AL/HR-TR-1994-0009

THE INTEGRATED MODEL DEVELOPMENT ENVIRONMENT

Bradley A. Lloyd, Capt, USAF

HUMAN RESOURCES DIRECTORATE
LOGISTICS RESEARCH DIVISION
2698 G STREET
WRIGHT-PATTERSON AIR FORCE BASE, OHIO 45433-7604

Patrick K. Clark

THE ANALYTIC SCIENCES CORPORATION (TASC)
2555 UNIVERSITY BOULEVARD
FAIRBORN, OHIO 45324

FEBRUARY 1994

FINAL TECHNICAL REPORT FOR PERIOD MARCH 1990 TO JUNE 1993

Approved for public release; distribution is unlimited

AIR FORCE SYSTEMS COMMAND
WRIGHT-PATTERSON AIR FORCE BASE, OHIO 45433-6573

NOTICES

When Government drawings, specifications, or other data are used for any purpose other than in connection with a definitely Government-related procurement, the United States Government incurs no responsibility or any obligation whatsoever. The fact that the Government may have formulated or in any way supplied the said drawings, specifications, or other data, is not to be regarded by implication, or otherwise in any manner construed, as licensing the holder, or any other person, corporation; or as conveying any rights or permission to manufacture, use, or sell any patented invention that may in any way be related thereto.

The Office of Public Affairs has reviewed this report, and it is releasable to the National Technical Information Service, where it will be available to the general public, including foreign nationals.

This report has been reviewed and is approved for publication.

Bradley A Lloyd

BRADLEY A. LLOYD, Capt
Project Director

BERTRAM W. CREAM, Chief
Logistics Research Division

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. AGENCY USE ONLY *(Leave blank)*	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	February 1994	Final - March 1990 to June 1993

4. TITLE AND SUBTITLE
The Integrated Model Development Environment

5. FUNDING NUMBERS
C - F33615-90-C-0007
PE - 62205F
PR - 1710
TA - 00
WU - 50

6. AUTHOR(S)
Bradley A. Lloyd, Capt, USAF
Patrick K. Clark

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
Armstrong Laboratory
Human Resources Directorate
Logistics Research Division
2698 G Street
Wright-Patterson AFB, OH 45433-7604

8. PERFORMING ORGANIZATION REPORT NUMBER

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)
The Analytic Sciences Corporation
2555 University Boulevard
Fairborn, Ohio 45324

10. SPONSORING/MONITORING AGENCY REPORT NUMBER
AL/HR-TR-1994-0009

11. SUPPLEMENTARY NOTES
Armstrong Laboratory Technical Monitor: Bradley A. Lloyd, AL/HRGO, (513) 255-2606

12a. DISTRIBUTION/AVAILABILITY STATEMENT
Approved for public release; distribution is unlimited

12b. DISTRIBUTION CODE

13. ABSTRACT *(Maximum 200 words)*
This report presents the results of the Integrated Model Development Environment (IMDE) exploratory development research project. The objective of the research was to demonstrate how object-oriented modeling and graphical user interface technologies could improve Air Force and DoD large-scale modeling efforts. This report documents the motivations for the research, describes the generic modeling tool that was developed to demonstrate the IMDE technologies, and discusses future laboratory research plans.

14. SUBJECT TERMS			15. NUMBER OF PAGES
airbase logistics	logistics decision support	simulation	36
IMDE	modeling		16. PRICE CODE
LCOM	object-oriented		
	operations research		

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
Unclassified	Unclassified	Unclassified	SAR

CONTENTS

FIGURES

Figure

PREFACE

This report documents research performed by The Analytic Sciences Corporation (TASC) for the Armstrong Laboratory, Human Resources Directorate, Logistics Research Division (AL/HRG) under contract number F33615-90-C-0007. The TASC effort, entitled "Integrated Model Development Environment (IMDE)," was designed to support the Productivity Improvements in Simulation Modeling (PRISM) project. The objective of PRISM is to enhance the Air Force's ability to perform simulation-based logistics capability assessment.

The background of PRISM is well documented. The first task, conducted as an in-house research project, investigated the limitations of current Air Force logistics modeling capabilities. This initial effort was entitled "Analysis of Air Force Logistics Capability Assessment Models" and is documented in AFHRL-TP-88-56. In general, the results of the study support the conclusion that current logistics modeling tools are hard to use, difficult to modify, and require an inordinate amount of data preparation.

The next PRISM research task investigated fledgling object-oriented simulation technology. This project was carried out by RAND Corporation. The final report is entitled "PSE: A CLOS-Based Persistent Simulation Environment With Prefetching Capabilities" and is documented in WD-4507-1-DARPA. Specifically, the research focused on how to provide persistent object-oriented modeling capabilities.

Although the in-house and RAND studies are considered successful stand-alone research projects, their underlying purpose was to lay the groundwork for IMDE. The objective of the IMDE research is to demonstrate how an object-oriented modeling approach along with state-of-the-art user interface technology could make large-scale logistics models less complicated, cheaper/easier to maintain, and more widely available. The approach taken to meet this objective was twofold. First, TASC developed a generic advanced modeling tool (i.e., the IMDE). Then, using the IMDE, they developed a small model of airbase logistics processes in order to test and demonstrate the IMDE concept.

Although recent demonstrations of the IMDE were successful, a comprehensive comparison of the modeling processes supported by the IMDE-based model relative to the processes supported by current models has not been conducted. The ongoing IMDE research is rectifying this situation by allowing United States Air Force (USAF) logistics analysts to thoroughly beta test and comment on the IMDE demonstration software. As part of this effort, AL/HRG is developing a large, complex model of F-16 airbase logistics operations to provide a basis for comparison.

In summary, this report documents the initial effort to develop the IMDE software, the reasoning behind it, and Fiscal Year 1994-95 (FY 94/95) research plans. In addition to this report, the demonstration IMDE software and its associated user and programmer manuals are available. (Requests for these products should be made through AL/HRG, Wright Patterson AFB, OH.) Pending the completion of the FY 94/95 effort in Dec 1994, these products, along with

results from the comparison between current and IMDE modeling processes, will be available to the general public through the Defense Technical Information Center.

The authors of this report wish to thank Mr Dick Cronk (ASC/XRE), Capt Pat Vincent (ASC/ALLH), Mr Herb Morgan (AFOTEC/SAL), and LtCol Terry Holtz (AFMEA/MEI) for their feedback and support throughout the IMDE project. Their contributions were essential to this successful research and development project.

The authors would also like to acknowledge the following former and current IMDE research team members: Dr Doug Popken (University of Dayton), Mr Kerris Renken (TASC), Mr Nick Stute (TASC), Ms Lynn Foreman (TASC), Mr Jeff Honious (TASC), and Mr Paul Barlow (TASC).

THE INTEGRATED MODEL DEVELOPMENT ENVIRONMENT

SUMMARY

The Integrated Model Development Environment (IMDE) is a set of software tools developed to address the shortcomings of Air Force capability assessment models such as the Logistics Composite Model (LCOM) and the Theater Simulation of Airbase Resources (TSAR). The shortcomings of these models were identified in an Armstrong Laboratory, Human Resources Directorate, Logistics Research Division (AL/HRG) study entitled "Analysis of Air Force Logistics Capability Assessment Models" and is documented in AFHRL-TP-88-56.

There were two phases in the IMDE effort. In the first phase, The Analytic Sciences Corporation (TASC) developed a state-of-the-art modeling tool (i.e., the IMDE) which utilizes advanced software technologies, including object-oriented programming, object-oriented data management, and extensive graphical user interfaces. In the second phase, TASC developed a small model of airbase logistics processes that support sortie production using the IMDE to demonstrate the IMDE's utility for Air Force logistics modeling studies. Although a small model was developed, the major focus of this effort was to develop the IMDE modeling software.

This paper identifies current United States Air Force (USAF) logistics modeling techniques and their associated limitations, describes the evolution of IMDE "state-of-the-art" modeling technologies, and discusses how the IMDE architecture (an integration of these technologies) addresses the current limitations. In addition, plans for future USAF logistics simulation research along with other potential areas for research are also discussed.

INTRODUCTION

USAF acquisition, test and evaluation, wing, and Major Command organizations, as well as contractors, use computer-based models of airbase logistics processes to study sortie production capability as a function of manpower resources, spare part availability, aircraft reliability, and numerous other factors. These models can be categorized as either stochastic or non-stochastic. One of the better known non-stochastic models is Dynametric, which is used by wing and Major Command supply analysts to investigate how spares' availability affects sortie production capability (Blazer & Zimmerman, 1991). One of the better known stochastic models is TSAR. A more powerful tool, TSAR allows a capability assessment analyst to study the full spectrum of variables affecting sortie production in a wartime environment (Emerson & Wegner, 1990).

Today, probably the most utilized and well-known stochastic model is LCOM. LCOM, like many of the other assessment models, was initially developed by Rand Corporation in the early 1970s (Boyle, 1990). Although it is a very powerful decision support tool, LCOM is limited by the fact that it utilizes software technology based on what was available in the early 1970s. On the basis of software technology, LCOM is very similar to other capability assessment models in that it is hard to use, difficult to modify, and requires an inordinate amount of data preparation (Popken et al., 1989).

Throughout the 1970s and early 1980s, these models (i.e., LCOM, TSAR, and Dynametric) represented the state of the art in Air Force logistics modeling and simulation analysis. However, in recent years, new vastly improved simulation technologies based on the object-oriented paradigm and extensive graphical modeling techniques have created an opportunity to revolutionize Air Force logistics modeling processes.

Within this setting, the objective of the IMDE project was to demonstrate how integrating object-oriented programming, object-oriented database management, and graphical user interface technologies could make USAF logistics models less complex and easier/cheaper to maintain and modify. This objective is being met by (1) developing a generic advanced modeling tool (i.e., the IMDE) and (2) developing a library of standard airbase logistics objects to demonstrate the IMDE's utility for airbase logistics modeling applications.

In the last three years, significant progress towards this objective has been made. A functional IMDE was first demonstrated at the IMDE Technical Status Review held in March 1993. In June 1993, a prototype of the airbase logistics model was demonstrated to LCOM analysts and laboratory researchers.

The purpose of this paper is to document the initial effort made under contract F33615-C-90-0007 to develop and test the IMDE. General information on current capability assessment techniques, their associated limitations, and advanced simulation technologies (utilized by the IMDE demonstration software) are provided to give the uninitiated reader background material. A detailed discussion of the IMDE modeling tool is also presented, followed by current plans for future research and potential high payoff research topics.

BACKGROUND

History of Logistics Modeling

Since the early 1970s, the Air Force has developed and utilized numerous discrete-event simulation (i.e., stochastic) modeling tools to help decision-makers (1) assess the capability of combat units, (2) determine and justify spare part and manpower requirements, and (3) test new systems. The most well-known tools are TSAR and LCOM. Both tools, developed in the late 1960s and early 1970s by RAND Corporation, model airbase logistics processes.

TSAR was developed to help decision-makers evaluate the impact of logistics process improvements on theater sortie production capability (Emerson & Wegner, 1990). It has traditionally been used by the Air Staff to perform special studies and to evaluate unit capability in the Air Force Capability Assessment Program (Dymond et al., 1987).

Although LCOM and TSAR both simulate airbase logistics processes in great detail, LCOM differs from TSAR in that it has primarily been used by the manpower community to perform manpower requirements studies. (Air Combat Command [ACC], Air Mobility Command [AMC], and Air Force Special Operations Command [AFSOC] are currently using LCOM to help determine and justify maintenance manpower requirements.) It has also been used extensively

for Logistics Support Analysis (LSA) and operational test and evaluation (OT&E) by the acquisition and test communities, respectively (Popken et al., 1989).

Limitations of Current Models

As stated previously, TSAR and LCOM, although very powerful, are complex and difficult to use. In part, this difficulty is caused by the complexity of the processes being modeled. Modeling all the maintenance functions at an airbase will, for the foreseeable future, be data intensive and complex; however, part of the complexity is also due to the software technologies these models utilize.

One feature current Air Force logistics modeling tools do not provide is the capability to graphically depict model logic and data. For example, redefining time sequences and relationships among simulation activities in TSAR, LCOM, and Dynametric requires "labor-intensive programming and data preparation" (Popken et al., 1989). Providing the capability to graphically depict and edit model logic and data would improve the simulation process by reducing the effort required to develop and verify data sets and their associated models.

Since Popken's study, some improvements have been made in this area. For example, manpower analysts at ACC, AMC, and AFSOC can graphically depict LCOM maintenance task networks. Unfortunately, changing these networks still requires editing column-sensitive LCOM ASCII input files. This situation is common to all Air Force logistics models. Although improvements have been made, most models still require extensive manipulation of input and output data.

In addition to being difficult to use, the models are also difficult to adapt to problems for which they were not originally intended. This fact is, in part, responsible for the proliferation of airbase logistics models over the last twenty years. Currently, the Air Force owns five different computer-based models that can be used to estimate sortie production capability as a function of airbase logistics constraints: All Mobile Tactical Air Forces (AMTAF), Weapon System Management Information System Supportability Assessment Model (WSMIS/SAM), Integrated Simulation Assessment of Airbase Capability (ISAAC), TSAR, and LCOM.

Even with the models currently available, Air Force logistics analysts still cannot investigate many complex relationships between sortie production and logistics constraints. Two examples of these limitations are listed below (Popken et al., 1989).

1. Current models cannot be used to evaluate how complex theater logistics support plans affect sortie production. For example, issues associated with theater command and control and lateral part supply cannot be studied.

2. Current models do not adequately address aircraft battle damage repair because they assume damaged aircraft equal attrited aircraft. This assumption causes estimated wartime maintenance manpower and spare parts requirements to be underestimated.

If current models were more flexible and easier to modify, many of these limitations could be overcome by modifying existing models, as opposed to developing new models.

State-of-the-Art Simulation Technologies

As previously stated, the purpose of the IMDE project is to demonstrate how advanced software technology can make USAF logistics models less complex and easier/cheaper to maintain and modify. The primary technologies exploited to date are object-oriented modeling (via ModSim), object-oriented database management (via the Versant™ Object-Oriented Database Management System), and extensive graphical programming capabilities. This section introduces these technologies along with their associated benefits so that their integration into the IMDE modeling tool can be better appreciated.

Object-Oriented Modeling

Object-oriented languages have been available for many years. Simula (Dahl & Nygaard, 1966), developed in the 1960s, is most frequently credited with being one of the first object-oriented simulation languages. Unfortunately, Simula, along with other early object-oriented languages (e.g., Smalltalk), never gained prominence within the Department of Defense (DoD) simulation community because of their poor run-time performance relative to other approaches. As a result, through the early 1980s, most DoD models were developed using generic programming languages like FORTRAN or FORTRAN-based simulation languages like Simscript.

Beginning in the mid-1980s, object-oriented simulation reemerged as a viable approach to discrete-event simulation. During this time, CACI Inc., under sponsorship from the U.S. Army Models Management Office, developed a new object-oriented simulation language called ModSim (Wallace & Herring, undated). Initially, ModSim was used by the U.S. Army Construction Engineering Research Laboratory (USACERL) to develop a large force structure tradeoff model (Herring et al., 1993). Since then, CACI evolved the ModSim used to support USACERL research into a commercially viable simulation language entitled, "Modsim II™."

Although developing an object-oriented simulation model can be complex, the basic concepts of object-oriented programming are simple. From a top-level perspective, any language that provides a data structure that encapsulates an entity's state and behaviors can be classified as "object-oriented." In object-oriented terminology, this type of data structure is called a **class**. Specific instances of a class are called **objects**. Each class and object have:

1. **attributes** which define the state of an object, and
2. **methods** which describe the object's behavior.

The distinction between classes, objects, attributes, and methods is illustrated by the following example. As shown in Figure 1, an airbase simulation may include three *aircraft* operating during a simulation run. Each aircraft is called an object. All three *aircraft* instances will have the same attributes and methods, which were derived from the generic *aircraft* class. The definition of the *aircraft* class could include attributes such as *mean_time_to_fail* and

aircraft_status. The definition of the *aircraft* class will also include the methods which define an *aircraft*'s behavior. For example, the *aircraft* class could have a *determine_repair_status* method.

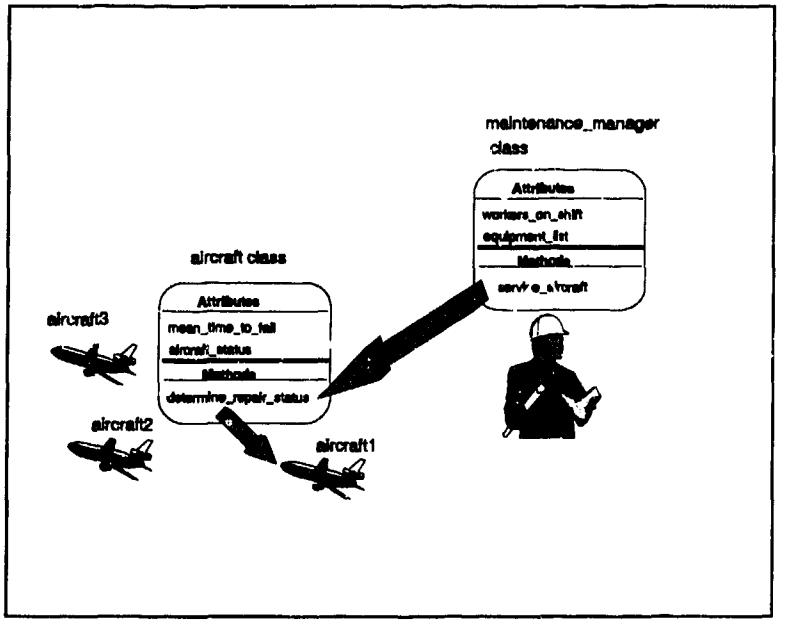

Figure 1
Message Sending Between Objects

Objects interact by sending **messages** to each other. Using the previous example, an *aircraft* object's *determine_repair_status* could be invoked by a *maintenance_manager* object. In ModSim, the command below would implement this type of message:

TELL aircraft1 TO determine_aircraft_status

This message might be sent by the *service_aircraft* method of *maintenance_manager1* to ascertain the type of service to provide.

As a result of the inherent modularity of this approach, the main benefit of object-oriented models is that they are easier to maintain, modify, and develop. Of course, the advantages of object-oriented programming do not come without a cost. Models based on traditional programming languages like FORTRAN and Simscript are more efficient. A common estimate of the run-time penalty associated with using an object-oriented approach is 10 to 15 percent; however, as computer hardware technology improves, run-time performance considerations will become less important relative to software modularity.

Object-Oriented Databases

Throughout the 1970s and 1980s, the relational database approach was the model for data-processing applications; however, relational databases never gained widespread application for engineering design applications (Elderstein, 1991). Beginning in the late 1980s, companies marketing fledgling object-oriented database management system software like Versant™ began to make inroads into the engineering design application market, primarily because object-oriented databases provide a richer environment for modeling data, providing interactivity, and supplying versioning and configuration control (Atwood, 1991).

At about the same time, the simulation research community began investigating the utility of object-oriented databases for simulation and modeling engineering applications. Early studies focused on providing persistence at simulation run-time. Many of these efforts (e.g., Cammarata & Burdoff, 1989; Herring et al., 1993; and Yu, 1992) have been successful. The purported benefit of run-time persistence for simulation is that larger and more complex simulations can be run more efficiently (Whitehurst & Herring, draft).

In addition to investigating how object-oriented databases can improve run-time execution of large models, many researchers have focused on how object-oriented databases can be used to support the overall simulation process. Object-oriented database support throughout the modeling process can significantly improve model maintenance, versioning, interconnectivity, and configuration control processes (Yu, 1992).

Graphical Modeling

The concept of graphical programming is not new to simulation. Event-based languages like SLAM™ and domain-specific modeling tools like COMNET II.5™ have offered graphical modeling capabilities for many years; however, recent advances in computer hardware and software technology have pushed graphical programming into the mainstream. It is generally accepted that these tools (with respect to simulation):

1. reduce the amount of time and effort required to develop models,
2. reduce the difficulty in verifying and validating a model, and
3. ease the model documentation task (Nickel, 1988).

Because of these benefits, it is not surprising that evolving object-oriented programming tools, like traditional modeling languages and tools, are also beginning to capitalize on the benefits of graphical programming.

INTEGRATED MODEL DEVELOPMENT ENVIRONMENT

As previously stated, the purpose of the IMDE project is to demonstrate how integrating object-oriented programming, object-oriented database management, and graphical user interface technologies can make USAF logistics models less complex and easier/cheaper to maintain and modify. This objective is being met by (1) developing an advanced modeling tool (i.e., the

IMDE) and (2) developing a library of standard airbase logistics objects to demonstrate the IMDE's utility for airbase logistics modeling applications. As of June 1993, the effort to develop the IMDE modeling tool is complete. Throughout this section, USAF airbase logistics examples are used to help explain IMDE functions; however, it is important to understand that IMDE is a **generic modeling tool**. It is just as applicable to manufacturing and communications network modeling as it is to USAF airbase logistics modeling.

Although the demonstration software developed under this contract is Government-owned, users must also have the commercial off-the-shelf software listed below in order to develop and run models using IMDE:

1. Modsim II™, version 1.9.2 or later; and
2. Versant™ OODBMS, version 2.1.3 or later.

Choices about what commercial software options to incorporate into IMDE were limited by the IMDE contract. For example, the contract stated that an object-oriented database management system must be utilized. As a result, three object-oriented database management systems were evaluated early in the contract: ONTOS™, Gemstone™, and Versant™. Although each product had merit, Versant™ was eventually chosen, primarily because it outperformed the two other databases in customized test queries that were performed early in 1990.

The IMDE contract also mandated that the modeling tool must support object-oriented model development. This fact drove the decision to use Modsim II™. Although, other object-oriented languages like C++ and Smalltalk were considered, Modsim was chosen because, unlike C++, it was specifically designed to support development of object-oriented simulation models. It was chosen over Smalltalk which, like Modsim, has direct support for object-oriented model development, because Modsim's run-time performance is superior to Smalltalk's.

Additionally, running the IMDE software currently requires a Sun Sparc workstation with the following minimum characteristics:

1. Sparc IPC™, Sparc IPX™, Sparc 2™, or a Sparc 10™;
2. 450 MB of disk space;
3. X11R4 or X11R5 Window Manager;
4. Sun OS™ 4.1.3; and
5. Sparcstation compatible color monitor.

Architecture

IMDE is a Computer-Aided Software Engineering (CASE) tool designed for use in the development, execution, and analysis of discrete-event simulation models. The IMDE architecture, depicted in Figure 2, was designed to address traditional problems associated with building, maintaining, and analyzing these types of models. The Model-Setup portion of the architecture supports the development of model parts (i.e., classes), the creation of models, and the specification of experiments. The Simulator supports the actual execution of an experiment and the Data Analysis portion supports analysis of raw data generated during an experiment.

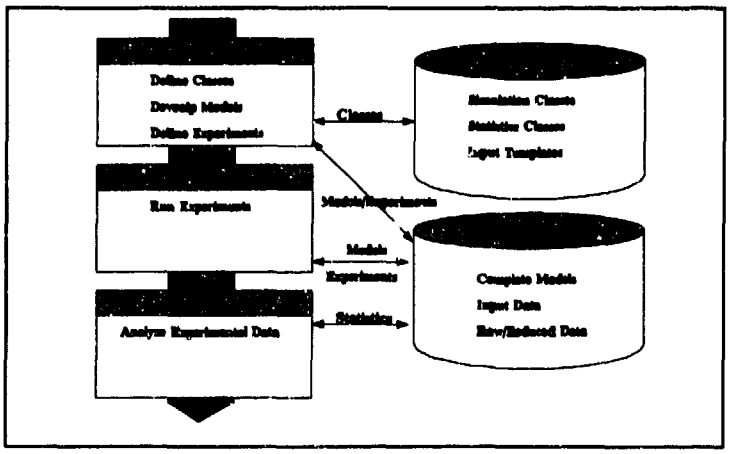

Figure 2
IMDE Architecture

Also important to the IMDE architecture are two object-oriented databases. The first database is called the "Premodel" database. The Premodel database stores and catalogues model parts (i.e., classes). The second database, the Project database, stores complete models and their associated input records, output records, and documentation. The following subsections describe the functions of the three different portions of the IMDE architecture and detail how they interact with these two databases.

Model Setup

Construction of models within IMDE is based on building up parts of models representing entities in a given simulation, then "hooking them together" to form a complete simulation model. These model parts will, in many cases, parallel the real-world items being modeled. In an airbase logistics simulation, aircraft, shops, test sets, runways, personnel, and spare parts might all be modeled. In IMDE terminology, these model parts are called **objects**, and templates for the design of these objects are called **classes**.

In IMDE, classes are defined in the Premodel Preprocessor which is in the Model Setup portion of the architecture. Users can define classes by writing Modsim II™ or, preferably, by using the Premodel Class Editor (see Fig. 3) to define the attributes and the Network Editor to define the methods. Users simply list the attributes of a class, then specify the attribute type from a list of available types.

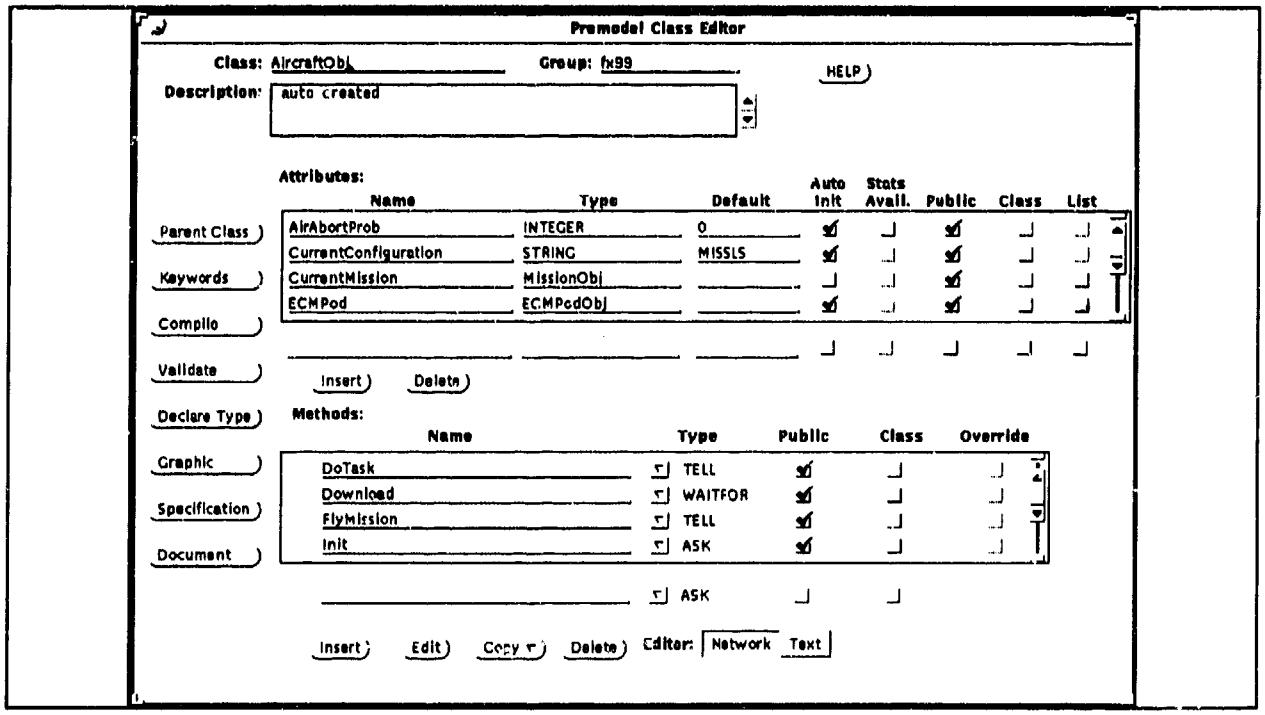

Figure 3
Premodel Class Editor

In addition to specifying an attribute's type, users may also indicate whether the attribute can be variable when included in a complete model, whether statistics can be collected on it, and whether it can be expanded as a list. These three options are associated with each attribute and will become important later when the Project Editor is discussed. For now, simply realize that once a class is defined it will have a list of attributes that can be parameterized, and a list of attributes for which statistics will be collected. In IMDE terminology, a list of inputs associated with a class is called a class input template (CIT), and a list of statistics is called a statistics set.

Each class may have zero, one, or more than one of each type of list. For example, the *AircraftObj* class shown in Figure 3 could have two different CITs and two different statistics sets associated with it, each with different combinations of the class' attributes contained in them. (Figures 4 and 5 display examples of a CIT and statistics set, respectively, for the *AircraftObj* class. Other CITs may have differing default values or minimum and maximum values. Other statistics sets may have more or fewer variables collected, and they may be collected in different ways (sampled, monitored, etc.).

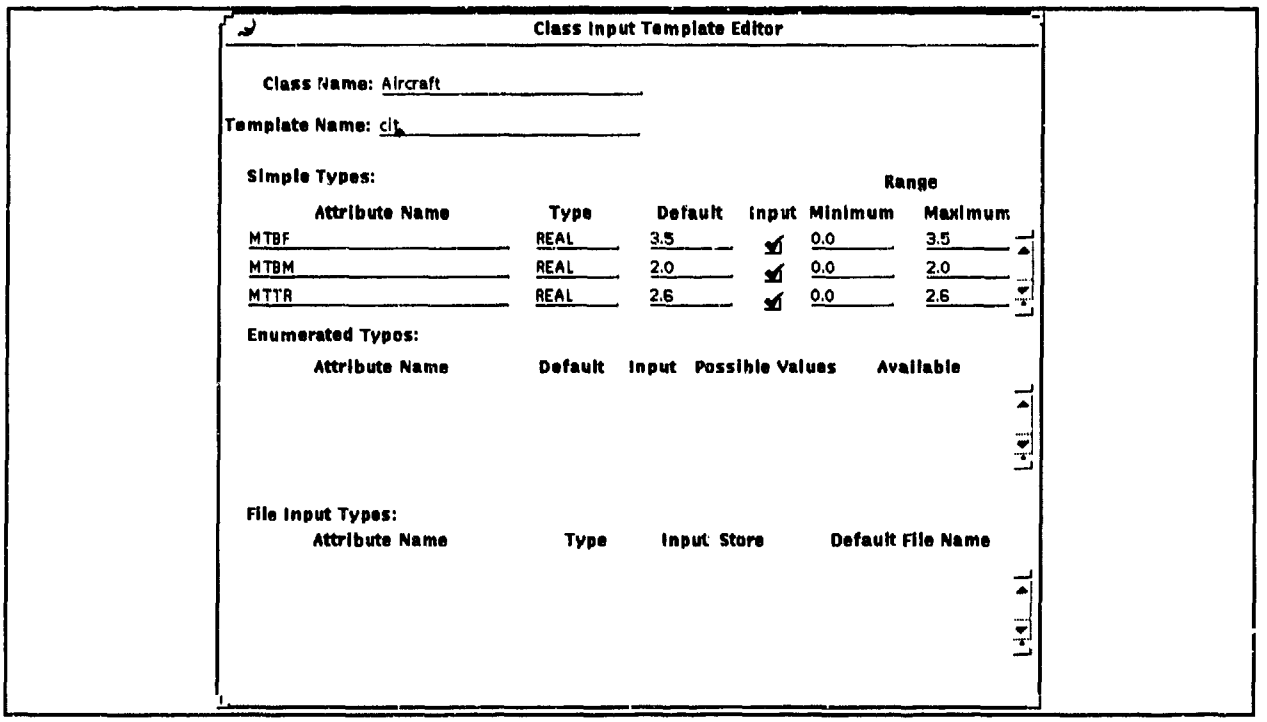

Figure 4
Class Input Template

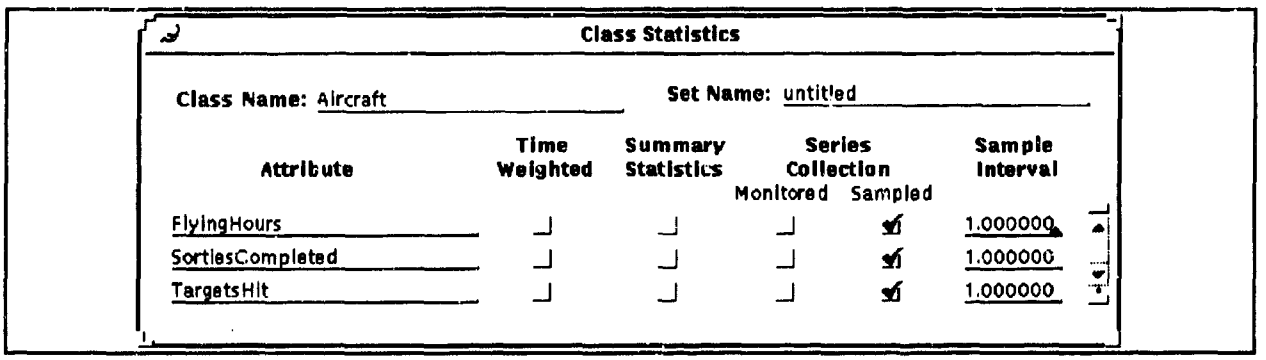

Figure 5
Statistics Set

Of course, to fully define a class, a user must also define its behaviors (which are described by methods). *DoTask*, *Download*, and *FlyMission* in Figure 3 are behaviors carried out by *AircraftObj* objects. The Network Editor (Figures 6 and 7) is used to graphically define behavior contained within each method.

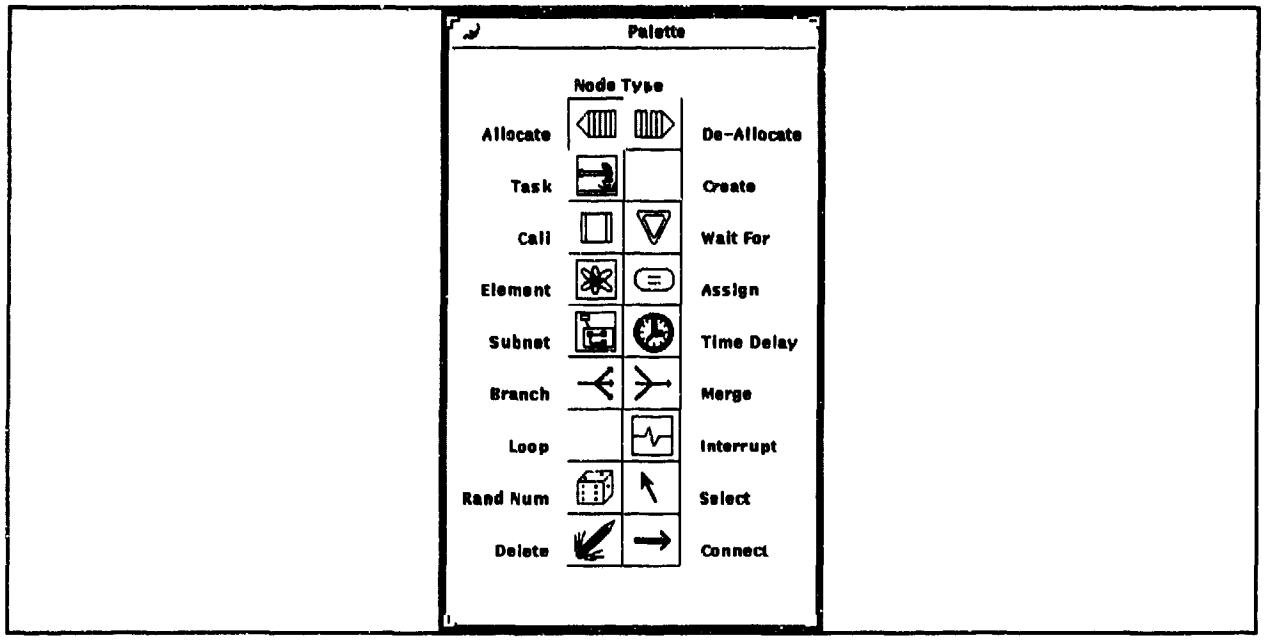

Figure 6
Network Editor Palette

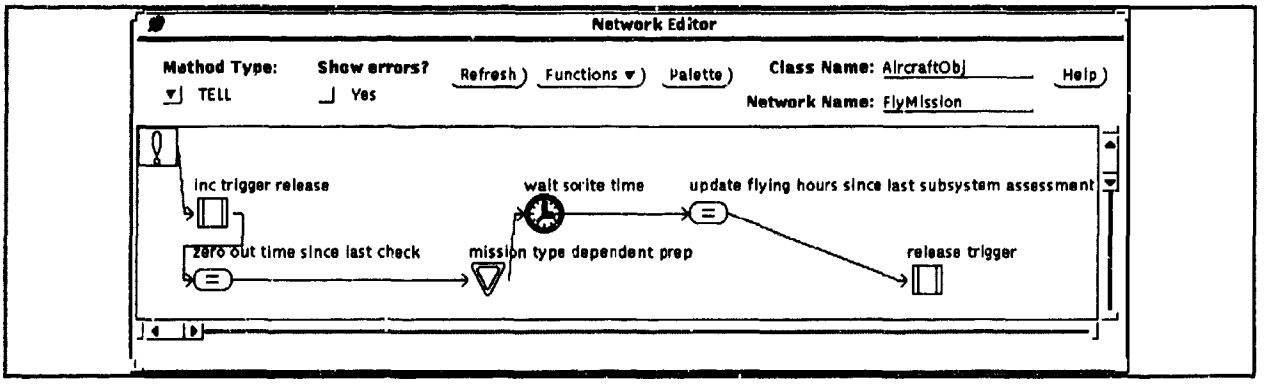

Figure 7
Network Editor

Although other simulation development tools allow process flow to be defined graphically, IMDE is unique in that the processes (represented by graphical networks) are part of the definition of a class. For example, *AircraftObj* may have several networks defining its associated behaviors including takeoff, flying a mission, landing, taxiing, and undergoing maintenance. These networks would be "bundled" within its description. In object-oriented terminology, the term for bundled is "encapsulated." Encapsulation is one of the primary object-oriented programming concepts which contributes to the main advantages of object-oriented programming, i.e., code reuse and maintainability.

Specifically, the Network Editor allows a modeler to chronologically define the sequence of actions an object takes for each different behavior. This flow chart description is constructed by "dropping" nodes representing simulation constructs onto a canvas and connecting them in the desired order. Each node type has a distinct attribute window that can be used to set appropriate parameters. For example, the time delay node requires the choice of distribution and of delay values. (The window corresponding to the time delay node is shown in Figure 8.) Other nodes are the allocate, de-allocate, call, wait for, element, assign, subnet, branch, merge, loop, and interrupt. Together this set of nodes represents all the basic structures necessary to construct complex simulation logic.

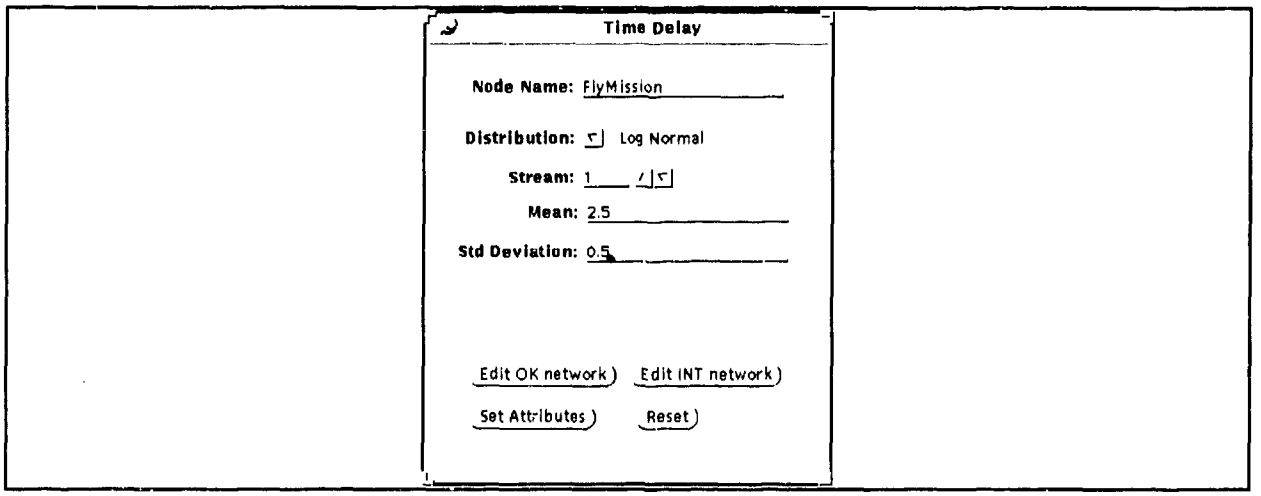

Figure 8
Time Delay Node

Once classes have been defined (using the Premodel Class Editor and the Network Editor), they are stored in the Premodel database for later use in a complete simulation model. To create a model, a class must be specified as the top-level object which will act as the "circuit board" of how the simulation will be filled in with classes stored in the Premodel database. In IMDE, these objects are called **frameworks**. In an airbase logistics simulation, the top-level object might be called *airbase* (for a single airbase simulation) or *scenario* (for multiple airbases).

Once the appropriate classes, including a framework, have been stored in the Premodel database, modelers use the Project Editor to specify all the objects that will populate the given simulation model. Figure 9 shows a portion of an airbase operations model (albeit a very simple one). Each box dropped on the construction canvas represents one or more simulation objects that will be part of the simulation. (Recall that "objects" are instances of classes.) In the model shown in Figure 9, there are some analogs of real-world objects such as airbases, aircraft, parts, and organizational units. There are also objects that are more abstract, like the *Scenario* object and the *MissionGenerator* object, which are used to provide a set of driving conditions for the simulation.

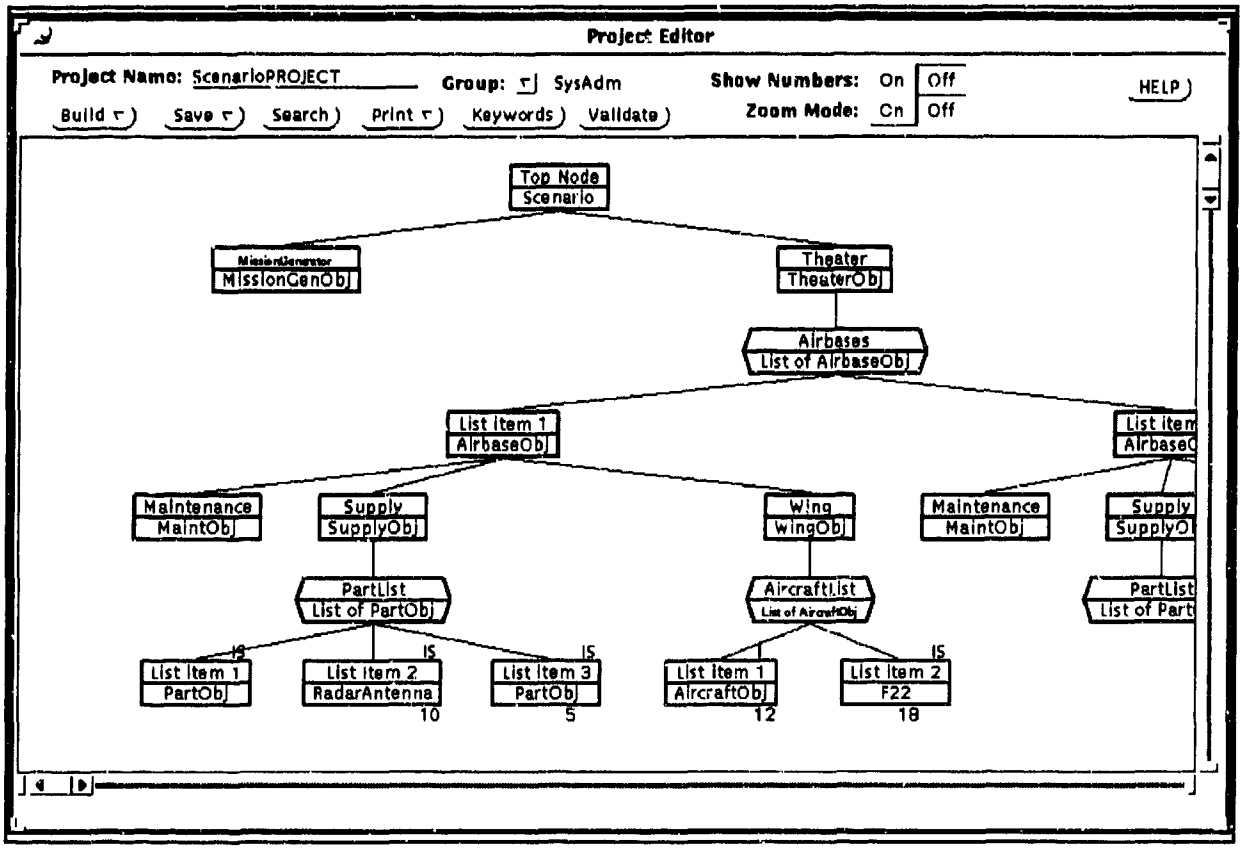

Figure 9
Project Editor

Project construction starts by selecting a simulation framework, which in Figure 9 is the *Scenario* class. (A class like *Scenario* is "turned into" a framework by selecting the IMDE standard "framework" class as a parent class using the parent function in Figure 3.) A framework class represents the top-level object in a simulation; it schedules the first events and the termination conditions of the simulation. Once a framework has been selected to start construction of an actual model, the Project Editor initially looks like Figure 10. This figure represents the basic structure associated with a *Scenario* object and can be "built-out" in a variety of ways to form complex models from a set of preconstructed classes.

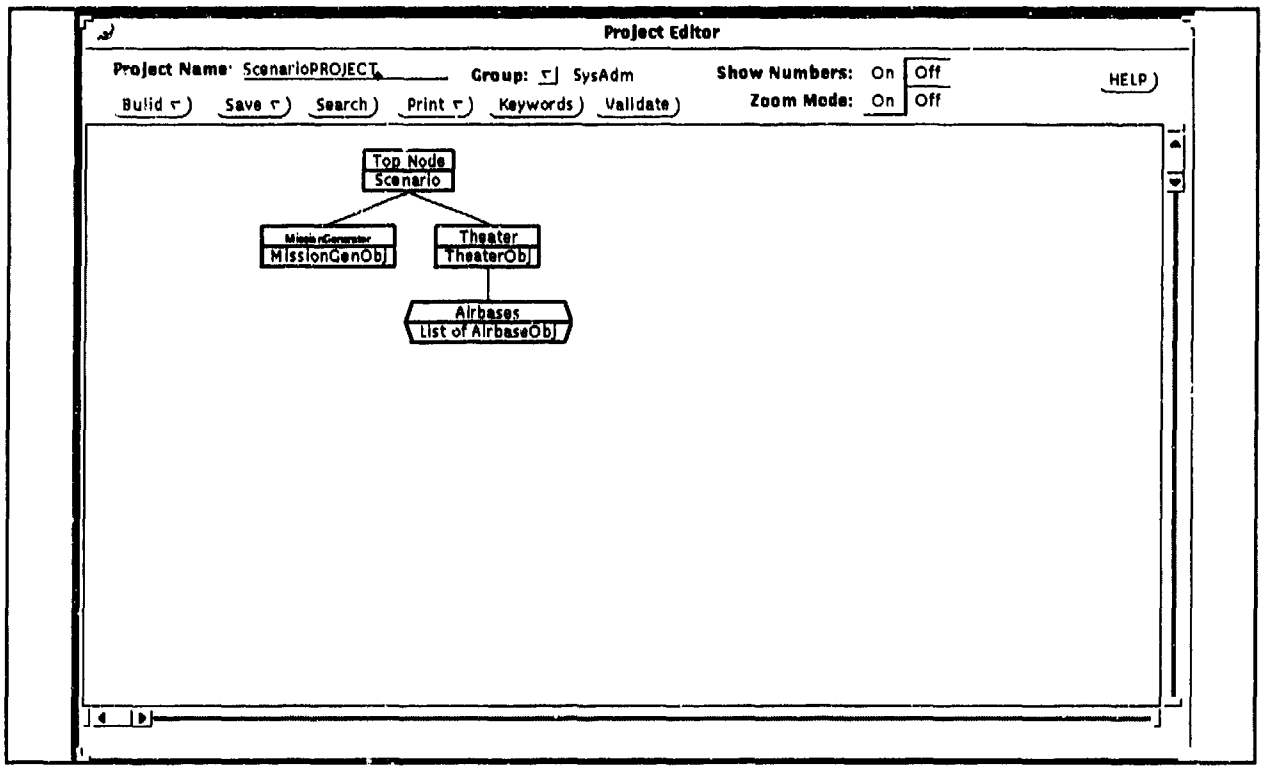

Figure 10
A Project Framework

There are four functions available for "building out" a framework. The first function, **list expansion**, is used to add objects to another object's list. In the example shown in Figure 10, it is used to add airbases to *Theater's List of AirbaseObj*. Specifying that a class has a list of objects (like *theater* has a list of *airbases*) involves checking the "List" box in the Premodel Class Editor (Figure 3). Expandable lists are indicated by irregular hexagons in the project diagram; all other objects are represented by rectangles. To arrive at the project shown in Figure 9, two other lists have been expanded (that we can see from the portion of the project shown in the diagram), including the *Supply* object's *PartList* and the *Wing* object's *AircraftList*. (Selecting a hexagon brings up a window that allows the user to easily add to or delete elements from a list.)

Using list expansion also allows a user to represent many of the same object types using one node. Figure 9 shows that the number of aircraft under the leftmost airbase is 30, consisting of 12 "generic" *AircraftObj* instances, and 18 more specific F22 instances. (If a node does not have a number below it, like the *AirbaseObj* which is "List Item 1" on *Scenario*'s list, then only one object is created at run time.) This facility is advantageous when objects read the same inputs and collect the same statistics. Without this facility, in the case of an 18-aircraft squadron, the user would have to drop 18 *AircraftObj* nodes on the Project Editor canvas.

Additional modification to the *Scenario* framework to develop the Project in Figure 9 involves a very powerful feature of Project construction called **substitution**. Substitution allows

a modeler constructing a project to snap in different classes than were originally specified. IMDE implements this capability by allowing any class in the **inheritance** tree of the original object to be substituted into the project. For example, Figure 9 shows *Wing* as having a list of generic *AircraftObj*s; however, using substitution, the user can substitute any class derived from the *AircraftObj* class. In this example, the *F22* class was derived from the *AircraftObj* class and then substituted in for a group of 18 aircraft belonging to the *Wing*. This is a very powerful feature because it allows modular evolution of projects. To look at the effectiveness of using a different type of aircraft, the modeler can simply derive a new class having the new characteristics and behavior, and snap it in place without having to change other objects that are unconcerned with the special behavior.

This approach to developing models represents a major advance in the construction of simulation models. In addition to **list expansion** and **substitution**, the major features facilitating this approach are **combination and input/statistics attachment.**

Combination allows different objects in the simulation to share access to other objects at run time. For example, an *aircraft* class may have *shelter* as an attribute. When two *aircraft* objects are dropped on the Project Editor canvas, each will have a *shelter* object attached below it. If a shelter can be used by two aircraft (determined by who defined the *shelter* class), the user can combine the two shelter objects into one. This combination will result in only one *shelter* being created at run time, and it will be used by both *aircraft* for whatever functions have been defined for aircraft-shelter interaction. Combination is useful in many simulation domains. Some examples are machines sharing the same conveyor, computers sharing the same bus, or workstations sharing the same test sets (combination is not illustrated in Figure 9).

Input/statistics attachment refers to the capability to provide different users with tailored lists of inputs that are available for variation and statistics that can be collected. As previously discussed, each class may also have several available CITs and statistics sets defined. For each object dropped on the canvas shown in Figure 9, users have the option of attaching one CIT and one statistics set from the object's available lists. If a CIT is attached, the attributes contained in the template will be available for variation in experiments carried out on the model. If a statistics set is attached, statistics will be collected on the attributes contained in the set when an experiment is carried out. The primary benefit of this facility is that different types of users can have input sets and output sets specifically tailored for their interests.

Once a complete model has been developed by defining classes using the Premodel Class Editor and the Network Editor, and linked together with the Project Editor using the facilities described above, users can define experiments. In IMDE, each Project eventually includes an executable MODSIM II™ simulation program, which can be used to run many experiments. This relationship is depicted in Figure 11. The experiment is composed of two parts: an Input Parameter Record (IPR) and an Experiment Control Record (ECR). The IPR is the set of parameters and Project structure information needed to run a single simulation iteration. The ECR is essentially the "outer loop" which determines how many iterations of the simulation will be run.

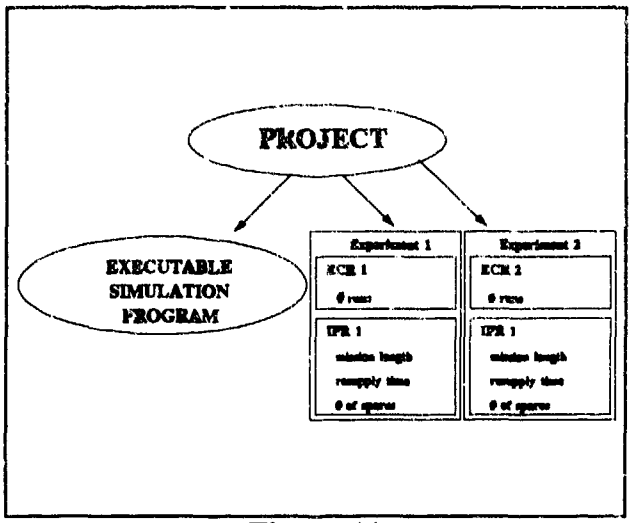

Figure 11
Project-Experiment Relationship

ECRs come in three types: simple treatment, hypothesis test, and sensitivity analysis. The simple treatment ECR specifies the number of runs for a given set of inputs. An example of the IMDE window used to define a simple treatment ECR is shown in Figure 12. The hypothesis test ECR allows the user to define a matrix of parameterizations to be run at one time (i.e., a full factorial experiment). A sensitivity analysis ECR allows a user to specify an independent variable to be incremented and decremented until a selected output parameter (i.e., dependent variable) exceeds a predetermined value. For example, mean time between failure for a part could be varied, at a user-defined increment, until aircraft availability reaches a specified level.

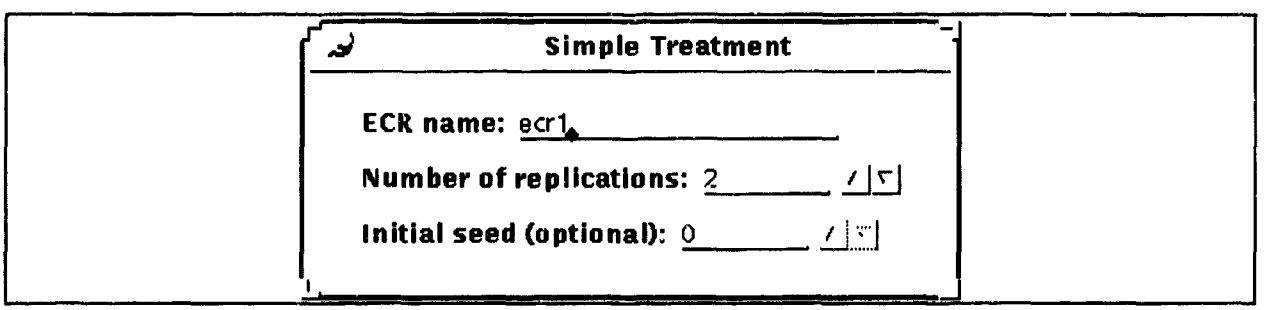

Figure 12
Experiment Control Record

Simulator & Data Analysis

The simulator portion of the IMDE architecture lets the user run an experiment by specifying the project and the specific experiment associated with that project. The raw data from the experiment is then stored in the Project database and tied to the relevant project and

input record. The output data can either be analyzed within IMDE using the integrated data analysis module or exported for use with other analysis packages.

The IMDE data analysis portion of the architecture provides calculation of mean, median, standard deviation, kurtosis, skew, minimum, and maximum values. It also supports creation of different graphs from run data, including time series, histograms, and scatterplots. Graphs can also be printed, stored, or exported in different formats for incorporation in presentation graphics packages. Run time graphics are also available through CACI Inc.'s Simgraphics II.5™.

User Levels

Another important aspect of the IMDE architecture is the concept of different user levels. When logging into IMDE, users are assigned specific user levels based on their simulation needs and expertise. As the IMDE main menu shows (Figure 13), four user levels are available: End User, Analyst, Developer, and Utility. The Utility level is equivalent to a system manager; a user at this level is responsible for assigning user levels and maintaining the IMDE software.

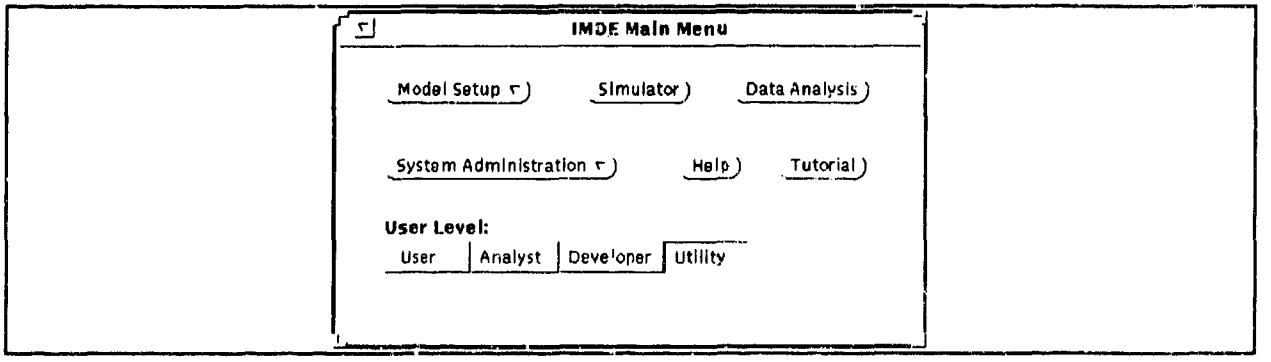

Figure 13
Integrated Model Development Environment Main Menu

The three other user levels correspond to varying degrees of programming, simulation theory, and domain knowledge. A user that is designated as a "Developer" can construct models from scratch by defining classes (i.e., model parts). As a result, to access the functions in the IMDE Premodel Preprocessor, a user must be a Developer. For example, the Developer is the only user level who can develop or modify classes using the Premodel Class Editor and Network Editor. Because they are developing simulation models, "Developers" must have in-depth knowledge of a simulation programming language. (Currently, the language must be Modsim II™.)

Conversely, "Analysts" do not require in-depth knowledge of a programming language. Instead, they must have a working knowledge of simulation theory and an in-depth knowledge of the domain area they are investigating. At the Analyst level, a user can "hook together"objects that were developed by Developer-level users to form new simulation projects within a specific domain area (e.g., airbase logistics).

As opposed to Analysts, "End Users" can only define experiments on models previously defined by Analysts. They are not allowed to hook together classes to form new models. Although End User access privileges seem limited, in practice they have the same capability that most Air Force logistics analysts have with current models; they can vary inputs with a previously defined model and analyze results.

Figure 14 shows how the three user levels interact in the simulation process. In general, Developers define classes which are stored in the Premodel database; Analysts hook classes together to form functional simulation models; and End Users run parameterizations on previously defined models. In practice, a simulation study may be performed by one person acting as a Developer, Analyst, and End User. Consequently, Developers have access to all the functions available to the Analyst and End User, and Analysts have access to the functions available to End Users.

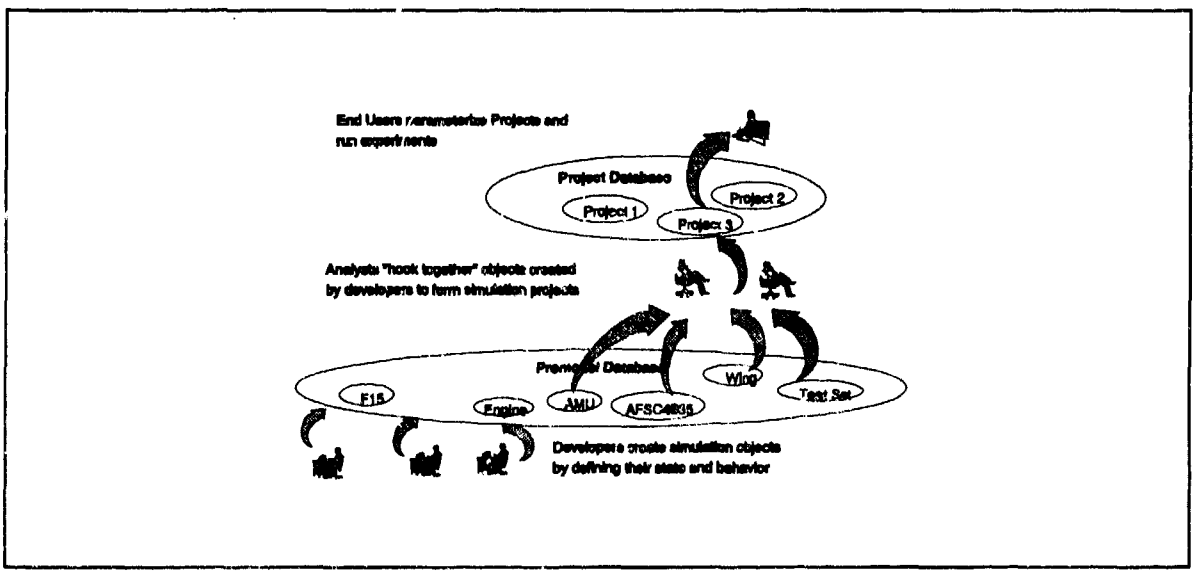

Figure 14
Integrated Model Development Environment Databases and User Levels

Advantages

USAF logistics modeling has been found to be difficult in the areas of network development, documentation, focusing capability, preprocessors and postprocessors, and configuration control (Popken et al., 1989). IMDE provides improvements in each of these areas.

Network Development

Network development refers to the capability to define the functionality of entities within models. In current USAF logistics models, redefining networks requires labor-intensive programming and data preparation (Popken et al., 1989). IMDE directly addresses the perceived difficulty in modifying networks by providing the Network Editor as a specific tool dedicated to

the interactive, graphical, and point-and-click description of the functionality of simulation entities. Networks developed using the Network Editor have the added benefit of providing standardized model documentation that non-programmers can look at to gain insight into the model.

Documentation

Poorly written documentation is also a common criticism associated with current USAF logistics modeling techniques. As a result, it is generally accepted that new users must be trained directly by the original model developer or one of the developer's disciples (Popken et al., 1989). In IMDE, modeling is done in a controlled development environment. Like a CASE tool, it provides for automated generation of functional flows (networks), structure diagrams (model diagrams), specifications, attached graphics, user annotation, and object-oriented source code. Although IMDE does not relieve a model developer of the entire documentation task, it does significantly reduce the effort required to produce quality documentation. It has the added benefit of enforcing a standardized method for developing and documenting source code (through the Premodel Class Editor, Network Editor, and Project Editor).

Focusing Capability

Focusing refers to the ability to allow users to track special areas of interest throughout the simulation without being deluged with tedious input formats, large input databases, and large, mostly unnecessary output files. According to Popken et al. (1989), current USAF logistics models do not have this capability. IMDE addresses this problem in several respects. First, CITs and Statistics Sets can be easily tailored by a modeler to meet the needs of a specific type of user. (Doing this with current USAF logistics models requires source code modification.) Secondly, development can be done hierarchically. Using the subnet node of the Network Editor, a modeler can show a simplistic block diagram at a high level, while describing the details of the process at a lower level. Figure 15 shows this concept. Finally, by virtue of IMDE's object-oriented approach, users can more easily focus on specific parts of model structure.

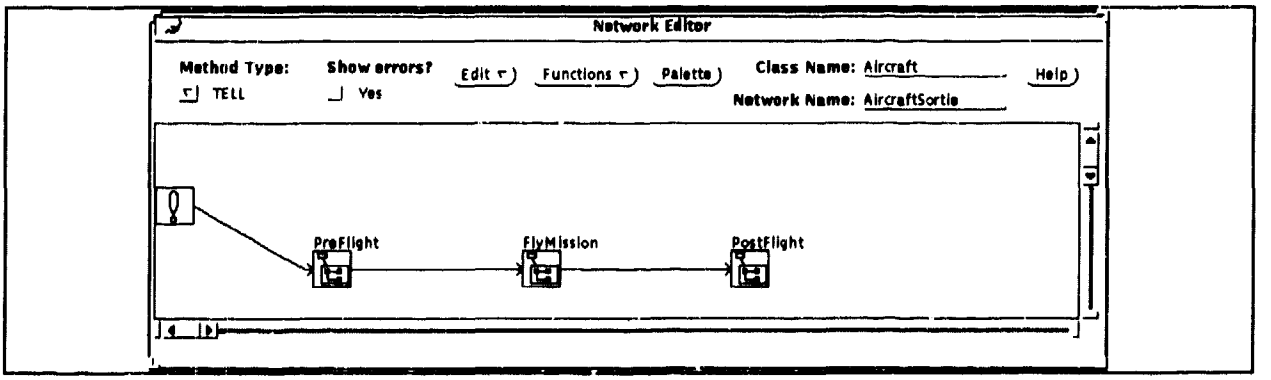

Figure 15
Use of Subnets for Hierarchical Modeling

Preprocessors and Postprocessors

In "modeling" lingo, "preprocessing" usually refers to actions taken to manipulate data to get it into the format required by a model. "Postprocessing" refers to actions taken to manipulate raw data generated from experiments performed using a model. According to Popken et al. (1989), current USAF logistics models have extensive preprocessing and postprocessing requirements. IMDE improves both of these phases of modeling. In the model setup phase, IMDE allows almost total graphical construction of a model from development of detailed algorithms to simple non-column sensitive parameter entry. This construction is done using the functions available in the Premodel Class Editor, the Network Editor, and the Project Editor. In terms of data analysis, since IMDE was constructed as an integrated tool set, the results of simulations are generated in a suitable format to be automatically loaded into the database. There they can be manipulated through a point-and-click interface to generate reduced statistics, charts and graphs, and formatted raw data for export to other statistical packages. Figure 16 shows a statistics summary window for a selected variable. Figure 17 shows a graph of run-time data for a selected variable.

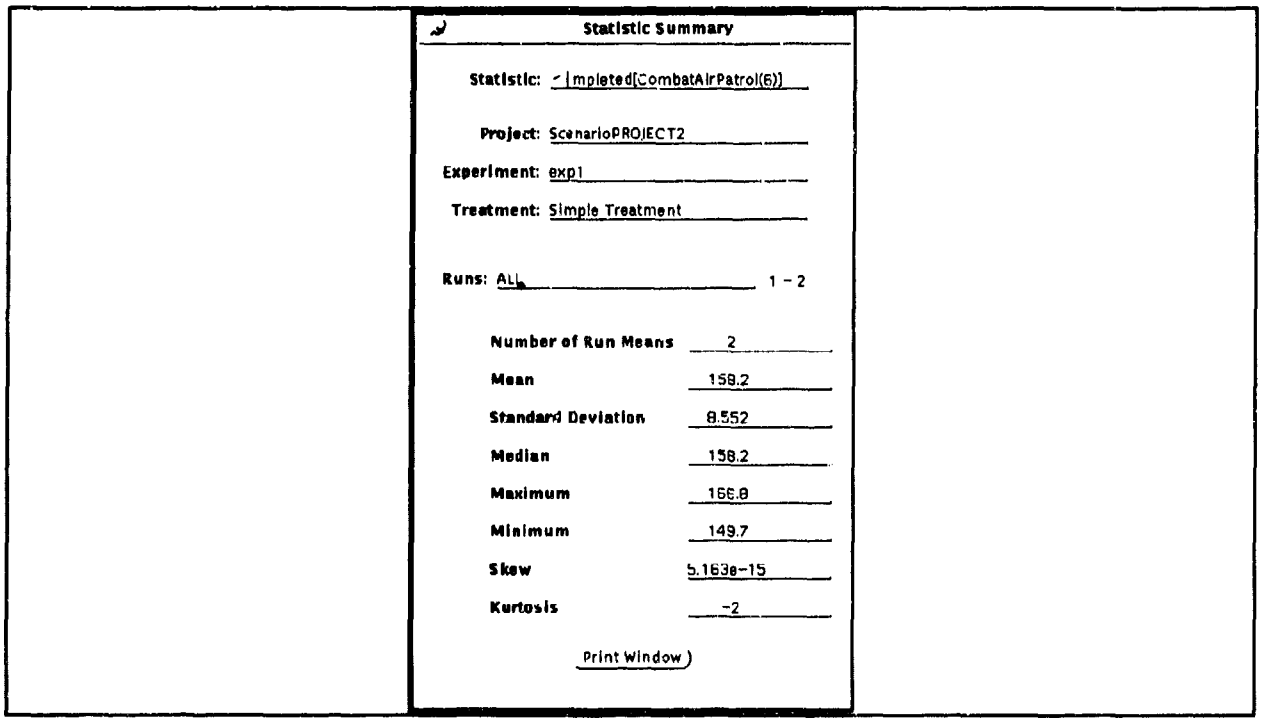

Figure 16
Statistics Summary Window

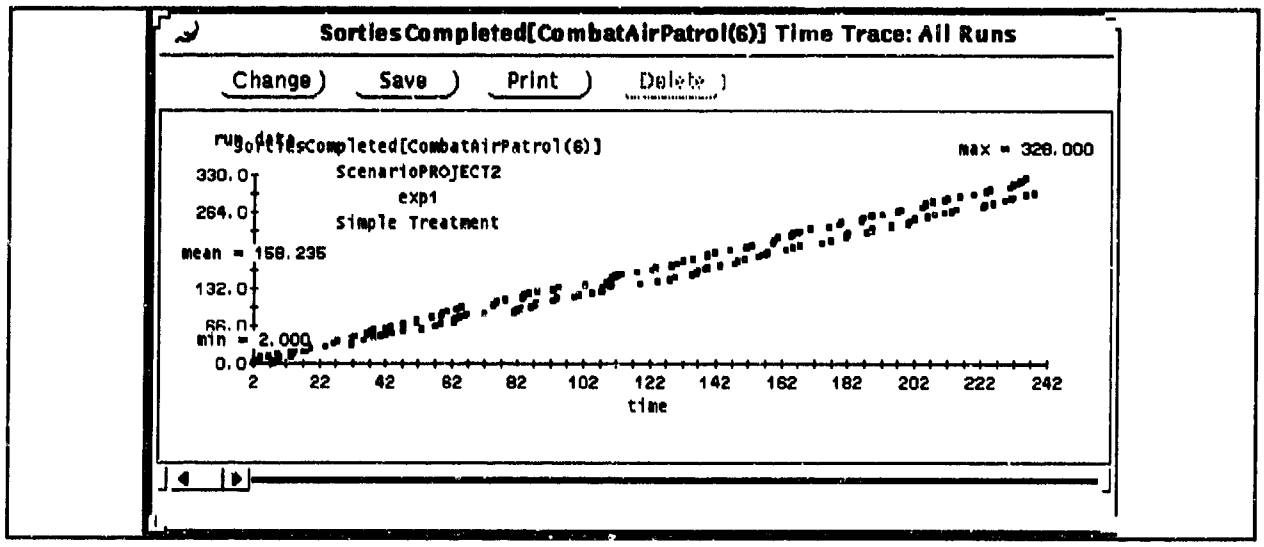

Figure 17
Time Trace of Run Data

Configuration Control

Maintaining configuration control of model versions, input sets, and output sets can often be a time-consuming and difficult task. Although Popken et al. (1989) identified configuration control as a deficiency of current USAF logistics models, the cited deficiency mostly focused on the need for a single Air Force Office of Primary Responsibility (OPR) that would maintain configuration control over an "official" version of each model referenced in the study. This deficiency is still a common problem. For example, there are currently two "accepted" versions of LCOM being circulated. Neither IMDE nor any other modeling tool will solve this problem.

Nonetheless, IMDE does have significant advantages over current modeling techniques in the area of configuration control. Often, in studies that last many months to many years, a tremendous effort is made to generate a large number of results. If these results cannot be duplicated, then the effort devoted to generating them will be wasted. To ensure that this situation does not occur, analysts frequently rely on significant off-line bookkeeping methods or complex on-line file-naming schemes. In IMDE, the analyst is freed from this cumbersome task. IMDE accomplishes this task by using the features of the object-oriented database management system (OODBMS) to build in configuration control. Each model's source code, object code, diagrams, documentation, input records, output records, and graphics are managed by the Project database. Additionally, each input record is linked to the model version it corresponds to as well as to its corresponding output record. These features allow Analysts to concentrate on defining and running experiments and Developers to concentrate on developing and modifying classes as opposed to worrying about the tedious bookkeeping tasks associated with their respective duties.

From a model developer's perspective, IMDE is also unique in that it provides a multi-user, multi-workgroup capability. This capability is very advantageous in the development and

use of large models. It is also necessary to achieve the object-oriented programming goals of reusability and maintainability.

PROGRESS AND PLANS

In general, two distinct groups who have expressed interest in the IMDE demonstration software: Air Force logistics analysts and simulation analysts from numerous other DoD and non-DoD domain areas. Initial evaluation of IMDE by both groups has been favorable; however, exposure to IMDE has been limited to one-hour demonstrations and half-day training sessions. The primary reason for the group's limited exposure to IMDE is that prior to June 1993, the IMDE demonstration software was not fully functional.

The planned Fiscal Year 1994-95 (FY 94/95) technical effort (carried out as a task under Air Force Materiel Command's [AFMC's] Supportability Investment Decision Analysis Center contract) will rectify this situation by allowing Air Force LCOM analysts to thoroughly test the IMDE demonstration software. In order to facilitate testing, a detailed airbase logistics model is being developed using LCOM F-16 input data as the basis for development. Using the model, LCOM analysts will gain an appreciation for and provide feedback on the utility of the IMDE approach to large-scale logistics modeling. (Recall that LCOM was one of the large-scale USAF logistics models studied in the initial review of USAF logistics models conducted by Popken, et al., 1989.)

Although full-scale testing is planned to begin in January 1994, some areas for enhancing IMDE were identified in demonstrations and training conducted during the technical effort documented in this report. One comment that LCOM analysts consistently made is that networks developed using the Network Editor were too complex, even for seemingly simple processes. One reason for this complexity is that modeling a task that requires more than one resource takes many nodes to represent. For example, a task that requires two different types of maintenance technicians, a piece of support equipment, and a line replaceable unit (LRU) currently requires twelve nodes to specify: four allocate nodes (to allocate each resource to the task), four time- delay nodes (to keep the resources allocated while the task is being carried out), and four de-allocate nodes (to de-allocate the resources once the task is finished).

The FY 94 technical effort will address this problem. One solution already being worked is to identify and incorporate additional Network Editor nodes that will allow higher-level simulation of object behavior. For example, a "task" node is currently being incorporated into the IMDE demonstration software. This new node will allow the modeler to specify a complete task description in a single node to include multiple resource requirements, selection of task duration, and subnetworks to be executed in the event the task is interrupted before it is completed. (In the example given in the previous paragraph, one task node will replace the twelve nodes currently required to define the task.)

Another feature users expressed a desire for throughout the IMDE effort was that IMDE be compatible with emerging Distributed Interactive Simulation (DIS) standards.[1] This was not surprising given the DoD simulation community's rising interest in DIS technology. Although there are no current plans to demonstrate a "DIS capability" within IMDE, doing so would not be difficult. To insert the capability into IMDE, a set of "DIS" classes would be created to communicate the state of the objects being simulated over the DIS network. (These classes would be transparent at the Analyst and End User levels. From the perspective of a Developer, the classes would serve as "parents" of all the objects in the simulation that need to communicate their state on the DIS network.) It would also be necessary to modify IMDE to run models in real time, as opposed to the current "fast-as-possible" analytic mode.

Potential users also commented that the IMDE demonstration software needs a top-down model design capability. (The current approach supported by IMDE is more of a bottom-up approach; that is, classes are first developed within the Premodel Preprocessor and then linked together to form a model.) A top-down design capability would allow a modeler to specify a model's objects and their associated attributes in the Project Editor. These objects would then be defined further in the Premodel Preprocessor. There are no plans to incorporate this facility into IMDE; however, incorporating the facility would make IMDE more in line with current thinking on object-oriented design (Booch, 1991), which suggests decomposing the modeling process into three steps: (1) identify the objects in the system being modeled, (2) define the interfaces between the objects in the system being modeled, and (3) implement the details of each object. Because of the modular design of IMDE, this capability would be easy to integrate into the current Project Editor.

Several groups visited during the course of the contract also expressed a strong interest in the potential of generating Ada or C++ simulation source code instead of Modsim II™. Reasons varied from wanting to comply with the requirement to use Ada to preferring one language over another. Although modifying the IMDE demonstration software to generate Ada or C++ as opposed to Modsim are not part of the current development plans, modifying IMDE to generate either C++ or Ada '9X would not be difficult.

CONCLUSIONS

Throughout the IMDE development contract, numerous Air Force and DoD study organizations were given demonstrations of the IMDE software and were interviewed about their modeling capabilities and requirements. Although most organizations have significantly improved their modeling capabilities since the original laboratory-sponsored study of USAF logistics models, the underlying software technology used in their models has not changed. Unfortunately, the simulation technology demonstrated in the IMDE software will do little to

[1] DIS standards are emerging standards for wide-area networked warfighting simulations. As the defense budget is downsized, DIS technology is expected to play a large role in operational training and effectiveness evaluations for concept definition, system acquisition, and operational test of new weapon systems. Simulators in this environment will range from high fidelity tank, fixed wing aircraft, and helicopter simulators to Semi-Automated Forces (SAFORs). SAFORs will be computerized simulations of forces (e.g., an entire tank battalion or fighter squadron) on a single workstation. Simulations developed and managed within IMDE would better meet the changing demands of DIS exercises. IMDE models are rapidly reconfigurable, well-documented, and under tight configuration control.

improve current models; however, these same technologies will significantly improve the usability and maintainability of future models.

With this in mind, future laboratory efforts in this area will focus on new Air Force applications of IMDE technology. In FY 94/95, a large airbase logistics simulation will be developed to facilitate a detailed comparison of the IMDE modeling approach to the current USAF manpower modeling process. Detailed testing will begin in January 1994 and continue through December 1994. Completing this comparison will satisfy the original goal of the IMDE research: demonstrating how object-oriented programming, object-oriented databases, and graphical user interfaces can enhance the productivity of USAF logistics analysts and formally mark the end of the IMDE exploratory development research.

REFERENCES

Atwood, T.M. (1991, February). The case for object-oriented databases. *IEEE Spectrum*, 44-47.

Booch, G. (1991). *Object-oriented design with applications*. Redwood City, CA: Benjamin/Cummings.

Blazer, D. & Zimmerman, D.L. (1991, Winter). The Air Force logistics assessment Architecture. *Air Force Journal of Logistics*, 12-16.

Boyle, E. (1990, July). *LCOM Explained* (AFHRL-TP-90-58, AD-A224497). Wright-Patterson AFB, OH: Logistics and Human Factors Division, Air Force Human Resources Laboratory.

Cammarata, S., & Burdorf, C. (1989, November). *User's Guide to the Persistent Simulation Environment* (WD-4684-DARPA). Santa Monica, CA: The Rand Corporation for the Defense Advanced Research Projects Agency.

Dahl, O., & Nygaard, K. (1966, September). *SIMULA: An ALGOL-based simulation language. Communications of the ACM*, 671-678.

Dymond, L.H., Hinds, B., Hopple, G., Gunkel, R., Schadle, W. & Bergeron, P. (1987). *Integrated and Enhanced War and Mobilization Planning System to Support AFIRMS Requirements for Sortie Tasking Data: Analysis of LCOM Capabilities* (final report). Synergy Incorporated for HQ USAF/XOOIM.

Elderstein, H. (1991, November). Relational vs. object-oriented. *DBMS*, 68-79.

Emerson, D.E., & Wegner, L.H. (1990, September). *TSAR User's Manual - A Program for Assessing the Effects of Conventional and Chemical Attacks on Sortie Generation: Vol. I, Program Features, Logic, and Interactions* (N-3011-AF). Santa Monica, CA: The Rand Corporation for the Long Range Planning and Doctrine Division, Directorate of Plans, HQ USAF.

Herring, C., Kalathil, B., & Teo, J. (1993, January). *Research in Persistent Simulation: Development of the Persistent ModSim Object-Oriented Programming Language* (Draft USACERL Interim Report F-93/XX). Champaign, IL: Construction Engineering Research Laboratory, U.S. Army Corps of Engineers.

Nickel, R.H. (1988, October). Report of the analysis working group. In W.B. LaBerge (Ed.), *MORS Workshop on Simulation Technology 1997* (SIMTECH 1997).

Popken, D.A., Cooke, G., & Dickinson, C. (1989). *Analysis of Air Force Logistics Capability Assessment Models* (AFHRL-TP-88-56). Wright-Patterson AFB, OH: Logistics and Human Factors Division, Air Force Human Resources Laboratory.

Wallace, J., & Herring, C. (undated). *Introduction to Frameworks and Programming in the ModSim/Modlog Object-Oriented Simulation Language* (draft). Champaign, IL: Construction Engineering Research Laboratory, U.S. Army Corps of Engineers.

Whitehurst, R., & Herring, C. *Adding Persistence to an Object-Oriented Simulation Language* (draft). Champaign, IL: Construction Engineering Research Laboratory, U.S. Army Corps of Engineers.

Yu, Matthew K., (1992, March). *System modeling via advanced reasoning techniques.* (unpublished paper). St Louis, MO: McDonnel Aircraft Company, Department of Logistics Technology Development.

ACRONYMS

ACC	Air Combat Command
AFMC	Air Force Material Command
AFSOC	Air Force Special Operations Command
AL/HRG	Armstrong Laboratory, Logistics Research Division
AMC	Air Mobility Command
AMTAF	All Mobile Tactical Air Forces
CASE	Computer-Aided Software Engineering
CIT	Class Input Template
DIS	Distributed Interactive Simulation
DoD	Department of Defense
ECR	Experiment Control Record
FY	Fiscal Year
IMDE	Integrated Model Development Environment
IPR	Input Parameter Record
ISAAC	Integrated Simulation Assessment of Airbase Capability
LCOM	Logistics Composite Model
LRU	Line Replaceable Unit
LSA	Logistics Support Analysis
OODBMS	Object-Oriented Database Management System
OPR	Office of Primary Responsibility
OT&E	Operational Test and Evaluation
PRISM	Productivity Improvements in Simulation Modeling
SAFORs	Semi-Automated Forces
TASC	The Analytic Sciences Corporation
TSAR	Theater Simulation of Airbase Resources
USACERL	United States Army Construction Engineering Research Laboratory
USAF	United States Air Force
WSMIS/SAM	Weapon System Management Information System Supportability Assesssment Model

U.S.G.P.O.:1994-550-057/81066