////////
// The debugger can't handle symbols more than 255 characters long.
// STL often creates symbols longer than that.
// When symbols are longer than 255 characters, the warning is issued.
#pragma warning(disable:4786)
/////////////////////////////
```

```cpp
//-----------------------------------------------------------------
// Project Include Files
//-----------------------------------------------------------------
#include "HwFederateAmbassador.hh"
#include "localFederate.h"
//-----------------------------------------------------------------
// System Include Files
//-----------------------------------------------------------------
#ifndef _MSC_VER
#include <stdio.h>
#include <stdlib.h>
#include <iostream.h>
#else
#include <iostream>
using std::cout;
using std::cerr;
using std::endl;
#endif
//-----------------------------------------------------------------
// Bad C like global variables being externed - bad boy!!!
//-----------------------------------------------------------------
extern RTI::Boolean        timeAdvGrant;
extern RTI::Boolean        TimeRegulation;
extern RTI::Boolean        TimeConstrained;
extern RTI::FedTime  &       grantTime;
extern fedModel *localFederate;
HwFederateAmbassador::HwFederateAmbassador()
{
}
HwFederateAmbassador::~HwFederateAmbassador()
throw(RTI::FederateInternalError)
{
    cerr << "FED_HW: HwFederateAmbassador::~HwFederateAmbassador destructor called in FED" << endl;
}
/////////////////////////////////////
// Federation Management Services //
/////////////////////////////////////
void HwFederateAmbassador::synchronizationPointRegistrationSucceeded (
  const char *label) // supplied C4)
throw (
  RTI::FederateInternalError)
{
    cerr << "FED_HW: synchronizationPointRegistrationSucceeded not supported in FED"
        << endl;
}
void HwFederateAmbassador::synchronizationPointRegistrationFailed (
  const char *label) // supplied C4)
throw (
  RTI::FederateInternalError)
{
    cerr << "FED_HW: synchronizationPointRegistrationFailed not supported in FED"
        << endl;
}
```

AFOSR F49620-01-1-0371

```cpp
void HwFederateAmbassador::announceSynchronizationPoint (
  const char *label, // supplied C4
  const char *tag)   // supplied C4
throw (
  RTI::FederateInternalError)
{
    cerr << "FED_HW: announceSynchronizationPoint not supported in FED" << endl;
}
void HwFederateAmbassador::federationSynchronized (
  const char *label) // supplied C4)
throw (
  RTI::FederateInternalError)
{
    cerr << "FED_HW: federationSynchronized not supported in FED" << endl;
}


void HwFederateAmbassador::initiateFederateSave (
  const char *label) // supplied C4
throw (
  RTI::UnableToPerformSave,
  RTI::FederateInternalError)
{
    cerr << "FED_HW: initiateFederateSave not supported in FED" << endl;
}


void HwFederateAmbassador::federationSaved ()
throw (
  RTI::FederateInternalError)
{
    cerr << "FED_HW: federationSaved not supported in FED" << endl;
}


void HwFederateAmbassador::federationNotSaved ()
throw (
  RTI::FederateInternalError)
{
    cerr << "FED_HW: federationNotSaved not supported in FED" << endl;
}


void HwFederateAmbassador::requestFederationRestoreSucceeded (
  const char *label) // supplied C4
throw (
  RTI::FederateInternalError)
{
    cerr << "FED_HW: requestFederationRestoreSucceeded not supported in FED" << endl;
}


void HwFederateAmbassador::requestFederationRestoreFailed (
  const char *label) // supplied C4
throw (
  RTI::FederateInternalError)
{
    cerr << "FED_HW: requestFederationRestoreFailed not supported in FED" << endl;
```

```cpp
}

void HwFederateAmbassador::requestFederationRestoreFailed (
  const char *label,
  const char *reason) // supplied C4
throw (
  RTI::FederateInternalError)
{
    cerr << "FED_HW: requestFederationRestoreFailed not supported in FED" << endl;
}


void HwFederateAmbassador::federationRestoreBegun ()
throw (
  RTI::FederateInternalError)
{
    cerr << "FED_HW: federationRestoreBegun not supported in FED" << endl;
}


void HwFederateAmbassador::initiateFederateRestore (
  const char              *label,   // supplied C4
        RTI::FederateHandle handle)  // supplied C1
throw (
  RTI::SpecifiedSaveLabelDoesNotExist,
  RTI::CouldNotRestore,
  RTI::FederateInternalError)
{
    cerr << "FED_HW: initiateFederateRestore not supported in FED" << endl;
}


void HwFederateAmbassador::federationRestored ()
throw (
  RTI::FederateInternalError)
{
    cerr << "FED_HW: federationRestored not supported in FED" << endl;
}


void HwFederateAmbassador::federationNotRestored ()
throw (
  RTI::FederateInternalError)
{
    cerr << "FED_HW: federationNotRestored not supported in FED" << endl;
}

///////////////////////////////////
// Declaration Management Services //
///////////////////////////////////
void HwFederateAmbassador::startRegistrationForObjectClass (
        RTI::ObjectClassHandle   theClass)      // supplied C1
throw (
  RTI::ObjectClassNotPublished,
  RTI::FederateInternalError)
{
    cerr << "FED_HW: startRegistrationForObjectClass not supported in FED" << endl;
```

```
}

void HwFederateAmbassador::stopRegistrationForObjectClass (
       RTI::ObjectClassHandle   theClass)       // supplied C1
throw (
  RTI::ObjectClassNotPublished,
  RTI::FederateInternalError)
{
   cerr << "FED_HW: stopRegistrationForObjectClass not supported in FED" << endl;
}


void HwFederateAmbassador::turnInteractionsOn (
  RTI::InteractionClassHandle theHandle) // supplied C1
throw (
  RTI::InteractionClassNotPublished,
  RTI::FederateInternalError)
{
   cerr << "FED_HW: turnInteractionsOn not supported in FED" << endl;
}
void HwFederateAmbassador::turnInteractionsOff (
  RTI::InteractionClassHandle theHandle) // supplied C1
throw (
  RTI::InteractionClassNotPublished,
  RTI::FederateInternalError)
{
   cerr << "FED_HW: turnInteractionsOff not supported in FED" << endl;
}
/////////////////////////////////
// Object Management Services //
/////////////////////////////////
void HwFederateAmbassador::discoverObjectInstance (
  RTI::ObjectHandle          theObject,      // supplied C1
  RTI::ObjectClassHandle     theObjectClass, // supplied C1
  const char *          theObjectName)  // supplied C4
throw (
  RTI::CouldNotDiscover,
  RTI::ObjectClassNotKnown,
  RTI::FederateInternalError)
{
//   cout << "FED_HW: Discovered object " << theObject << endl;
//   cout << "FED_HW:    Object name = " << theObjectName << endl;
//   localFederate->receivePublisher(theObject, theObjectClass, theObjectName);
}
void HwFederateAmbassador::reflectAttributeValues (
       RTI::ObjectHandle                 theObject,     // supplied C1
  const RTI::AttributeHandleValuePairSet& theAttributes, // supplied C4
  const RTI::FedTime&                    theTime,       // supplied C1
  const char                            *theTag,        // supplied C4
       RTI::EventRetractionHandle        theHandle)     // supplied C1
throw (
  RTI::ObjectNotKnown,
  RTI::AttributeNotKnown,
  RTI::FederateOwnsAttributes,
```

```cpp
  RTI::InvalidFederationTime,
  RTI::FederateInternalError)
{
        return;                // Not needed for this federation execution
}
void HwFederateAmbassador::reflectAttributeValues (
        RTI::ObjectHandle                  theObject,     // supplied C1
  const RTI::AttributeHandleValuePairSet& theAttributes, // supplied C4
  const char                             *theTag)        // supplied C4
throw (
  RTI::ObjectNotKnown,
  RTI::AttributeNotKnown,
  RTI::FederateOwnsAttributes,
  RTI::FederateInternalError)
{
        return;                // Not needed for this federation execution
}
void HwFederateAmbassador::receiveInteraction (
        RTI::InteractionClassHandle       theInteraction, // supplied C1
  const RTI::ParameterHandleValuePairSet& theParameters,  // supplied C4
  const RTI::FedTime&                     theTime,        // supplied C4
  const char                             *theTag,         // supplied C4
        RTI::EventRetractionHandle        theHandle)      // supplied C1
throw (
  RTI::InteractionClassNotKnown,
  RTI::InteractionParameterNotKnown,
  RTI::InvalidFederationTime,
  RTI::FederateInternalError)
{
   localFederate->receiveInteraction( theInteraction, theParameters, theTime, theTag, theHandle );
}
void HwFederateAmbassador::receiveInteraction (
        RTI::InteractionClassHandle       theInteraction, // supplied C1
  const RTI::ParameterHandleValuePairSet& theParameters,  // supplied C4
  const char                             *theTag)         // supplied C4
throw (
  RTI::InteractionClassNotKnown,
  RTI::InteractionParameterNotKnown,
  RTI::FederateInternalError)
{
//  Pass the interaction off to the local federate object
// so that it can be processed.
   // localFederate->receiveInteraction( theInteraction, theParameters, theTag );
// This function won't be used in the disaster simulation.
}
void HwFederateAmbassador::removeObjectInstance (
        RTI::ObjectHandle          theObject, // supplied C1
  const RTI::FedTime&              theTime,   // supplied C4
  const char                      *theTag,    // supplied C4
        RTI::EventRetractionHandle theHandle) // supplied C1
throw (
  RTI::ObjectNotKnown,
  RTI::InvalidFederationTime,
```

```cpp
  RTI::FederateInternalError)
{
   return;            // Not needed for IFD
}
void HwFederateAmbassador::removeObjectInstance (
        RTI::ObjectHandle         theObject, // supplied C1
  const char                      *theTag)   // supplied C4
throw (
  RTI::ObjectNotKnown,
  RTI::FederateInternalError)
{
   return;            // Not needed for IFD
}
void HwFederateAmbassador::attributesInScope (
        RTI::ObjectHandle         theObject,    // supplied C1
  const RTI::AttributeHandleSet& theAttributes) // supplied C4
throw (
  RTI::ObjectNotKnown,
  RTI::AttributeNotKnown,
  RTI::FederateInternalError)
{
   cerr << "FED_HW: attributesInScope not supported in FED" << endl;
}
void HwFederateAmbassador::attributesOutOfScope (
        RTI::ObjectHandle         theObject,    // supplied C1
  const RTI::AttributeHandleSet& theAttributes) // supplied C4
throw (
  RTI::ObjectNotKnown,
  RTI::AttributeNotKnown,
  RTI::FederateInternalError)
{
   cerr << "FED_HW: attributesOutOfScope not supported in FED" << endl;
}
void HwFederateAmbassador::provideAttributeValueUpdate (
        RTI::ObjectHandle         theObject,    // supplied C1
  const RTI::AttributeHandleSet& theAttributes) // supplied C4
throw (
  RTI::ObjectNotKnown,
  RTI::AttributeNotKnown,
  RTI::AttributeNotOwned,
  RTI::FederateInternalError)
{
        return;        // Not needed for the IFD federate
}
void HwFederateAmbassador::turnUpdatesOnForObjectInstance (
        RTI::ObjectHandle         theObject,    // supplied C1
  const RTI::AttributeHandleSet& theAttributes) // supplied C4
throw (
  RTI::ObjectNotKnown,
  RTI::AttributeNotOwned,
  RTI::FederateInternalError)
{
   cerr << "FED_HW: turnUpdatesOnForObjectInstance not supported in FED" << endl;
```

```
}
void HwFederateAmbassador::turnUpdatesOffForObjectInstance (
        RTI::ObjectHandle        theObject,        // supplied C1
  const RTI::AttributeHandleSet& theAttributes) // supplied C4
throw (
  RTI::ObjectNotKnown,
  RTI::AttributeNotOwned,
  RTI::FederateInternalError)
{
    cerr << "FED_HW: turnUpdatesOffForObjectInstance not supported in FED" << endl;
}
/////////////////////////////////
// Ownership Management Services //
/////////////////////////////////
void HwFederateAmbassador::requestAttributeOwnershipAssumption (
        RTI::ObjectHandle        theObject,           // supplied C1
  const RTI::AttributeHandleSet& offeredAttributes, // supplied C4
  const char                    *theTag)             // supplied C4
throw (
  RTI::ObjectNotKnown,
  RTI::AttributeNotKnown,
  RTI::AttributeAlreadyOwned,
  RTI::AttributeNotPublished,
  RTI::FederateInternalError)
{
    cerr << "FED_HW: requestAttributeOwnershipAssumption not supported in FED" << endl;
}
void HwFederateAmbassador::attributeOwnershipDivestitureNotification (
        RTI::ObjectHandle        theObject,           // supplied C1
  const RTI::AttributeHandleSet& releasedAttributes) // supplied C4
throw (
  RTI::ObjectNotKnown,
  RTI::AttributeNotKnown,
  RTI::AttributeNotOwned,
  RTI::AttributeDivestitureWasNotRequested,
  RTI::FederateInternalError)
{
    cerr << "FED_HW: attributeOwnershipDivestitureNotification not supported in FED"
        << endl;
}
void HwFederateAmbassador::attributeOwnershipAcquisitionNotification (
        RTI::ObjectHandle        theObject,        // supplied C1
  const RTI::AttributeHandleSet& securedAttributes) // supplied C4
throw (
  RTI::ObjectNotKnown,
  RTI::AttributeNotKnown,
  RTI::AttributeAcquisitionWasNotRequested,
  RTI::AttributeAlreadyOwned,
  RTI::AttributeNotPublished,
  RTI::FederateInternalError)
{
    cerr << "FED_HW: attributeOwnershipAcquisitionNotification not supported in FED"
        << endl;
```

```cpp
}
void HwFederateAmbassador::attributeOwnershipUnavailable (
        RTI::ObjectHandle        theObject,           // supplied C1
  const RTI::AttributeHandleSet& theAttributes) // supplied C4
throw (
  RTI::ObjectNotKnown,
  RTI::AttributeNotKnown,
  RTI::AttributeAlreadyOwned,
  RTI::AttributeAcquisitionWasNotRequested,
  RTI::FederateInternalError)
{
   cerr << "FED_HW: attributeOwnershipUnavailable not supported in FED" << endl;
}
void HwFederateAmbassador::requestAttributeOwnershipRelease (
        RTI::ObjectHandle        theObject,           // supplied C1
  const RTI::AttributeHandleSet& candidateAttributes, // supplied C4
  const char                     *theTag)             // supplied C4
throw (
  RTI::ObjectNotKnown,
  RTI::AttributeNotKnown,
  RTI::AttributeNotOwned,
  RTI::FederateInternalError)
{
   cerr << "FED_HW: requestAttributeOwnershipRelease not supported in FED" << endl;
}
void HwFederateAmbassador::confirmAttributeOwnershipAcquisitionCancellation (
        RTI::ObjectHandle        theObject,           // supplied C1
  const RTI::AttributeHandleSet& theAttributes) // supplied C4
throw (
  RTI::ObjectNotKnown,
  RTI::AttributeNotKnown,
  RTI::AttributeAlreadyOwned,
  RTI::AttributeAcquisitionWasNotCanceled,
  RTI::FederateInternalError)
{
   cerr << "FED_HW: confirmAttributeOwnershipAcquisitionCancellation not"
        << " supported in FED" << endl;
}
void HwFederateAmbassador::informAttributeOwnership (
  RTI::ObjectHandle    theObject,    // supplied C1
  RTI::AttributeHandle theAttribute, // supplied C1
  RTI::FederateHandle  theOwner)     // supplied C1
throw (
  RTI::ObjectNotKnown,
  RTI::AttributeNotKnown,
  RTI::FederateInternalError)
{
   cerr << "FED_HW: informAttributeOwnership not supported in FED" << endl;
}
 void HwFederateAmbassador::attributeIsNotOwned (
  RTI::ObjectHandle    theObject,    // supplied C1
  RTI::AttributeHandle theAttribute) // supplied C1
throw (
```

```
  RTI::ObjectNotKnown,
  RTI::AttributeNotKnown,
  RTI::FederateInternalError)
{
   cerr << "FED_HW: attributeIsNotOwned not supported in FED" << endl;
}
void HwFederateAmbassador::attributeOwnedByRTI (
  RTI::ObjectHandle    theObject,    // supplied C1
  RTI::AttributeHandle theAttribute) // supplied C1
throw (
  RTI::ObjectNotKnown,
  RTI::AttributeNotKnown,
  RTI::FederateInternalError)
{
   cerr << "FED_HW: attributeOwnedByRTI not supported in FED" << endl;
}
/////////////////////////////
// Time Management Services //
/////////////////////////////
void HwFederateAmbassador::timeRegulationEnabled (
 const  RTI::FedTime& theFederateTime) // supplied C4
throw (
  RTI::InvalidFederationTime,
  RTI::EnableTimeRegulationWasNotPending,
  RTI::FederateInternalError)
{
 //  cout << "FED_HW: Time granted (timeRegulationEnabled) to: "
  //      << theFederateTime << endl;
   grantTime = theFederateTime;
   timeAdvGrant = RTI::RTI_TRUE;
   TimeRegulation = RTI::RTI_TRUE;
}
void HwFederateAmbassador::timeConstrainedEnabled (
  const RTI::FedTime& theFederateTime) // supplied C4
throw (
  RTI::InvalidFederationTime,
  RTI::EnableTimeConstrainedWasNotPending,
  RTI::FederateInternalError)
{
 //  cout << "FED_HW: Time granted (timeConstrainedEnabled) to: "
  //      << theFederateTime << endl;
   grantTime = theFederateTime;
   timeAdvGrant = RTI::RTI_TRUE;
   TimeConstrained = RTI::RTI_TRUE;
}
void HwFederateAmbassador::timeAdvanceGrant (
  const RTI::FedTime& theTime) // supplied C4
throw (
  RTI::InvalidFederationTime,
  RTI::TimeAdvanceWasNotInProgress,
  RTI::FederateInternalError)
{
 //  cout << "FED_HW: Time granted (timeAdvanceGrant) to: "
```

-A-105-

```
  //          << theTime << endl;
   grantTime = theTime;
   timeAdvGrant = RTI::RTI_TRUE;
}
void HwFederateAmbassador::requestRetraction (
  RTI::EventRetractionHandle theHandle) // supplied C1
throw (
  RTI::EventNotKnown,
  RTI::FederateInternalError)
{
   cerr << "FED_HW: requestRetraction not supported in FED" << endl;
}
```

Appendix B: Sample Data Sets

To illustrate the dynamics of the interactions, the first 16 reports of each report type which were captured during a typical DIRE run are listed below. The total number of reports of each type captured on this run (Test 01-001-01) are indicated on the report title lines. For those report types for which 16 or less reports were captured, all reports are listed. All fields of all listed reports are entered, with the exception of certain report types which have a large and variable number of parameter fields. For these reports, the first few parameter fields are listed.

Details of the report types and their data fields are given in Sec. 6.2 of this report.

Casualty Observation Reports (16 /11,138)

Rept Type	SrcDest	Rptr Type	Juris ID	Time	Rept Cnt	TrackID	Loc_X	Loc_Y	CTractID	LErCov_X	LErCov_Y	Near Node
13	DFtoED01	31	1	1	1	100	923485	3798971	6.04E+09	400	400	75160557
13	DFtoED01	31	1	0	2	200	919779	3802921	6.04E+09	400	400	1453072
13	DFtoED01	31	2	0	3	300	906486	3790344	6.04E+09	400	400	1488661
13	DFtoED01	31	1	6	4	400	917988	3798270	6.04E+09	400	400	1724897
13	DFtoED01	31	1	7	5	500	917383	3795675	6.04E+09	400	400	1899571
13	DFtoED01	31	1	2	6	600	921560	3802299	6.04E+09	400	400	1638512
13	DFtoED01	31	1	4	7	700	918642	3797872	6.04E+09	400	400	2093495
13	DFtoED01	31	2	4	8	800	908758	3796287	6.04E+09	400	400	1902993
13	DFtoED01	31	1	5	9	900	919417	3797511	6.04E+09	400	400	2190567
13	DFtoED01	31	1	140	10	1000	914692	3798640	6.04E+09	400	400	2029349
13	DFtoED01	31	1	7	11	1100	916516	3791644	6.04E+09	400	400	75151110
13	DFtoED01	31	1	9	12	1200	926126	3798654	6.04E+09	400	400	2008600
13	DFtoED01	31	2	4	13	1300	906698	3798703	6.04E+09	400	400	1834907
13	DFtoED01	31	1	5	14	1400	914910	3797307	6.04E+09	400	400	1481003
13	DFtoED01	31	2	7	15	1500	901465	3792210	6.04E+09	400	400	1903707
13	DFtoED01	31	1	104	16	1600	922948	3793826	6.04E+09	400	400	1723457

Cas_ID	Cas_Age	Cas_Race	Cas_Sex	Severity	Sev Pr 1	Sev Pr 2	Sev Pr 3	Sev Pr 4	RCnt_Pol	RCnt_Amb	RCnt_Civ	CumAsPr
1E+08	3	4	2	4	0	0.175258	5.15E-02	0.773196	0	0	165	1
1E+08	2	2	2	1	0.75025	0.16983	3.00E-02	5.00E-02	0	0	165	1
1E+08	4	1	1	1	0.956688	4.33E-02	0	0	0	0	165	1
9E+08	3	2	2	1	0.956688	4.33E-02	0	0	0	0	165	1
9E+08	4	1	2	1	0.956688	4.33E-02	0	0	0	0	165	1
9E+08	2	1	1	2	0.334692	0.649392	9.55E-03	6.37E-03	0	0	165	1
9E+08	1	2	1	1	0.956688	4.33E-02	0	0	0	0	165	1
9E+08	3	1	1	1	0.956688	4.33E-02	0	0	0	0	165	1
9E+08	3	2	2	1	0.75025	0.16983	3.00E-02	5.00E-02	0	0	165	1
9E+08	4	2	2	1	0.956688	4.33E-02	0	0	0	0	165	1
9E+08	4	2	2	1	0.956688	4.33E-02	0	0	0	0	165	1
9E+08	4	1	1	2	0.334692	0.649392	9.55E-03	6.37E-03	0	0	165	1
1E+08	1	1	2	1	0.296721	0.156723	0.296326	0.25023	0	0	165	1
9E+08	4	2	2	2	0.334692	0.649392	9.55E-03	6.37E-03	0	0	165	1
9E+08	1	1	2	2	0.334692	0.649392	9.55E-03	6.37E-03	0	0	165	1
9E+08	3	2	2	1	0.956688	4.33E-02	0	0	0	0	165	1

Roadway Damage Reports

Rept Type	SrcDest	Rptr Type	Juris ID	Time	Rept Cnt	TrackID	Bridge_ID	Link_ID	Sev
15	DFtoED03	28	1	430	1	104	80458	1336125	3
15	DFtoED03	28	1	440	2	204	80089	75153034	3
15	DFtoED03	28	1	450	3	304	80463	2192079	5
15	DFtoED03	28	1	470	4	404	80349	1467887	4
15	DFtoED03	28	1	732	5	204	80089	75153034	3
15	DFtoED03	28	1	726	6	304	80463	2192079	5
15	DFtoED03	28	1	720	7	104	80458	1336125	3
15	DFtoED03	28	1	728	8	404	80349	1467887	4
15	DFtoED03	28	1	680	9	504	80101	1814195	4
15	DFtoED03	28	1	880	10	604	0	1722831	4
15	DFtoED03	28	1	890	11	704	80103	0	4
15	DFtoED03	28	1	910	12	504	80101	1814195	4
15	DFtoED03	28	1	1210	13	704	80103	1722831	4

Casualty Pickup Reports (16/32)

Rept Type	SrcDest	Rptr Type	Juris ID	Time	Rept Cnt	TrackID	Nrst Node	Cas_ID	Cas_Age	Cas_Race	Cas_Sex	Cas_Sev
16	DFtoED04	27	1	472	1	200200	1397380	900207616	1	1	1	2
16	DFtoED04	27	1	495	2	213700	1481226	900215922	3	2	2	2
16	DFtoED04	27	1	688	3	233100	1434162	900201559	1	2	2	2
16	DFtoED04	27	1	640	4	282000	26360966	100214010	4	2	1	2
16	DFtoED04	27	1	755	5	338200	1898653	900201602	5	1	2	2
16	DFtoED04	27	1	827	6	382100	2194802	900211733	1	2	2	2
16	DFtoED04	27	1	985	7	400900	2157695	900213460	4	1	2	3
16	DFtoED04	27	1	889	8	417600	2017239	900209902	3	2	2	2
16	DFtoED04	27	1	889	9	417700	2017239	900215808	3	2	1	3
16	DFtoED04	27	2	892	10	424300	26316570	900214834	3	4	1	2
16	DFtoED04	27	1	946	11	456200	1385729	900201608	2	2	1	3
16	DFtoED04	27	1	1030	12	495200	1638519	900211506	4	1	1	2
16	DFtoED04	27	1	1030	13	495300	1638519	900211526	2	1	1	1
16	DFtoED04	27	1	1030	14	495400	1638519	100210022	4	2	1	2
16	DFtoED04	27	1	1036	15	495500	1898881	900208825	3	1	2	3
16	DFtoED04	27	1	1030	16	495600	1649189	900216287	4	4	2	1

Casualty Arrival Reports (16/5715)

Rept Type	SrcDest	SrcDest	Rptr Type	NA	Juris ID	Time	Rept Cnt	TrackID	Loc_X	Loc_Y	Hosp_ID	Cas_ID	Cas_Age	Cas_Race	Cas_Sex	Cas_Sev
17	DFtoED05	DFtoED05	25	0	2	77	1	26000			0	100208501	4	1	1	1
17	DFtoED05	DFtoED05	25	0	1	214	2	39500			0	100208430	3	2	2	2
17	DFtoED05	DFtoED05	25	0	1	123	3	43000			0	100208191	4	2	2	2
17	DFtoED05	DFtoED05	25	0	1	123	4	43100			0	100208191	4	2	1	1
17	DFtoED05	DFtoED05	25	0	1	257	5	43200			0	100208191	4	2	2	2
17	DFtoED05	DFtoED05	25	0	1	140	6	47100			0	100200021	3	2	1	1
17	DFtoED05	DFtoED05	25	0	2	275	7	50400			0	100211631	4	1	1	1
17	DFtoED05	DFtoED05	25	0	2	267	8	50500			0	100211631	4	2	1	1
17	DFtoED05	DFtoED05	25	0	2	148	9	50600			0	100211631	3	4	1	1
17	DFtoED05	DFtoED05	25	0	2	293	10	50700			0	100211631	4	1	1	1
17	DFtoED05	DFtoED05	25	0	1	281	11	57300			0	100200747	4	1	1	1
17	DFtoED05	DFtoED05	25	0	1	295	12	57400			0	100203172	3	4	2	2
17	DFtoED05	DFtoED05	25	0	1	305	13	63900			0	100214932	4	2	1	1
17	DFtoED05	DFtoED05	25	0	1	333	14	73700			0	100216361	3	1	2	2
17	DFtoED05	DFtoED05	25	0	1	220	15	73800			0	100216361	3	1	1	1
17	DFtoED05	DFtoED05	25	0	1	220	16	73900			0	100216361	3	1	1	1

Medical Facility Capacity Reports (16/3460)

Rept Type	SrcDest	SrcDest	Rptr Type	NA	Juris ID	Time	Rept Cnt	TrackID	Sev_1	Sev_2	Sev_3
20	DFtoED08	DFtoED08	61	60001	0	1804	1	808	1	0	0
20	DFtoED08	DFtoED08	61	60002	0	1804	2	808	20	2	2
20	DFtoED08	DFtoED08	61	60003	0	1804	3	808	13	1	1
20	DFtoED08	DFtoED08	61	60004	0	1804	4	808	7	0	0
20	DFtoED08	DFtoED08	61	60005	0	1804	5	808	5	0	0
20	DFtoED08	DFtoED08	61	60006	0	1804	6	808	5	0	0
20	DFtoED08	DFtoED08	61	60007	0	1804	7	808	20	2	1
20	DFtoED08	DFtoED08	61	60008	0	1804	8	808	13	1	1
20	DFtoED08	DFtoED08	61	60009	0	1804	9	808	5	0	0
20	DFtoED08	DFtoED08	61	60010	0	1804	10	808	7	0	0
20	DFtoED08	DFtoED08	61	60011	0	1804	11	808	13	1	1
20	DFtoED08	DFtoED08	61	60012	0	1804	12	808	1	0	0
20	DFtoED08	DFtoED08	61	60013	0	1804	13	808	10	1	1
20	DFtoED08	DFtoED08	61	60014	0	1804	14	808	13	1	1
20	DFtoED08	DFtoED08	61	60015	0	1804	15	808	10	1	1
20	DFtoED08	DFtoED08	61	60016	0	1804	16	808	19	2	1

Casualty Treatment Delay Reports (16/3460)

Rept Type	SrcDest	Rptr Type	NA	Juris ID	Time	Rept Cnt	TrackID	Delay_1	Delay_2	Delay_3
21	DFtoED09	62	60001	0	1804	1	908	54	54	55
21	DFtoED09	62	60002	0	1804	2	908	11	11	11
21	DFtoED09	62	60003	0	1804	3	908	17	17	18
21	DFtoED09	62	60004	0	1804	4	908	28	28	29
21	DFtoED09	62	60005	0	1804	5	908	35	35	37
21	DFtoED09	62	60006	0	1804	6	908	36	36	38
21	DFtoED09	62	60007	0	1804	7	908	11	11	11
21	DFtoED09	62	60008	0	1804	8	908	17	17	18
21	DFtoED09	62	60009	0	1804	9	908	36	36	38
21	DFtoED09	62	60010	0	1804	10	908	27	27	29
21	DFtoED09	62	60011	0	1804	11	908	17	17	18
21	DFtoED09	62	60012	0	1804	12	908	55	55	56
21	DFtoED09	62	60013	0	1804	13	908	23	23	24
21	DFtoED09	62	60014	0	1804	14	908	18	18	19
21	DFtoED09	62	60015	0	1804	15	908	22	22	23
21	DFtoED09	62	60016	0	1804	16	908	11	11	11

Ambulance Idle Reports (16/181)

Rept Type	SrcDest	SrcDest	Rptr Type	NA	Juris ID	Time	Rept Cnt	TrackID	Loc_X	Loc_Y	Nr_Node	Pos	OnBrd_1	OnBrd_2
22	DFtoED10	DFtoED10	24	20001	2	15	1	101	911840	3790378	2051798	10000000	0	0
22	DFtoED10	DFtoED10	24	20002	2	15	2	201	911803	3790413	2051798	10000000	0	0
22	DFtoED10	DFtoED10	24	20003	2	15	3	301	911807	3790417	2051798	10000000	0	0
22	DFtoED10	DFtoED10	24	20004	2	15	4	401	911797	3790431	2051798	10000000	0	0
22	DFtoED10	DFtoED10	24	20005	2	15	5	501	911839	3790367	2051798	10000000	0	0
22	DFtoED10	DFtoED10	24	20006	2	15	6	601	903097	3793909	75155672	10000000	0	0
22	DFtoED10	DFtoED10	24	20007	2	15	7	701	903047	3793964	75155672	10000000	0	0
22	DFtoED10	DFtoED10	24	20008	2	15	8	801	903082	3793895	75155672	10000000	0	0
22	DFtoED10	DFtoED10	24	20009	2	15	9	901	903072	3793945	75155672	10000000	0	0
22	DFtoED10	DFtoED10	24	20010	2	15	10	1001	903051	3794000	75155672	10000000	0	0
22	DFtoED10	DFtoED10	24	20011	1	15	11	1101	915771	3789027	1671749	10000000	0	0
22	DFtoED10	DFtoED10	24	20012	1	15	12	1201	915776	3789027	1671749	10000000	0	0
22	DFtoED10	DFtoED10	24	20013	1	15	13	1301	915783	3789062	1671749	10000000	0	0
22	DFtoED10	DFtoED10	24	20014	1	15	14	1401	915739	3789062	1671749	10000000	0	0
22	DFtoED10	DFtoED10	24	20015	1	15	15	1501	915746	3789027	1671749	10000000	0	0
22	DFtoED10	DFtoED10	24	20016	2	15	16	1601	910165	3797619	1935627	10000000	0	0

Ambulance Stuck Reports (16/63)

Rept Type	SrcDest	SrcDest	Rptr Type	NA	Juris ID	Time	Rept Cnt	TrackID	Loc_X	Loc_Y	Nr_Node	Link)ID	OnBrd_1	On_Brd_2
23	DFtoED11	DFtoED11	23	20070	1	367	1	9203	913749	3802969	1902795	1399546	0	0
23	DFtoED11	DFtoED11	23	20083	1	496	2	9003	918388	3793186	1452484	1898424	0	0
23	DFtoED11	DFtoED11	23	20084	1	504	3	9303	918366	3793163	1452484	1898424	0	0
23	DFtoED11	DFtoED11	23	20003	2	512	4	9403	912117	3789938	2051798	75156220	0	0
23	DFtoED11	DFtoED11	23	20006	2	407	5	9503	902301	3793259	75155672	75152413	0	0
23	DFtoED11	DFtoED11	23	20018	2	404	6	9603	910272	3797681	1935627	26352666	0	0
23	DFtoED11	DFtoED11	23	20008	2	410	7	9703	903632	3793321	75155672	26339791	0	0
23	DFtoED11	DFtoED11	23	20009	2	412	8	9803	903624	3793304	75155672	26339791	0	0
23	DFtoED11	DFtoED11	23	20010	2	413	9	9903	903648	3793277	75155672	26339791	0	0
23	DFtoED11	DFtoED11	23	20029	1	420	10	10003	919321	3801025	1455209	1336125	0	0
23	DFtoED11	DFtoED11	23	20011	1	553	11	10103	916323	3789089	1671749	1476134	0	0
23	DFtoED11	DFtoED11	23	20021	1	431	12	10203	918626	3801863	2085674	75153034	0	0
23	DFtoED11	DFtoED11	23	20024	1	435	13	10303	918654	3801933	2085674	75153034	0	0
23	DFtoED11	DFtoED11	23	20022	1	446	14	10403	918684	3801334	2085674	2192079	0	0
23	DFtoED11	DFtoED11	23	20033	1	456	15	10503	919554	3788863	1925692	1467887	0	0
23	DFtoED11	DFtoED11	23	20034	1	462	16	10603	919558	3788838	1925692	1467887	0	0

Travel Delay Reports (16/904)

Rept Type	SrcDest	SrcDest	Rptr Type	NA	Juris ID	Time	Rept Cnt	TrackID	Link_ID	Sev
24	DFtoED12	DFtoED12	29	20029	1	347	1	8503	75166815	0
24	DFtoED12	DFtoED12	29	20033	1	445	2	8603	1388830	0
24	DFtoED12	DFtoED12	29	20035	1	468	3	8703	1925668	0
24	DFtoED12	DFtoED12	29	20049	1	349	4	8803	1634573	0
24	DFtoED12	DFtoED12	29	20053	1	346	5	8903	75165185	0
24	DFtoED12	DFtoED12	29	20055	1	346	6	9003	75165185	0
24	DFtoED12	DFtoED12	29	20056	1	347	7	9103	1898393	0
24	DFtoED12	DFtoED12	29	20021	1	305	8	103	75153073	0
24	DFtoED12	DFtoED12	29	20022	1	306	9	203	75153073	0
24	DFtoED12	DFtoED12	29	20024	1	306	10	303	75153073	0
24	DFtoED12	DFtoED12	29	20025	1	305	11	403	75153073	0
24	DFtoED12	DFtoED12	29	20031	1	309	12	503	2107521	0
24	DFtoED12	DFtoED12	29	20032	1	309	13	603	2107521	0
24	DFtoED12	DFtoED12	29	20056	1	308	14	703	1898364	0
24	DFtoED12	DFtoED12	29	20078	2	422	15	803	1357174	0
24	DFtoED12	DFtoED12	29	20079	2	306	16	903	1357174	0

Cluster Identity Reports (16/33402)   *Note: Table truncated to show only first two Param fields*

Rept Type	SrcDest	Rptr Type	Juris ID	Time	Rept Cnt	TrackID	Clus_ID	Clus_Len	Cell_Cnt	C1_Params	C2_Params...
25	DFtoED13	0	0	454	1	1308	0	150	4	1 920621.90092739649 3795723.2997216624 0 0	1 920366.59610897489 3795873.601731922 0 0
25	DFtoED13	0	0	454	2	1308	1	150	4	1 924349. 3797873. 0 0	1 923943.60098657245 3797921.8014203263 0 0
25	DFtoED13	0	0	454	3	1308	2	150	4	1 920024.33362365223 3801745.6726526767 0 0	1 919719.62558558153 3801844.1264935508 0 0
25	DFtoED13	0	0	454	4	1308	3	150	8	1 921078.01320520043 3802133.4887639959 0 0	1 921126. 3802270. 0 0
25	DFtoED13	0	0	454	5	1308	0	150	4	1 920621.90092739649 3795723.2997216624 0 0	1 920366.59610897489 3795873.601731922 0 0
25	DFtoED13	0	0	454	6	1308	1	150	4	1 924349. 3797873. 0 0	1 923943.60098657245 3797921.8014203263 0 0
25	DFtoED13	0	0	454	7	1308	2	150	4	1 920024.33362365223 3801745.6726526767 0 0	1 919719.62558558153 3801844.1264935508 0 0
25	DFtoED13	0	0	454	8	1308	3	150	8	1 921078.01320520043 3802133.4887639959 0 0	1 921126. 3802270. 0 0
25	DFtoED13	0	0	454	9	1308	0	150	4	1 920621.90092739649 3795723.2997216624 0 0	1 920366.59610897489 3795873.601731922 0 0
25	DFtoED13	0	0	454	10	1308	1	150	4	1 924349. 3797873. 0 0	1 923943.60098657245 3797921.8014203263 0 0
25	DFtoED13	0	0	454	11	1308	2	150	4	1 920024.33362365223 3801745.6726526767 0 0	1 919719.62558558153 3801844.1264935508 0 0
25	DFtoED13	0	0	454	12	1308	3	150	8	1 921078.01320520043 3802133.4887639959 0 0	1 921126. 3802270. 0 0
25	DFtoED13	0	0	454	13	1308	0	150	4	1 920621.90092739649 3795723.2997216624 0 0	1 920366.59610897489 3795873.601731922 0 0
25	DFtoED13	0	0	454	14	1308	1	150	4	1 924349. 3797873. 0 0	1 923943.60098657245 3797921.8014203263 0 0
25	DFtoED13	0	0	454	15	1308	2	150	4	1 920024.33362365223 3801745.6726526767 0 0	1 919719.62558558153 3801844.1264935508 0 0
25	DFtoED13	0	0	454	16	1308	3	150	8	1 921078.01320520043 3802133.4887639959 0 0	1 921126. 3802270. 0 0

Casualty Observation Reports (16/11,138)

Rept Type	SrcDest	SrcDest	Rptr Type	NA	Juris ID	Time	Rept Cnt	TrackID	Loc_X	Loc_Y	CenTr_ID	LErCov_X	LErCov_Y	Cas_ID
39	DFtoL201	DFtoL201	31	0	1	1	1	100	923485	3798971	6.04E+09	400	400	1E+08
39	DFtoL201	DFtoL201	31	0	1	0	2	200	919779	3802921	6.04E+09	400	400	1E+08
39	DFtoL201	DFtoL201	31	0	2	0	3	300	906486	3790344	6.04E+09	400	400	1E+08
39	DFtoL201	DFtoL201	31	0	1	6	4	400	917988	3798270	6.04E+09	400	400	9E+08
39	DFtoL201	DFtoL201	31	0	1	7	5	500	917383	3795675	6.04E+09	400	400	9E+08
39	DFtoL201	DFtoL201	31	0	1	2	6	600	921560	3802299	6.04E+09	400	400	9E+08
39	DFtoL201	DFtoL201	31	0	1	4	7	700	918642	3797872	6.04E+09	400	400	9E+08
39	DFtoL201	DFtoL201	31	0	2	4	8	800	908758	3796287	6.04E+09	400	400	9E+08
39	DFtoL201	DFtoL201	31	0	1	5	9	900	919417	3797511	6.04E+09	400	400	9E+08
39	DFtoL201	DFtoL201	31	0	1	140	10	1000	914692	3798640	6.04E+09	400	400	9E+08
39	DFtoL201	DFtoL201	31	0	1	7	11	1100	916516	3791644	6.04E+09	400	400	9E+08
39	DFtoL201	DFtoL201	31	0	1	9	12	1200	926126	3798654	6.04E+09	400	400	9E+08
39	DFtoL201	DFtoL201	31	0	2	4	13	1300	906698	3798703	6.04E+09	400	400	1E+08
39	DFtoL201	DFtoL201	31	0	1	5	14	1400	914910	3797307	6.04E+09	400	400	9E+08
39	DFtoL201	DFtoL201	31	0	2	7	15	1500	901465	3792210	6.04E+09	400	400	9E+08
39	DFtoL201	DFtoL201	31	0	1	104	16	1600	922948	3793826	6.04E+09	400	400	9E+08

Cas_Age	Cas_Race	Cas_Sex	Inj_Type	Sev	PrSev_1	PrSev_2	PrSev_3	PrSev_4	RC_Pol	RC_Amb	RC_Civ	CumPr	FAR
3	4	2	0	4	0	0.175258	5.15E-02	0.773196	0	0	165	1	3.3
2	2	2	0	1	0.75025	0.16983	3.00E-02	5.00E-02	0	0	165	1	3.3
4	1	1	0	1	0.956688	4.33E-02	0	0	0	0	165	1	3.3
3	2	2	0	1	0.956688	4.33E-02	0	0	0	0	165	1	3.3
4	1	2	0	1	0.956688	4.33E-02	0	0	0	0	165	1	3.3
2	1	1	0	2	0.334692	0.649392	9.55E-03	6.37E-03	0	0	165	1	3.3
1	2	1	0	1	0.956688	4.33E-02	0	0	0	0	165	1	3.3
3	1	1	0	1	0.956688	4.33E-02	0	0	0	0	165	1	3.3
3	2	2	0	1	0.75025	0.16983	3.00E-02	5.00E-02	0	0	165	1	3.3
4	2	2	0	1	0.956688	4.33E-02	0	0	0	0	165	1	3.3
4	2	2	0	1	0.956688	4.33E-02	0	0	0	0	165	1	3.3
4	1	1	0	2	0.334692	0.649392	9.55E-03	6.37E-03	0	0	165	1	3.3
1	1	2	0	1	0.296721	0.156723	0.296326	0.25023	0	0	165	1	3.3
4	2	2	0	2	0.334692	0.649392	9.55E-03	6.37E-03	0	0	165	1	3.3
1	1	2	0	2	0.334692	0.649392	9.55E-03	6.37E-03	0	0	165	1	3.3
3	2	2	0	1	0.956688	4.33E-02	0	0	0	0	165	1	3.3

Casualty Pickup Reports (16/32)

Rept Type	SrcDest	Rptr Type	NA	Juris ID	Time	Rept Cnt	TrackID	Cas_ID	Nr_Node	Inj_Type	Sev
40	DFtoL202	27	20048	1	472	1	200200	900207616	1397380	0	2
40	DFtoL202	27	20023	1	495	2	213700	900215922	1481226	0	2
40	DFtoL202	27	20037	1	688	3	233100	900201559	1434162	0	2
40	DFtoL202	27	20026	1	640	4	282000	100214010	26360966	0	2
40	DFtoL202	27	20038	1	755	5	338200	900201602	1898653	0	2
40	DFtoL202	27	20032	1	827	6	382100	900211733	2194802	0	2
40	DFtoL202	27	20031	1	985	7	400900	900213460	2157695	0	3
40	DFtoL202	27	20025	1	889	8	417600	900209902	2017239	0	2
40	DFtoL202	27	20025	1	889	9	417700	900215808	2017239	0	3
40	DFtoL202	27	20007	2	892	10	424300	900214834	26316570	0	2
40	DFtoL202	27	20037	1	946	11	456200	900201608	1385729	0	3
40	DFtoL202	27	20030	1	1030	12	495200	900211506	1638519	0	2
40	DFtoL202	27	20030	1	1030	13	495300	900211526	1638519	0	1
40	DFtoL202	27	20030	1	1030	14	495400	100210022	1638519	0	2
40	DFtoL202	27	20047	1	1036	15	495500	900208825	1898881	0	3
40	DFtoL202	27	20071	1	1030	16	495600	900216287	1649189	0	1

Ambulance Route Reports (16/232)   *Note: Table truncated to show only first two Segment fields*

Rept Type	SrcDest	Rptr Type	NA	Juris ID	Time	Rept Cnt	Track_ID	Cas_ID	Hosp_ID	Rt-Type	Seg-Cnt	Seg_1	Seg_2...
38	DPtoRG01	0	0	0	304	1	20001	12400	0	0	28	0 2051798	2051851 2051801
38	DPtoRG01	0	0	0	304	2	20002	36800	0	0	32	0 2051798	2051784 2051823
38	DPtoRG01	0	0	0	304	3	20003	32500	0	0	28	0 2051798	26346767 2051835
38	DPtoRG01	0	0	0	304	4	20004	32300	0	0	50	0 2051798	2051851 2051801
38	DPtoRG01	0	0	0	304	5	20005	16400	0	0	41	0 2051798	2051784 2051823
38	DPtoRG01	0	0	0	304	6	20006	1500	0	0	25	0 75155672	75155671 75155668
38	DPtoRG01	0	0	0	304	7	20007	52200	0	0	31	0 75155672	75155671 75155668
38	DPtoRG01	0	0	0	304	8	20008	33100	0	0	34	0 75155672	75155671 75155668
38	DPtoRG01	0	0	0	304	9	20009	51300	0	0	37	0 75155672	75155671 75155668
38	DPtoRG01	0	0	0	304	10	20010	52300	0	0	37	0 75155672	75155671 75155668
38	DPtoRG01	0	0	0	304	11	20011	54700	0	0	10	0 1671749	1671768 1671738
38	DPtoRG01	0	0	0	304	12	20012	38800	0	0	52	0 1671749	1671751 1671767
38	DPtoRG01	0	0	0	304	13	20013	32400	0	0	62	0 1671749	1671751 1671767
38	DPtoRG01	0	0	0	304	14	20014	57000	0	0	51	0 1671749	1671751 1671767
38	DPtoRG01	0	0	0	304	15	20015	26500	0	0	53	0 1671749	1671751 1671767
38	DPtoRG01	0	0	0	304	16	20016	28600	0	0	18	0 1935627	1935632 1935600

Casualty Observation Reports (16/11,137)

Rept Type	SrcDest	Rptr Type	Juris ID	Time	Rept Cnt	TrackID	Loc_X	Loc_Y	Nr_Node	Cas_ID	Sev	PrSev_1	PrSev_2	PrSev_3	PrSev_4
26	EDtoDP01	31	1	125	1	100	923485	3798971	75160557	100203883	4	0	0.175258	5.15E-02	0.773196
26	EDtoDP01	31	1	125	2	200	919779	3802921	1453072	100210789	1	0.75025	0.16983	3.00E-02	5.00E-02
26	EDtoDP01	31	2	125	3	300	906486	3790344	1488661	100209040	1	0.956688	4.33E-02	0	0
26	EDtoDP01	31	1	125	4	400	917988	3798270	1724897	900207072	1	0.956688	4.33E-02	0	0
26	EDtoDP01	31	1	125	5	500	917383	3795675	1899571	900216652	1	0.956688	4.33E-02	0	0
26	EDtoDP01	31	1	125	6	600	921560	3802299	1638512	900210028	2	0.334692	0.649392	9.55E-03	6.37E-03
26	EDtoDP01	31	1	125	7	700	918642	3797872	2093495	900205095	1	0.956688	4.33E-02	0	0
26	EDtoDP01	31	2	125	8	800	908758	3796287	1902993	900212714	1	0.956688	4.33E-02	0	0
26	EDtoDP01	31	1	125	9	900	919417	3797511	2190567	900205203	1	0.75025	0.16983	3.00E-02	5.00E-02
26	EDtoDP01	31	1	125	10	1000	914692	3798640	2029349	900204661	1	0.956688	4.33E-02	0	0
26	EDtoDP01	31	1	125	11	1100	916516	3791644	75151110	900207881	1	0.956688	4.33E-02	0	0
26	EDtoDP01	31	1	125	12	1200	926126	3798654	2008600	900213038	2	0.334692	0.649392	9.55E-03	6.37E-03
26	EDtoDP01	31	2	125	13	1300	906698	3798703	1834907	100210409	1	0.296721	0.156723	0.296326	0.25023
26	EDtoDP01	31	1	125	14	1400	914910	3797307	1481003	900200178	2	0.334692	0.649392	9.55E-03	6.37E-03
26	EDtoDP01	31	2	125	15	1500	901465	3792210	1903707	900212388	2	0.334692	0.649392	9.55E-03	6.37E-03
26	EDtoDP01	31	1	125	16	1600	922948	3793826	1723457	900201968	1	0.956688	4.33E-02	0	0

Roadway Damage Reports

Rept Type	SrcDest	Rptr Type	NA	Juris ID	Time	Rept Cnt	TrackID	Link_ID	Sev
27	EDtoDP04	28	20029	1	545	1	104	1336125	3
27	EDtoDP04	28	20024	1	545	2	204	75153034	3
27	EDtoDP04	28	20022	1	545	3	304	2192079	5
27	EDtoDP04	28	20034	1	545	4	404	1467887	4
27	EDtoDP04	28	20024	1	725	5	204	75153034	3
27	EDtoDP04	28	20022	1	725	6	304	2192079	5
27	EDtoDP04	28	20029	1	725	7	104	1336125	3
27	EDtoDP04	28	20034	1	725	8	404	1467887	4
27	EDtoDP04	28	20035	1	785	9	504	1814195	4
27	EDtoDP04	28	0	1	965	10	604	1722831	4
27	EDtoDP04	28	20012	1	965	11	704	0	4
27	EDtoDP04	28	20035	1	1025	12	504	1814195	4
27	EDtoDP04	28	20014	1	1325	13	704	1722831	4

Medical Facility Capacity Reports (16/3,340)

Rept Type	SrcDest	Rptr Type	NA	Juris ID	Time	Rept Cnt	TrackID	Sev_1	Sev_2	Sev_3
28	EDtoDP05	61	60001	0	1925	1	808	1	0	0
28	EDtoDP05	61	60002	0	1925	2	808	20	2	2
28	EDtoDP05	61	60003	0	1925	3	808	13	1	1
28	EDtoDP05	61	60004	0	1925	4	808	7	0	0
28	EDtoDP05	61	60005	0	1925	5	808	5	0	0
28	EDtoDP05	61	60006	0	1925	6	808	5	0	0
28	EDtoDP05	61	60007	0	1925	7	808	20	2	1
28	EDtoDP05	61	60008	0	1925	8	808	13	1	1
28	EDtoDP05	61	60009	0	1925	9	808	5	0	0
28	EDtoDP05	61	60010	0	1925	10	808	7	0	0
28	EDtoDP05	61	60011	0	1925	11	808	13	1	1
28	EDtoDP05	61	60012	0	1925	12	808	1	0	0
28	EDtoDP05	61	60013	0	1925	13	808	10	1	1
28	EDtoDP05	61	60014	0	1925	14	808	13	1	1
28	EDtoDP05	61	60015	0	1925	15	808	10	1	1
28	EDtoDP05	61	60016	0	1925	16	808	19	2	1

AFOSR F49620-01-1-0371

Travel Delay Reports (16/904)

Rept Type	SrcDest	SrcDest	Rptr Type	NA	Juris ID	Time	Rept Cnt	TrackID	Link_ID	Sev
29	EDtoDP06	EDtoDP06	29	20029	1	425	1	8503	75166815	0
29	EDtoDP06	EDtoDP06	29	20033	1	425	2	8603	1388830	0
29	EDtoDP06	EDtoDP06	29	20035	1	425	3	8703	1925668	0
29	EDtoDP06	EDtoDP06	29	20049	1	425	4	8803	1634573	0
29	EDtoDP06	EDtoDP06	29	20053	1	425	5	8903	75165185	0
29	EDtoDP06	EDtoDP06	29	20055	1	425	6	9003	75165185	0
29	EDtoDP06	EDtoDP06	29	20056	1	425	7	9103	1898393	0
29	EDtoDP06	EDtoDP06	29	20021	1	425	8	103	75153073	0
29	EDtoDP06	EDtoDP06	29	20022	1	425	9	203	75153073	0
29	EDtoDP06	EDtoDP06	29	20024	1	425	10	303	75153073	0
29	EDtoDP06	EDtoDP06	29	20025	1	425	11	403	75153073	0
29	EDtoDP06	EDtoDP06	29	20031	1	425	12	503	2107521	0
29	EDtoDP06	EDtoDP06	29	20032	1	425	13	603	2107521	0
29	EDtoDP06	EDtoDP06	29	20056	1	425	14	703	1898364	0
29	EDtoDP06	EDtoDP06	29	20078	2	425	15	803	1357174	0
29	EDtoDP06	EDtoDP06	29	20079	2	425	16	903	1357174	0

Casualty Treatment Delay Reports (16/3340)

Rept Type	SrcDest	Rptr Type	NA	Juris ID	Time	Rept Cnt	TrackID	Delay_1	Delay_2	Delay_3
30	EDtoDP07	62	60001	0	1925	1	908	54	54	55
30	EDtoDP07	62	60002	0	1925	2	908	11	11	11
30	EDtoDP07	62	60003	0	1925	3	908	17	17	18
30	EDtoDP07	62	60004	0	1925	4	908	28	28	29
30	EDtoDP07	62	60005	0	1925	5	908	35	35	37
30	EDtoDP07	62	60006	0	1925	6	908	36	36	38
30	EDtoDP07	62	60007	0	1925	7	908	11	11	11
30	EDtoDP07	62	60008	0	1925	8	908	17	17	18
30	EDtoDP07	62	60009	0	1925	9	908	36	36	38
30	EDtoDP07	62	60010	0	1925	10	908	27	27	29
30	EDtoDP07	62	60011	0	1925	11	908	17	17	18
30	EDtoDP07	62	60012	0	1925	12	908	55	55	56
30	EDtoDP07	62	60013	0	1925	13	908	23	23	24
30	EDtoDP07	62	60014	0	1925	14	908	18	18	19
30	EDtoDP07	62	60015	0	1925	15	908	22	22	23
30	EDtoDP07	62	60016	0	1925	16	908	11	11	11

Ambulance Idle Reports (16/181)

Rept Type	SrcDest	Rptr Type	NA	Juris ID	Time	Rept Cnt	TrackID	Loc_X	Loc_Y	Nr_Node	Pos	OnBd_2	OnBd_3
31	EDtoDP08	24	20001	2	125	1	101	911840	3790378	2051798	10000000	0	0
31	EDtoDP08	24	20002	2	125	2	201	911803	3790413	2051798	10000000	0	0
31	EDtoDP08	24	20003	2	125	3	301	911807	3790417	2051798	10000000	0	0
31	EDtoDP08	24	20004	2	125	4	401	911797	3790431	2051798	10000000	0	0
31	EDtoDP08	24	20005	2	125	5	501	911839	3790367	2051798	10000000	0	0
31	EDtoDP08	24	20006	2	125	6	601	903097	3793909	75155672	10000000	0	0
31	EDtoDP08	24	20007	2	125	7	701	903047	3793964	75155672	10000000	0	0
31	EDtoDP08	24	20008	2	125	8	801	903082	3793895	75155672	10000000	0	0
31	EDtoDP08	24	20009	2	125	9	901	903072	3793945	75155672	10000000	0	0
31	EDtoDP08	24	20010	2	125	10	1001	903051	3794000	75155672	10000000	0	0
31	EDtoDP08	24	20011	1	125	11	1101	915771	3789027	1671749	10000000	0	0
31	EDtoDP08	24	20012	1	125	12	1201	915776	3789027	1671749	10000000	0	0
31	EDtoDP08	24	20013	1	125	13	1301	915783	3789062	1671749	10000000	0	0
31	EDtoDP08	24	20014	1	125	14	1401	915739	3789062	1671749	10000000	0	0
31	EDtoDP08	24	20015	1	125	15	1501	915746	3789027	1671749	10000000	0	0
31	EDtoDP08	24	20016	2	125	16	1601	910165	3797619	1935627	10000000	0	0

Ambulance Stuck Reports (16/63)

Rept Type	SrcDest	Rptr Type	NA	Juris ID	Time	Rept Cnt	TrackID	Loc_X	Loc_Y	Nr_Node	Link_ID	OnBd_2	OnBd_3
32	EDtoDP09	23	20070	1	485	1	9203	913749	3802969	1902795	1399546	0	0
32	EDtoDP09	23	20083	1	485	2	9003	918388	3793186	1452484	1898424	0	0
32	EDtoDP09	23	20084	1	485	3	9303	918366	3793163	1452484	1898424	0	0
32	EDtoDP09	23	20003	2	485	4	9403	912117	3789938	2051798	75156220	0	0
32	EDtoDP09	23	20006	2	485	5	9503	902301	3793259	75155672	75152413	0	0
32	EDtoDP09	23	20018	2	485	6	9603	910272	3797681	1935627	26352666	0	0
32	EDtoDP09	23	20008	2	545	7	9703	903632	3793321	75155672	26339791	0	0
32	EDtoDP09	23	20009	2	545	8	9803	903624	3793304	75155672	26339791	0	0
32	EDtoDP09	23	20010	2	545	9	9903	903648	3793277	75155672	26339791	0	0
32	EDtoDP09	23	20029	1	545	10	10003	919321	3801025	1455209	1336125	0	0
32	EDtoDP09	23	20011	1	545	11	10103	916323	3789089	1671749	1476134	0	0
32	EDtoDP09	23	20021	1	545	12	10203	918626	3801863	2085674	75153034	0	0
32	EDtoDP09	23	20024	1	545	13	10303	918654	3801933	2085674	75153034	0	0
32	EDtoDP09	23	20022	1	545	14	10403	918684	3801334	2085674	2192079	0	0
32	EDtoDP09	23	20033	1	545	15	10503	919554	3788863	1925692	1467887	0	0
32	EDtoDP09	23	20034	1	545	16	10603	919558	3788838	1925692	1467887	0	0

Cluster Indentity Reports (16/48,032) *Note: Table truncated to show only first two Param fields*

Rept Type	SrcDest	Rptr Type	Juris ID	Time	Rept Cnt	TrackID	Clus_ID	Clus_Len	Cell_Cnt	Cell 1 Params	Cell 2 Params...
33	EDtoDP10	0	0	545	1	1308	0	150	4	1 920621.90092739649 3795723.2997216624 0 0	1 920366.59610897489 3795873.601731922 0 0
33	EDtoDP10	0	0	545	2	1308	1	150	4	1 924349. 3797873. 0 0	1 923943.60098657245 3797921.8014203263 0 0
33	EDtoDP10	0	0	545	3	1308	2	150	4	1 920024.33362365223 3801745.6726526767 0 0	1 919719.62558558153 3801844.1264935508 0 0
33	EDtoDP10	0	0	545	4	1308	3	150	8	1 921078.01320520043 3802133.4887639959 0 0	1 921126. 3802270. 0 0
33	EDtoDP10	0	0	545	5	1308	0	150	4	1 920621.90092739649 3795723.2997216624 0 0	1 920366.59610897489 3795873.601731922 0 0
33	EDtoDP10	0	0	545	6	1308	1	150	4	1 924349. 3797873. 0 0	1 923943.60098657245 3797921.8014203263 0 0
33	EDtoDP10	0	0	545	7	1308	2	150	4	1 920024.33362365223 3801745.6726526767 0 0	1 919719.62558558153 3801844.1264935508 0 0
33	EDtoDP10	0	0	545	8	1308	3	150	8	1 921078.01320520043 3802133.4887639959 0 0	1 921126. 3802270. 0 0
33	EDtoDP10	0	0	605	9	1308	0	150	4	1 920621.90092739649 3795723.2997216624 0 0	1 920366.59610897489 3795873.601731922 0 0
33	EDtoDP10	0	0	605	10	1308	1	150	4	1 924349. 3797873. 0 0	1 923943.60098657245 3797921.8014203263 0 0
33	EDtoDP10	0	0	605	11	1308	2	150	4	1 920024.33362365223 3801745.6726526767 0 0	1 919719.62558558153 3801844.1264935508 0 0
33	EDtoDP10	0	0	605	12	1308	3	150	8	1 921078.01320520043 3802133.4887639959 0 0	1 921126. 3802270. 0 0
33	EDtoDP10	0	0	605	13	1308	0	150	4	1 920621.90092739649 3795723.2997216624 0 0	1 920366.59610897489 3795873.601731922 0 0
33	EDtoDP10	0	0	605	14	1308	1	150	4	1 924349. 3797873. 0 0	1 923943.60098657245 3797921.8014203263 0 0
33	EDtoDP10	0	0	605	15	1308	2	150	4	1 920024.33362365223 3801745.6726526767 0 0	1 919719.62558558153 3801844.1264935508 0 0
33	EDtoDP10	0	0	605	16	1308	3	150	8	1 921078.01320520043 3802133.4887639959 0 0	1 921126. 3802270. 0 0

Casualty Delivery Reports (16/5656)

Rept Type	SrcDest	Rptr Type	Juris ID	Time	Rept Cnt	TrackID	Hosp_ID	Cas_ID	Cas_Sex
35	EDtoMF02	25	2	185	1	26000	0	100208501	1
35	EDtoMF02	25	1	245	2	39500	0	100208430	2
35	EDtoMF02	25	1	245	3	43000	0	100208191	2
35	EDtoMF02	25	1	245	4	43100	0	100208191	1
35	EDtoMF02	25	1	245	5	43200	0	100208191	2
35	EDtoMF02	25	1	245	6	47100	0	100200021	1
35	EDtoMF02	25	2	245	7	50400	0	100211631	1
35	EDtoMF02	25	2	245	8	50500	0	100211631	1
35	EDtoMF02	25	2	245	9	50600	0	100211631	1
35	EDtoMF02	25	2	245	10	50700	0	100211631	1
35	EDtoMF02	25	1	245	11	57300	0	100200747	1
35	EDtoMF02	25	1	245	12	57400	0	100203172	2
35	EDtoMF02	25	1	305	13	63900	0	100214932	1
35	EDtoMF02	25	1	305	14	73700	0	100216361	2
35	EDtoMF02	25	1	305	15	73800	0	100216361	1
35	EDtoMF02	25	1	305	16	73900	0	100216361	1

Casualty Observation Reports (16/11,138)

Rept Type	SrcDest	Rptr Type	Juris ID	Time	Rept Cnt	TrackID	Loc_X	Loc_Y	ErCov_X	ErCov_Y	Nr_Node
43	EDtoVZ01	31	1	125	1	100	923485	3798971	400	400	75160557
43	EDtoVZ01	31	1	125	2	200	919779	3802921	400	400	1453072
43	EDtoVZ01	31	2	125	3	300	906486	3790344	400	400	1488661
43	EDtoVZ01	31	1	125	4	400	917988	3798270	400	400	1724897
43	EDtoVZ01	31	1	125	5	500	917383	3795675	400	400	1899571
43	EDtoVZ01	31	1	125	6	600	921560	3802299	400	400	1638512
43	EDtoVZ01	31	1	125	7	700	918642	3797872	400	400	2093495
43	EDtoVZ01	31	2	125	8	800	908758	3796287	400	400	1902993
43	EDtoVZ01	31	1	125	9	900	919417	3797511	400	400	2190567
43	EDtoVZ01	31	1	125	10	1000	914692	3798640	400	400	2029349
43	EDtoVZ01	31	1	125	11	1100	916516	3791644	400	400	75151110
43	EDtoVZ01	31	1	125	12	1200	926126	3798654	400	400	2008600
43	EDtoVZ01	31	2	125	13	1300	906698	3798703	400	400	1834907
43	EDtoVZ01	31	1	125	14	1400	914910	3797307	400	400	1481003
43	EDtoVZ01	31	2	125	15	1500	901465	3792210	400	400	1903707
43	EDtoVZ01	31	1	125	16	1600	922948	3793826	400	400	1723457

Cas_ID	Cas_Age	Cas_Race	Cas_Sex	Sev	PrSev_1	PrSev_2	PrSev_3	PrSev_4	RC_Pol	RC_Amb	RC_Civ	CumProb
1E+08	3	4	2	4	0	0.175258	5.15E-02	0.773196	0	0	165	1
1E+08	2	2	2	1	0.75025	0.16983	3.00E-02	5.00E-02	0	0	165	1
1E+08	4	1	1	1	0.956688	4.33E-02	0	0	0	0	165	1
9E+08	3	2	2	1	0.956688	4.33E-02	0	0	0	0	165	1
9E+08	4	1	2	1	0.956688	4.33E-02	0	0	0	0	165	1
9E+08	2	1	1	2	0.334692	0.649392	9.55E-03	6.37E-03	0	0	165	1
9E+08	1	2	1	1	0.956688	4.33E-02	0	0	0	0	165	1
9E+08	3	1	1	1	0.956688	4.33E-02	0	0	0	0	165	1
9E+08	3	2	2	1	0.75025	0.16983	3.00E-02	5.00E-02	0	0	165	1
9E+08	4	2	2	1	0.956688	4.33E-02	0	0	0	0	165	1
9E+08	4	2	2	1	0.956688	4.33E-02	0	0	0	0	165	1
9E+08	4	1	1	2	0.334692	0.649392	9.55E-03	6.37E-03	0	0	165	1
1E+08	1	1	2	1	0.296721	0.156723	0.296326	0.25023	0	0	165	1
9E+08	4	2	2	2	0.334692	0.649392	9.55E-03	6.37E-03	0	0	165	1
9E+08	1	1	2	2	0.334692	0.649392	9.55E-03	6.37E-03	0	0	165	1
9E+08	3	2	2	1	0.956688	4.33E-02	0	0	0	0	165	1

Medical Facility Capacity Reports (16/3340)

Rept Type	SrcDest	Rptr Type	NA	Juris ID	Time	Rept Cnt	TrackID	Sev_1	Sev_2	Sev_3
45	EDtoVZ03	61	60001	0	1925	1	808	1	0	0
45	EDtoVZ03	61	60002	0	1925	2	808	20	2	2
45	EDtoVZ03	61	60003	0	1925	3	808	13	1	1
45	EDtoVZ03	61	60004	0	1925	4	808	7	0	0
45	EDtoVZ03	61	60005	0	1925	5	808	5	0	0
45	EDtoVZ03	61	60006	0	1925	6	808	5	0	0
45	EDtoVZ03	61	60007	0	1925	7	808	20	2	1
45	EDtoVZ03	61	60008	0	1925	8	808	13	1	1
45	EDtoVZ03	61	60009	0	1925	9	808	5	0	0
45	EDtoVZ03	61	60010	0	1925	10	808	7	0	0
45	EDtoVZ03	61	60011	0	1925	11	808	13	1	1
45	EDtoVZ03	61	60012	0	1925	12	808	1	0	0
45	EDtoVZ03	61	60013	0	1925	13	808	10	1	1
45	EDtoVZ03	61	60014	0	1925	14	808	13	1	1
45	EDtoVZ03	61	60015	0	1925	15	808	10	1	1
45	EDtoVZ03	61	60016	0	1925	16	808	19	2	1

Cluster Indentification Reports (16/48,031) *Note: Table truncated to show only first few Param fields*

Rept Type	SrcDest	Rptr Type	Juris ID	Time	Rept Cnt	TrackID	Clus_ID	Clus_Len	Cell_Cnt	Cell 1 Params	Cell 2 Params
46	EDtoVZ04	0	0	545	1	1308	0	150	4	1 920621.90092739649 3795723.2997216624 0 0	1 920459. 3795883. 0 0
46	EDtoVZ04	0	0	545	2	1308	1	150	4	1 924349. 3797873. 0 0	1 924098.14369439124 3797937.7151631867 0 0
46	EDtoVZ04	0	0	545	3	1308	2	150	4	1 920024.33362365223 3801745.6726526767 0 0	1 919862.66776124085 3801859.6622065217 0 0
46	EDtoVZ04	0	0	545	4	1308	3	150	8	1 921078.01320520043 3802133.4887639959 0 0	1 920613. 3802423. 0 0
46	EDtoVZ04	0	0	545	5	1308	0	150	4	1 920621.90092739649 3795723.2997216624 0 0	1 920459. 3795883. 0 0
46	EDtoVZ04	0	0	545	6	1308	1	150	4	1 924349. 3797873. 0 0	1 924098.14369439124 3797937.7151631867 0 0
46	EDtoVZ04	0	0	545	7	1308	2	150	4	1 920024.33362365223 3801745.6726526767 0 0	1 919862.66776124085 3801859.6622065217 0 0
46	EDtoVZ04	0	0	545	8	1308	3	150	8	1 921078.01320520043 3802133.4887639959 0 0	1 920613. 3802423. 0 0
46	EDtoVZ04	0	0	605	9	1308	0	150	4	1 920621.90092739649 3795723.2997216624 0 0	1 920459. 3795883. 0 0
46	EDtoVZ04	0	0	605	10	1308	1	150	4	1 924349. 3797873. 0 0	1 924098.14369439124 3797937.7151631867 0 0
46	EDtoVZ04	0	0	605	11	1308	2	150	4	1 920024.33362365223 3801745.6726526767 0 0	1 919862.66776124085 3801859.6622065217 0 0
46	EDtoVZ04	0	0	605	12	1308	3	150	8	1 921078.01320520043 3802133.4887639959 0 0	1 920613. 3802423. 0 0
46	EDtoVZ04	0	0	605	13	1308	0	150	4	1 920621.90092739649 3795723.2997216624 0 0	1 920459. 3795883. 0 0
46	EDtoVZ04	0	0	605	14	1308	1	150	4	1 924349. 3797873. 0 0	1 924098.14369439124 3797937.7151631867 0 0
46	EDtoVZ04	0	0	605	15	1308	2	150	4	1 920024.33362365223 3801745.6726526767 0 0	1 919862.66776124085 3801859.6622065217 0 0
46	EDtoVZ04	0	0	605	16	1308	3	150	8	1 921078.01320520043 3802133.4887639959 0 0	1 920613. 3802423. 0 0

Roadway Damage Reports

Rept Type	SrcDest	Rptr Type	NA	Juris ID	Time	Rept Cnt	TrackID	Bridge_ID	Link)ID	Sev
48	EDtoVZ06	28	20029	1	545	1	104	80458	1336125	3
48	EDtoVZ06	28	20024	1	545	2	204	80089	75153034	3
48	EDtoVZ06	28	20022	1	545	3	304	80463	2192079	5
48	EDtoVZ06	28	20034	1	545	4	404	80349	1467887	4
48	EDtoVZ06	28	20024	1	725	5	204	80089	75153034	3
48	EDtoVZ06	28	20022	1	725	6	304	80463	2192079	5
48	EDtoVZ06	28	20029	1	725	7	104	80458	1336125	3
48	EDtoVZ06	28	20034	1	725	8	404	80349	1467887	4
48	EDtoVZ06	28	20035	1	785	9	504	80101	1814195	4
48	EDtoVZ06	28	0	1	965	10	604	0	1722831	4
48	EDtoVZ06	28	20012	1	965	11	704	80103	0	4
48	EDtoVZ06	28	20035	1	1025	12	504	80101	1814195	4
48	EDtoVZ06	28	20014	1	1325	13	704	80103	1722831	4

Cluster Identification Reports (16/700)   *Note: Table truncated to show only first two Param fields*

Rept Type	SrcDest	Rptr Type	Juris ID	Time	Rept Cnt	Clus_ID	Cell_Len	Cell_Cnt	Cell 1 Params	Cell 2 Params...
41	L2toRG01	0	0	454	1	0	150	4	1 920621.90092739649 3795723.2997216624 0 0	1 920366.59610897489 3795873.601731922 0 0
41	L2toRG01	0	0	454	2	1	150	4	1 924349. 3797873. 0 0	1 923943.60098657245 3797921.8014203263 0 0
41	L2toRG01	0	0	454	3	2	150	4	1 920024.33362365223 3801745.6726526767 0 0	1 919719.62558558153 3801844.1264935508 0 0
41	L2toRG01	0	0	454	4	3	150	8	1 921078.01320520043 3802133.4887639959 0 0	1 921126. 3802270. 0 0
41	L2toRG01	0	0	604	5	0	150	4	1 916640.59994875628 3788851.4003188782 0 0	1 916711.42846154328 3788838.9997507944 0 0
41	L2toRG01	0	0	604	6	1	150	2	1 920621.9013752006 3795723.2994241701 0 0	1 920366.59397341241 3795873.6021811445 0 0
41	L2toRG01	0	0	604	7	2	150	3	1 924218.50043045415 3797697.251373624 0 0	1 923943.60385692376 3797921.8053470668 0 0
41	L2toRG01	0	0	604	8	3	150	6	1 920024.33348770277 3801745.6734671337 0 0	1 919719.62529914442 3801844.1282049729 0 0
41	L2toRG01	0	0	754	9	0	150	5	1 916640.59994873998 3788851.4003189793 0 5	1 916711.42752124416 3788839.0004158448 0 7
41	L2toRG01	0	0	754	10	1	150	3	1 920621.90182332438 3795723.2991264649 0 0	1 920366.59183701547 3795873.6026305426 0 0
41	L2toRG01	0	0	754	11	2	150	8	1 924217.25055030861 3797694.7533275667 0 0	1 924187.5036186683 3797740.999413189 0 0
41	L2toRG01	0	0	754	12	3	150	8	1 919859.49370888714 3801735.5028981529 0 0	1 920024.33335163083 3801745.6742823245 0 0
41	L2toRG01	0	0	904	13	0	150	5	1 916640.59994872368 3788851.4003190803 0 0	1 916711.42658076191 3788839.0010810248 0 0
41	L2toRG01	0	0	904	14	1	150	8	1 917787.50761004945 3793756.9871379444 0 0	1 918357.49936711462 3793753.4997890382 0 0
41	L2toRG01	0	0	904	15	2	150	3	1 920621.90227176796 3795723.2988285474 0 0	1 920366.58969978371 3795873.6030801162 0 0
41	L2toRG01	0	0	904	16	3	150	8	1 924217.25044694601 3797694.7550515765 0 0	1 924187.50450919895 3797740.9992687786 0 0

Medical Facility Capacity Reports (16/20)

Rept Type	SrcDest	Rptr Type	NA	Juris ID	Time	Rept Cnt	TrackID	Sev_1	Sev_2	Sev_3
36	MFtoRG01	61	60001	0	1804	1	0	1	0	0
36	MFtoRG01	61	60002	0	1804	2	0	20	2	2
36	MFtoRG01	61	60003	0	1804	3	0	13	1	1
36	MFtoRG01	61	60004	0	1804	4	0	7	0	0
36	MFtoRG01	61	60005	0	1804	5	0	5	0	0
36	MFtoRG01	61	60006	0	1804	6	0	5	0	0
36	MFtoRG01	61	60007	0	1804	7	0	20	2	1
36	MFtoRG01	61	60008	0	1804	8	0	13	1	1
36	MFtoRG01	61	60009	0	1804	9	0	5	0	0
36	MFtoRG01	61	60010	0	1804	10	0	7	0	0
36	MFtoRG01	61	60011	0	1804	11	0	13	1	1
36	MFtoRG01	61	60012	0	1804	12	0	1	0	0
36	MFtoRG01	61	60013	0	1804	13	0	10	1	1
36	MFtoRG01	61	60014	0	1804	14	0	13	1	1
36	MFtoRG01	61	60015	0	1804	15	0	10	1	1
36	MFtoRG01	61	60016	0	1804	16	0	19	2	1

Casualty Treatment Delay Reports (16/20)

Rept Type	SrcDest	Rptr Type	NA	Juris ID	Time	Rept Cnt	TrackID	Dealy_1	Delay_2	Delay_3
37	MFtoRG02	62	60001	0	1804	1	0	54	54	55
37	MFtoRG02	62	60002	0	1804	2	0	11	11	11
37	MFtoRG02	62	60003	0	1804	3	0	17	17	18
37	MFtoRG02	62	60004	0	1804	4	0	28	28	29
37	MFtoRG02	62	60005	0	1804	5	0	35	35	37
37	MFtoRG02	62	60006	0	1804	6	0	36	36	38
37	MFtoRG02	62	60007	0	1804	7	0	11	11	11
37	MFtoRG02	62	60008	0	1804	8	0	17	17	18
37	MFtoRG02	62	60009	0	1804	9	0	36	36	38
37	MFtoRG02	62	60010	0	1804	10	0	27	27	29
37	MFtoRG02	62	60011	0	1804	11	0	17	17	18
37	MFtoRG02	62	60012	0	1804	12	0	55	55	56
37	MFtoRG02	62	60013	0	1804	13	0	23	23	24
37	MFtoRG02	62	60014	0	1804	14	0	18	18	19
37	MFtoRG02	62	60015	0	1804	15	0	22	22	23
37	MFtoRG02	62	60016	0	1804	16	0	11	11	11

Casualty Observation Reports (16/11,228)

Rept Type	SrcDest	Rptr Type	Juris ID	Time	Rept Cnt	TrackID	Loc_X	Loc_Y	CenTrack	Nr_Node	Cas_ID	Cas_Age	Cas_Race	Cas_Sex	Inj_Type	Sev
0	RGtoDF01	31	1	1	1	0	923485	3798971	6037104701	75160557	100203883	3	4	2	0	4
0	RGtoDF01	31	1	0	2	0	919779	3802921	6037120000	1453072	100210789	2	2	2	0	0
0	RGtoDF01	31	2	0	3	0	906486	3790344	6037111301	1488661	100209040	4	1	1	0	1
0	RGtoDF01	31	1	6	4	0	917988	3798270	6037133101	1724897	900207072	6	2	2	0	1
0	RGtoDF01	31	1	7	5	0	917383	3795675	6037104800	1899571	900216652	5	1	2	0	1
0	RGtoDF01	31	1	2	6	0	921560	3802299	6037127601	1638512	900210028	2	1	1	0	2
0	RGtoDF01	31	1	4	7	0	918642	3797872	6037137102	2093495	900205095	1	2	1	0	1
0	RGtoDF01	31	2	4	8	0	908758	3796287	6037128600	1902993	900212714	3	1	1	0	1
0	RGtoDF01	31	1	5	9	0	919417	3797511	6037137102	2190567	900205203	6	5	2	0	0
0	RGtoDF01	31	1	140	10	0	914692	3798640	6037134500	2029349	900204661	5	3	2	0	1
0	RGtoDF01	31	1	7	11	0	916516	3791644	6037113301	75151110	900207881	5	2	2	0	1
0	RGtoDF01	31	1	9	12	0	926126	3798654	6037141302	2008600	900213038	4	1	1	0	2
0	RGtoDF01	31	2	4	13	0	906698	3798703	6037128301	1834907	100210409	1	1	2	0	3
0	RGtoDF01	31	1	5	14	0	914910	3797307	6037134421	1481003	900200178	4	2	2	0	2
0	RGtoDF01	31	2	7	15	0	901465	3792210	6037109602	1903707	900212388	1	1	3	0	2
0	RGtoDF01	31	1	104	16	0	922948	3793826	6037320200	1723457	900201968	3	5	2	0	1

Roadway Damage Reports (16/18)

Rept Type	SrcDest	Rptr Type	NA	Juris ID	Time	Rept Cnt	TrackID	BridgeID	Link_ID	Sev
2	RGtoDF03	28	20029	1	430	1	0	80458	1336125	2
2	RGtoDF03	28	20021	1	440	2	0	80089	75153034	2
2	RGtoDF03	28	20024	1	440	3	0	80089	75153034	2
2	RGtoDF03	28	20022	1	450	4	0	80463	2192079	4
2	RGtoDF03	28	20033	1	460	5	0	80349	1467887	4
2	RGtoDF03	28	20034	1	470	6	0	80349	1467887	3
2	RGtoDF03	28	20021	1	610	7	0	80089	75153034	2
2	RGtoDF03	28	20022	1	726	8	0	80463	2192079	4
2	RGtoDF03	28	20024	1	732	9	0	80089	75153034	2
2	RGtoDF03	28	20029	1	720	10	0	80458	1336125	2
2	RGtoDF03	28	20033	1	610	11	0	80349	1467887	3
2	RGtoDF03	28	20034	1	728	12	0	80349	1467887	3
2	RGtoDF03	28	20035	1	680	13	0	80101	1814195	3
2	RGtoDF03	28	0	1	880	14	0	0	1722831	3
2	RGtoDF03	28	20012	1	890	15	0	80103	0	3
2	RGtoDF03	28	20035	1	910	16	0	80101	1814195	3

AFOSR F49620-01-1-0371

Casualty Pickup Reports (16/32)

Rept Type	SrcDest	Rptr Type	NA	Juris ID	Time	Rept Cnt	Loc_X	Loc_Y	Nr_Node	Cas_ID	Cas_Age	Cas_Race	Cas_sex	Inj_Type	Sev
3	RGtoDF04	27	20048	1	472	1	919638	3795966	1397380	900207616	1	1	1	0	2
3	RGtoDF04	27	20023	1	495	2	917224	3801846	1481226	900215922	6	2	2	0	2
3	RGtoDF04	27	20037	1	688	3	923119	3796968	1434162	900201559	1	2	2	0	2
3	RGtoDF04	27	20026	1	640	4	919751	3801925	26360966	100214010	5	5	1	0	2
3	RGtoDF04	27	20038	1	755	5	924731	3796556	1898653	900201602	5	1	2	0	2
3	RGtoDF04	27	20032	1	827	6	919402	3792400	2194802	900211733	1	2	3	0	2
3	RGtoDF04	27	20031	1	985	7	919135	3792286	2157695	900213460	5	1	2	0	3
3	RGtoDF04	27	20025	1	889	8	919323	3803437	2017239	900209902	3	2	2	0	2
3	RGtoDF04	27	20025	1	889	9	919375	3803376	2017239	900215808	3	2	1	0	3
3	RGtoDF04	27	20007	2	892	10	905233	3795125	26316570	900214834	3	4	1	0	2
3	RGtoDF04	27	20037	1	946	11	923172	3796990	1385729	900201608	2	2	0	0	3
3	RGtoDF04	27	20030	1	1030	12	921117	3802071	1638519	900211506	4	1	1	0	2
3	RGtoDF04	27	20030	1	1030	13	921067	3802012	1638519	900211526	2	1	1	0	5
3	RGtoDF04	27	20030	1	1030	14	921035	3802036	1638519	100210022	4	5	1	0	2
3	RGtoDF04	27	20047	1	1036	15	920982	3795645	1898881	900208825	6	1	2	0	3
3	RGtoDF04	27	20071	1	1030	16	913670	3800340	1649189	900216287	4	4	3	0	1

Casualty Delivery Reports (16/5798)

Rept Type	SrcDest	Rptr Type	Juris ID	Time	Rept Cnt	TrackID	Loc_X	Loc_Y	Nr_Node	Cas_ID	Cas_Age	Cas_Race	Cas_Sex	Inj_Type	Sev
4	RGtoDF05	25	2	77	1	910993	3797156	1834548	60007	100208501	5	1	1	0	1
4	RGtoDF05	25	1	214	2	920539	3801100	1724314	60009	100208430	3	2	2	0	2
4	RGtoDF05	25	1	123	3	917862	3795701	1725024	60015	100208191	5	2	2	0	2
4	RGtoDF05	25	1	123	4	917862	3795701	1725024	60015	100208191	4	2	1	0	1
4	RGtoDF05	25	1	257	5	917862	3795701	1725024	60015	100208191	5	2	2	0	2
4	RGtoDF05	25	1	140	6	921431	3796956	2100876	60014	100200021	3	3	0	0	0
4	RGtoDF05	25	2	275	7	904836	3791578	1732516	60018	100211631	4	1	1	0	1
4	RGtoDF05	25	2	267	8	904836	3791578	1732516	60018	100211631	4	2	1	0	1
4	RGtoDF05	25	2	148	9	904836	3791578	1732516	60018	100211631	3	4	1	0	1
4	RGtoDF05	25	2	293	10	904836	3791578	1732516	60018	100211631	4	1	1	0	1
4	RGtoDF05	25	1	281	11	915097	3793982	26330470	60016	100200747	5	1	1	0	1
4	RGtoDF05	25	1	295	12	922680	3799402	1899141	60011	100203172	3	4	2	0	2
4	RGtoDF05	25	1	305	13	920360	3794235	1450335	60015	100214932	5	3	1	0	1
4	RGtoDF05	25	1	333	14	916078	3799676	2114445	60008	100216361	3	1	2	0	2
4	RGtoDF05	25	1	220	15	916078	3799676	2114445	60008	100216361	3	1	1	0	1
4	RGtoDF05	25	1	220	16	916078	3799676	2114445	60008	100216361	3	1	1	0	1

Medical Facility Capacity Reports (16/3580)

Rept Type	SrcDest	Rptr Type	NA	Juris ID	Time	Rept Cnt	TrackID	Sev_1	Sev_2	Sev_3
7	RGtoDF08	61	60001	0	1804	1	0	1	0	0
7	RGtoDF08	61	60002	0	1804	2	0	20	2	2
7	RGtoDF08	61	60003	0	1804	3	0	13	1	1
7	RGtoDF08	61	60004	0	1804	4	0	7	0	0
7	RGtoDF08	61	60005	0	1804	5	0	5	0	0
7	RGtoDF08	61	60006	0	1804	6	0	5	0	0
7	RGtoDF08	61	60007	0	1804	7	0	20	2	1
7	RGtoDF08	61	60008	0	1804	8	0	13	1	1
7	RGtoDF08	61	60009	0	1804	9	0	5	0	0
7	RGtoDF08	61	60010	0	1804	10	0	7	0	0
7	RGtoDF08	61	60011	0	1804	11	0	13	1	1
7	RGtoDF08	61	60012	0	1804	12	0	1	0	0
7	RGtoDF08	61	60013	0	1804	13	0	10	1	1
7	RGtoDF08	61	60014	0	1804	14	0	13	1	1
7	RGtoDF08	61	60015	0	1804	15	0	10	1	1
7	RGtoDF08	61	60016	0	1804	16	0	19	2	1

Casualty Treatment Delay Reports (16/3580)

Rept Type	SrcDest	Rptr Type	NA	Juris ID	Time	Rept Cnt	TrackID	Delay_1	Delay_2	Delay_3
8	RGtoDF09	62	60001	0	1804	1	0	54	54	55
8	RGtoDF09	62	60002	0	1804	2	0	11	11	11
8	RGtoDF09	62	60003	0	1804	3	0	17	17	18
8	RGtoDF09	62	60004	0	1804	4	0	28	28	29
8	RGtoDF09	62	60005	0	1804	5	0	35	35	37
8	RGtoDF09	62	60006	0	1804	6	0	36	36	38
8	RGtoDF09	62	60007	0	1804	7	0	11	11	11
8	RGtoDF09	62	60008	0	1804	8	0	17	17	18
8	RGtoDF09	62	60009	0	1804	9	0	36	36	38
8	RGtoDF09	62	60010	0	1804	10	0	27	27	29
8	RGtoDF09	62	60011	0	1804	11	0	17	17	18
8	RGtoDF09	62	60012	0	1804	12	0	55	55	56
8	RGtoDF09	62	60013	0	1804	13	0	23	23	24
8	RGtoDF09	62	60014	0	1804	14	0	18	18	19
8	RGtoDF09	62	60015	0	1804	15	0	22	22	23
8	RGtoDF09	62	60016	0	1804	16	0	11	11	11

Ambulance Idle Reports (16/181)

Rept Type	SrcDest	SrcDest	Rptr Type	NA	Juris ID	Time	Rept Cnt	TrackID	Loc_X	Loc_Y	Pos_ID	OnBd_2	OnBd_3
9	RGtoDF10	RGtoDF10	24	20001	2	15	1	911840	3790378	2051798	10000000	0	0
9	RGtoDF10	RGtoDF10	24	20002	2	15	2	911803	3790413	2051798	10000000	0	0
9	RGtoDF10	RGtoDF10	24	20003	2	15	3	911807	3790417	2051798	10000000	0	0
9	RGtoDF10	RGtoDF10	24	20004	2	15	4	911797	3790431	2051798	10000000	0	0
9	RGtoDF10	RGtoDF10	24	20005	2	15	5	911839	3790367	2051798	10000000	0	0
9	RGtoDF10	RGtoDF10	24	20006	2	15	6	903097	3793909	75155672	10000000	0	0
9	RGtoDF10	RGtoDF10	24	20007	2	15	7	903047	3793964	75155672	10000000	0	0
9	RGtoDF10	RGtoDF10	24	20008	2	15	8	903082	3793895	75155672	10000000	0	0
9	RGtoDF10	RGtoDF10	24	20009	2	15	9	903072	3793945	75155672	10000000	0	0
9	RGtoDF10	RGtoDF10	24	20010	2	15	10	903051	3794000	75155672	10000000	0	0
9	RGtoDF10	RGtoDF10	24	20011	1	15	11	915771	3789027	1671749	10000000	0	0
9	RGtoDF10	RGtoDF10	24	20012	1	15	12	915776	3789027	1671749	10000000	0	0
9	RGtoDF10	RGtoDF10	24	20013	1	15	13	915783	3789062	1671749	10000000	0	0
9	RGtoDF10	RGtoDF10	24	20014	1	15	14	915739	3789062	1671749	10000000	0	0
9	RGtoDF10	RGtoDF10	24	20015	1	15	15	915746	3789027	1671749	10000000	0	0
9	RGtoDF10	RGtoDF10	24	20016	2	15	16	910165	3797619	1935627	10000000	0	0

Ambulance Stuck Reports (16/70)

Rept Type	SrcDest	Rptr Type	NA	Juris ID	Time	Rept Cnt	TrackID	Loc_X	Loc_Y	Nr-Node	OnBd_2	OnBd_3
10	RGtoDF11	23	20081	1	411	1	919416	3791123	1722376	1897921	0	0
10	RGtoDF11	23	20082	1	444	2	919409	3791115	1722376	1897921	0	0
10	RGtoDF11	23	20051	1	318	3	919190	3793730	75165200	2093287	0	0
10	RGtoDF11	23	20054	1	318	4	919221	3793668	75165200	2093287	0	0
10	RGtoDF11	23	20028	1	320	5	919506	3801799	1455209	1934645	0	0
10	RGtoDF11	23	20059	1	349	6	918388	3793196	1452487	1898424	0	0
10	RGtoDF11	23	20060	1	349	7	918408	3793146	1452487	1898424	0	0
10	RGtoDF11	23	20070	1	367	8	913749	3802969	1902795	1399546	0	0
10	RGtoDF11	23	20083	1	496	9	918388	3793186	1452484	1898424	0	0
10	RGtoDF11	23	20084	1	504	10	918366	3793163	1452484	1898424	0	0
10	RGtoDF11	23	20003	2	512	11	912117	3789938	2051798	75156220	0	0
10	RGtoDF11	23	20006	2	407	12	902301	3793259	75155672	75152413	0	0
10	RGtoDF11	23	20018	2	404	13	910272	3797681	1935627	26352666	0	0
10	RGtoDF11	23	20008	2	410	14	903632	3793321	75155672	26339791	0	0
10	RGtoDF11	23	20009	2	412	15	903624	3793304	75155672	26339791	0	0
10	RGtoDF11	23	20010	2	413	16	903648	3793277	75155672	26339791	0	0

Travel Delay Reports (16/904)

Rept Type	SrcDest	Rptr Type	NA	Juris ID	Time	Rept Cnt	Link_ID	Sev
11	RGtoDF12	29	20021	1	305	1	75153073	0
11	RGtoDF12	29	20022	1	306	2	75153073	0
11	RGtoDF12	29	20024	1	306	3	75153073	0
11	RGtoDF12	29	20025	1	305	4	75153073	0
11	RGtoDF12	29	20031	1	309	5	2107521	0
11	RGtoDF12	29	20032	1	309	6	2107521	0
11	RGtoDF12	29	20056	1	308	7	1898364	0
11	RGtoDF12	29	20078	2	422	8	1357174	0
11	RGtoDF12	29	20079	2	306	9	1357174	0
11	RGtoDF12	29	20080	2	429	10	1357174	0
11	RGtoDF12	29	20081	1	305	11	1897921	0
11	RGtoDF12	29	20082	1	305	12	1897921	0
11	RGtoDF12	29	20001	2	312	13	99879094	0
11	RGtoDF12	29	20003	2	313	14	75149088	0
11	RGtoDF12	29	20004	2	312	15	99879094	0
11	RGtoDF12	29	20006	2	314	16	75155676	0

Cluster Identification Reports (16/51,547)   *Note: Table truncated to show only first two Param fields*

Rept Type	SrcDest	Rptr Type	Time	Rept Cnt	TrackID	Clus_ID	Clus_Len	Cell_Ct	Cell 1 Params	Cell 2 Params...
12	RGtoDF13	0	454	1	0	0	150	4	1 920621.90092739649 3795723.2997216624 0 0	1 920366.59610897489 3795873.601731922 0 0
12	RGtoDF13	0	454	2	0	1	150	4	1 924349. 3797873. 0 0	1 923943.60098657245 3797921.8014203263 0 0
12	RGtoDF13	0	454	3	0	2	150	4	1 920024.33362365223 3801745.6726526767 0 0	1 919719.62558558153 3801844.1264935508 0 0
12	RGtoDF13	0	454	4	0	3	150	8	1 921078.01320520043 3802133.4887639959 0 0	1 921126. 3802270. 0 0
12	RGtoDF13	0	454	5	0	0	150	4	1 920621.90092739649 3795723.2997216624 0 0	1 920366.59610897489 3795873.601731922 0 0
12	RGtoDF13	0	454	6	0	1	150	4	1 924349. 3797873. 0 0	1 923943.60098657245 3797921.8014203263 0 0
12	RGtoDF13	0	454	7	0	2	150	4	1 920024.33362365223 3801745.6726526767 0 0	1 919719.62558558153 3801844.1264935508 0 0
12	RGtoDF13	0	454	8	0	3	150	8	1 921078.01320520043 3802133.4887639959 0 0	1 921126. 3802270. 0 0
12	RGtoDF13	0	454	9	0	0	150	4	1 920621.90092739649 3795723.2997216624 0 0	1 920366.59610897489 3795873.601731922 0 0
12	RGtoDF13	0	454	10	0	1	150	4	1 924349. 3797873. 0 0	1 923943.60098657245 3797921.8014203263 0 0
12	RGtoDF13	0	454	11	0	2	150	4	1 920024.33362365223 3801745.6726526767 0 0	1 919719.62558558153 3801844.1264935508 0 0
12	RGtoDF13	0	454	12	0	3	150	8	1 921078.01320520043 3802133.4887639959 0 0	1 921126. 3802270. 0 0
12	RGtoDF13	0	454	13	0	0	150	4	1 920621.90092739649 3795723.2997216624 0 0	1 920366.59610897489 3795873.601731922 0 0
12	RGtoDF13	0	454	14	0	1	150	4	1 924349. 3797873. 0 0	1 923943.60098657245 3797921.8014203263 0 0
12	RGtoDF13	0	454	15	0	2	150	4	1 920024.33362365223 3801745.6726526767 0 0	1 919719.62558558153 3801844.1264935508 0 0
12	RGtoDF13	0	454	16	0	3	150	8	1 921078.01320520043 3802133.4887639959 0 0	1 921126. 3802270. 0 0

Hospital Location Report

Rept Type	SrcDest	Hosp_Ct	Hosp 1 Params	Hosp 2 Params	Hosp 3 Params...
50	RGtoDP01	20	904274 3792798 2067823 60001 31	912075 3790018 2051856 60002 191	902851 3793357 1862612 60003 118

Hospital Location Report

Rept Type	SrcDest	Rept Cnt	Cns_Ct	Census Tract 1 Params	Census Tract 2 Params	Census Tract 3 Params
52	RGtoL201	1	206	6037104102 141 35 6 10 0	6037104201 108 28 5 9 0	6037104202 152 39 6 12 0

Hospital Location Report

Rept Type	SrcDest	Rept Cnt	TrackID	Hosp_Ct	Hosp 1 Params	Hosp 2 Params	Hosp 3 Params
55	RGtoL202	1	0	20	904274 3792798 2067823 60001 31	912075 3790018 2051856 60002 191	902851 3793357 1862612 60003 118

Hospital Location Report

Rept Type	SrcDest	Hosp_Ct	Hosp 1 Params	Hosp 2 Params	Hosp 3 Params...
42	RGtoMF01	20	904274 3792798 2067823 60001 31	912075 3790018 2051856 60002 191	902851 3793357 1862612 60003 118

Hospital Location Report

Rept Type	SrcDest	Hosp_Ct	Hosp 1 Params	Hosp 2 Params	Hosp 3 Params...
51	RGtoVZ01	20	904274 3792798 2067823 60001 31	912075 3790018 2051856 60002 191	902851 3793357 1862612 60003 118