

REPORT DOCUMENTATION PAGE**Form Approved**
OMB No. 0704-0188

Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instructions, searching data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden to Washington Headquarters Service, Directorate for Information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302, and to the Office of Management and Budget, Paperwork Reduction Project (0704-0188) Washington, DC 20503.

PLEASE DO NOT RETURN YOUR FORM TO THE ABOVE ADDRESS.

1. REPORT DATE (DD-MM-YYYY) 4 April 20		2. REPORT TYPE Final Technical Report		3. DATES COVERED (From - To) 1 Oct 2005 to 30 Sept 2008	
4. TITLE AND SUBTITLE Design of Adaptive Organizations for Effects Based Operations				5a. CONTRACT NUMBER	
				5b. GRANT NUMBER N00014-06-1-0081	
				5c. PROGRAM ELEMENT NUMBER	
6. AUTHOR(S) Levis, Alexander H. Wagenhals, Lee W. Liles, Stewart W.				5d. PROJECT NUMBER	
				5e. TASK NUMBER	
				5f. WORK UNIT NUMBER	
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) System Architectures Laboratory Dept. of Electrical and Computer Engineering The Volgenau School of Information Technology and Engineering George Mason University, Fairfax, VA 22030				8. PERFORMING ORGANIZATION REPORT NUMBER SAL/FR-09-01	
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES) OFFICE OF NAVAL RESEARCH 875 N. RANDOLPH ST. ONE LIBERTY CENTER ARLINGTON VA 22203-1995				10. SPONSOR/MONITOR'S ACRONYM(S)	
				11. SPONSORING/MONITORING AGENCY REPORT NUMBER	
12. DISTRIBUTION AVAILABILITY STATEMENT Unclassified Unlimited					
13. SUPPLEMENTARY NOTES					
14. ABSTRACT Results from several basic research tasks are reported. In order to respond to operations other than conventional major theater war, organizational designs that enable adaptation to to changing situations are needed. In order to enable such adaptation, the supporting physical system of systems must also be adaptable and agile. Measures for assessing the adaptivity and agility of systems of systems that support the command and control functions were developed and a methodology based on executable models of architectures was designed and applied to a naval example. A second thread re-examined organization design algorithms to enhance them so that they can address cultural differences in coalition operations. A third thread used Timed Influence nets to develop and evaluate Courses of Action that would result in achieving desired effects. Model Driven Experimentaion was used to develop and assess measures for evaluating the extent of to which effectst based planning resulted in achieving the desired effects.					
15. SUBJECT TERMS Organization Design, Adaptive Organizations, System of Systems, Architecture Framework, Timed Influence Nets, Colored Petri Nets					
16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT SAR	18. NUMBER OF PAGES x + 224	19a. NAME OF RESPONSIBLE PERSON Alexander H. Levis
a. REPORT Unclassified	b. ABSTRACT Unclassified	c. THIS PAGE Unclassified			19b. TELEPHONE NUMBER (Include area code) 703 993 1619



THE VOLGENAU SCHOOL OF INFORMATION
TECHNOLOGY AND ENGINEERING
DEPARTMENT OF ELECTRICAL AND COMPUTER
ENGINEERING
SYSTEM ARCHITECTURES LABORATORY



DESIGN OF ADAPTIVE ORGANIZATIONS FOR EFFECTS BASED OPERATIONS

FINAL TECHNICAL REPORT

Contract No. N00014-06-1-0081

for the period

1 October 2005 to 30 September 2008

20090408190

Submitted to:
Office of Naval Research
Attn: Mr. Gerald Malecki
Code 342
825 N. Randolph Street, Suite 1425
Arlington, VA 22203-1995

Prepared by:

Alexander H. Levis
Principal Investigator

April 4, 2009

TABLE OF CONTENTS

List of Figures	v
List of Tables	ix
1. Executive Summary	1
1.1 Objectives	1
1.2 Tasks	1
2. Computational Models of Multi-National Organizations	5
2.1 Introduction	5
2.2 Decision Maker Model and Organization Design	2
2.3 Modeling Cultural Attributes	7
2.4 Coalition Modeling Using CAESAR III	9
2.5 Conclusions	15
2.6 References	15
3. Using Architectures to Support Experimentation	17
3.1 Introduction	17
3.2 Architecture-Based Model Driven Experimentation	18
3.3 Application of Model Driven Experimentation	20
3.4 Conclusions	27
3.5 References	27
4. Course of Action Analysis in a Cultural Landscape Using Influence Nets	29
4.1 Introduction	29
4.2 Timed Influence Nets	30
4.3 Case Study	34
4.4 Observations and Comments	43
4.5 References	44
5. Service Oriented Architectures, the DoD Architecture Framework 1.5, and Executable Architectures	45
5.1 Introduction	45
5.2 Background and Challenges	46
5.3 Process Descriptions	50
5.4 Case Study	54
5.5 Comments and Conclusion	85
5.6 References	86
6. Analysis and Evaluation of System of Systems Architectures	87
6.1 Introduction	87
6.2 Related Work	90
6.3 Assessment Measures	107
6.4 Methodology	135

6.5	Transformation	153
6.6	Conclusion	161
6.7	References	164
7.	Case Study: Expeditionary Strike Group	171
7.1	Introduction	171
7.2	Scenario	171
7.3	Operational Architecture	172
7.4	System Architectures	178
7.5	System of System Instance Architecture	190
7.6	Assessment Measure Calculation for P2P SOSI	208
7.7	Case Study Results	210
7.8	P2P Results	211
7.9	CS Results	213
7.10	SOA Results	217
7.11	Overall Results	218
7.12	Conclusions	221

LIST OF FIGURES

Fig. 2.1.	Model of the Five Stage Decision Maker	6
Fig. 2.2.	One Sided Interaction Between DM_i and DM_j	6
Fig. 2.3.	Flowchart for Culturally Constrained Solution	9
Fig. 2.4.	Level-I Organizational Block Diagram	11
Fig. 2.5.	Solution Space for Level-I Organization Design as Seen in CAESAR III	11
Fig. 2.6.	MINO of Level-I Design	11
Fig. 2.7.	MAXO of Level-I Design	12
Fig. 2.8.	Block Diagram of ACE	12
Fig. 2.9.	Block Diagram of GCE	12
Fig. 2.10.	Block Diagram for CSSE	13
Fig. 2.11.	GCE Structure Selected for US	14
Fig. 2.12.	GCE Structure Selected for Country A	14
Fig. 2.13.	GCE Structure Selected for Country B	14
Fig. 2.14.	Percent of Tasks Unserved for Coalition Options	15
Fig. 3.1.	Architecture Based MDE Process	19
Fig. 3.2.	External System Diagram	23
Fig. 3.3.	A0 Page	23
Fig. 3.4.	Colored Petri Net Executable Model, A0 Page	24
Fig. 4.1.	An Example Timed Influence Net (TIN)	31
Fig. 4.2.	Probability profile for Node C	32
Fig. 4.3.	Complete TIN Model	36
Fig. 4.4.	Static Quantitative COA Comparison	38
Fig. 4.5.	Dynamic Temporal Analysis Input	40
Fig. 4.6.	Probability profiles of Scenario (COA) of Fig. 4.5.	41
Fig. 4.7.	Comparison of the Effect of Different Scenarios	42
Fig. 5.1.	Architecture Design and Evaluation	49
Fig. 5.2.	DoDAF Architecture Design Process	51
Fig. 5.3.	Architecture Evaluation with Executable Model	54
Fig. 5.4.	OV-1, Operational Concept Graphic	56
Fig. 5.5.	OV-4 Organizational Relationship Diagram	57
Fig. 5.6.	Initial Sketch of Systems Nodes and System	58
Fig. 5.7.	Basic UML Activity Diagram (OV-5)	58
Fig. 5.8.	Activity Diagram OV-5 with Swim Lanes	59
Fig. 5.9.	UML Sequence Diagram used for OV-6c	60
Fig. 5.10.	UML Communications Diagram	61
Fig. 5.11.	State machine Diagram for the Sense Node (OV-6b)	61
Fig. 5.12.	Class Diagram	62
Fig. 5.13.	OV-3 Operational Information Exchange Matrix	63
Fig. 5.14.	OV-5 Based on IDEF0	63
Fig. 5.15.	OV-7 Based on IDEF1x	64
Fig. 5.16.	Mapping form Operational Activities to Systems and their Functions	65

Fig. 5.17.	SV-5a, Operational Activity to System Function Traceability Matrix	66
Fig. 5.18.	SV-5b, Operational Activity to Systems Traceability Matrix	67
Fig. 5.19.	SV-5c, Operational Activity to Services Traceability Matrix	68
Fig. 5.20.	SV-4a, Systems Functionality Description	69
Fig. 5.21.	SV-10c, Services Event Trace Description	70
Fig. 5.22.	SV-4b, Service Specification	70
Fig. 5.23.	Systems Communication Diagram	71
Fig. 5.24.	Component Diagram	71
Fig. 5.25.	Component Diagram (with Services)	72
Fig. 5.26.	Component Diagram (with Interfaces)	73
Fig. 5.27.	SV-11, Physical Schema	74
Fig. 5.28.	SV-1, Systems Interface Description	74
Fig. 5.29.	SV-1, Services Interface Description	75
Fig. 5.30.	SV-6, Systems/Services Data Exchange matrix	76
Fig. 5.31.	SV-2, Systems/Services Communications Description	77
Fig. 5.32.	SV-3, Systems to Systems Matrix	78
Fig. 5.33.	SV-7, Systems Performance Parameters Matrix	78
Fig. 5.34.	SV-8, Systems/Services Evolution Description	79
Fig. 5.35.	SV-9, Systems/Services Technology Forecast	79
Fig. 5.36.	CPN Model of the ATIS Operational View	80
Fig. 5.37.	Parameter Locus	81
Fig. 5.38.	Simulation Run	82
Fig. 5.39.	Performance Locus for Simulation Run	83
Fig. 5.40.	Projection of the Performance Locus onto the Leaks/ Average Response Time Plane	83
Fig. 5.41.	Requirements Locus Superimposed on Performance Locus	84
Fig. 5.42.	Projection of Requirements Locus onto the Parameter Locus	84
Fig. 6.1.	System of Systems [Brown, 2005]	89
Fig. 6.2.	Graphical View of DeLaurentis Taxonomy	93
Fig. 6.3.	Department of Defense (DoD) Architecture Registry System (DARS) Role	98
Fig. 6.4.	Role of UML Common Core [OMG, 2007a]	100
Fig. 6.5.	UML-MOF Meta-Levels [OMG, 2007b]	100
Fig. 6.6.	Four Layer Meta-Model Hierarchy [OMG, 2007b]	101
Fig. 6.7.	Model Driven Architecture Fundamental Concept [OMG, 2007b]	103
Fig. 6.8.	SOSI Taxonomy	109
Fig. 6.9.	System of Systems (SOS) Boundary	111
Fig. 6.10.	SOS Taxonomy Relationships Venn Diagram	113
Fig. 6.11.	Simple Information Flow Path	116
Fig. 6.12.	Cohesion Example	117
Fig. 6.13.	Node with Loops	117
Fig. 6.14.	Cobb-Douglas Production Function	121
Fig. 6.15.	Adaptability Plot	122
Fig. 6.16.	Adaptability Contour	122
Fig. 6.17.	Contour Calculation	123
Fig. 6.18.	Effects of Elasticity Constant on Adaptability	124

Fig 6.19.	Relationship of Cohesion and Coupling to Adaptability	125
Fig 6.20.	Agility Plot	127
Fig 6.21.	Working Example for Adaptability Calculations with Nodes	128
Fig 6.22.	CPN for n1,1	129
Fig 6.23.	Colored Petri Net (CPN) for n1,2	129
Fig 6.24.	Colored Petri Net (CPN) for n1,3	130
Fig 6.25.	Coupling Results for SOSI f1	131
Fig 6.26.	Adaptability Results for SOSI f1	132
Fig 6.27.	SOSI f1 with Three SOSIC	132
Fig 6.28.	Example Results for Adaptability and Agility	134
Fig 6.29.	Methodology	136
Fig 6.30.	Piecewise Constant SOSI	136
Fig 6.31.	SOSI Analysis Process	139
Fig 6.32.	Operational Node Connectivity Diagrams, OV-2s	141
Fig 6.33.	Operational Data Model, OV-7	142
Fig 6.34.	Operational Activity Diagram, OV-5	143
Fig 6.35.	System Interface Descriptions, SV-1s	145
Fig 6.36.	Physical Schema, SV-11	146
Fig 6.37.	Systems Functionality Description, SV-4	148
Fig 6.38.	SOSI Architecture SV-1	149
Fig 6.39.	SOSI Architecture SV-11	150
Fig 6.40.	SOSI Architecture SV-4	151
Fig 6.41.	SOS Model Driven Development Process	153
Fig 6.42.	Transformation Process	154
Fig 6.43.	Element Activity Diagrams	155
Fig 6.44.	Element SOSIC Participation	156
Fig 6.45.	Activity Diagram Transformation Rules	157
Fig 6.46.	CPN Data Model	158
Fig 6.47.	Transformation Process	159
Fig 6.48.	Example CPN Data Model	160
Fig 6.49.	Top-level CPN Representation	160
Fig 6.50.	Sub-pages for Example CPN	161
Fig 7.1.	ESG Operational Concept Graphic (OV-1)	172
Fig 7.2.	Planning and Coordination OV-2	173
Fig 7.3.	Blue Force Tracking OV-2	174
Fig 7.4.	Process and Disseminate Intelligence Information OV-2	174
Fig 7.5.	Planning and Coordination Capability OV-7	177
Fig 7.6.	Blue Force Tracking Capability OV-7	177
Fig 7.7.	Process and Disseminate Intelligence Information Capability OV-7	178
Fig 7.8.	Planning and Coordination OV-5	179
Fig 7.9.	Blue Force Tracking OV-5	179
Fig 7.10.	Process and Disseminate Intelligence Information OV-5	180
Fig 7.11.	Relationship Among Architecture Views	180
Fig 7.12.	Planning and Coordination SV-1	181

Fig. 7.13.	Blue Force Tracking SV-1	182
Fig. 7.14.	Process and Disseminate Intelligence Information SV-1	182
Fig. 7.15.	Planning and Coordination SV-11	185
Fig. 7.16.	Blue Force Tracking SV-11	185
Fig. 7.17.	Process and Disseminate Intelligence information SV-11	186
Fig. 7.18.	Planning and Coordination SV-4	187
Fig. 7.19.	Blue Force Tracking SV-4	188
Fig. 7.20.	Process and Disseminate Intelligence Information SV-4	189
Fig. 7.21.	Tactical Level Command and Control System Activity Diagram	190
Fig. 7.22.	Peer-to-Peer Architecture	192
Fig. 7.23.	Client Server SOSI with One Server	193
Fig. 7.24.	Service Oriented Architecture with One Instance of Each Service	194
Fig. 7.25.	P2P SOSI Architecture SV-11, part 1	194
Fig. 7.26.	P2P_1 SV-1 Six Nodes	195
Fig. 7.27.	P2P_2 SV-1 Eight Nodes	196
Fig. 7.28.	P2P_3 SV-1 Four Nodes	196
Fig. 7.29.	P2P SOSI Architecture SV-11	199
Fig. 7.30.	P2P SOSI Architecture Process and Disseminate Intelligence Information SOSIC	200
Fig. 7.31.	Conduct Support Operations SOSIC	201
Fig. 7.32.	Blue Force Tracking SOSIC	202
Fig. 7.33.	Conduct Security Operations SOSIC	203
Fig. 7.34.	P2P_1 SOSI Alternative Top Level CPN Representation	205
Fig. 7.35.	P2P_1 Port Node CPN	206
Fig. 7.36.	P2P_1 SOSI Ground Station Node CPN	206
Fig. 7.37.	P2P_1 SOSI Tarawa Node Colored Petri Net	206
Fig. 7.38.	P2P_1 SOSI Harper's Ferry Node CPN	207
Fig. 7.39.	P2P_1 SOSI Beach Node CPN	207
Fig. 7.40.	P2P_1 SOSI Satellite Node CPN	208
Fig. 7.41.	P2P SOSI Architecture Results	212
Fig. 7.42.	P2P SOSI Group Adaptability	213
Fig. 7.43.	CS1 SOSI Group Adaptability	214
Fig. 7.44.	CS2 SOSI Group Adaptability	214
Fig. 7.45.	CS3 SOSI Group Adaptability	214
Fig. 7.46.	High Cohesion Node Example CS Architecture	215
Fig. 7.47.	CS Low Cohesion Node	216
Fig. 7.48.	CS SOSI Architecture Adaptability	216
Fig. 7.49.	SOA1 SOSI Group Adaptability	217
Fig. 7.50.	SOA2 SOSI Group Adaptability	217
Fig. 7.51.	SOA3 SOSI Group Adaptability	218
Fig. 7.52.	SOA SOSI Architecture Adaptability	218
Fig. 7.53.	Impact of Highly Reused Elements	219
Fig. 7.54.	Case Study Adaptability Results	220
Fig. 7.55.	Case Study Exclusiveness Results	220
Fig. 7.56.	Case Study Agility Results	221
Fig. 7.57.	Summary Graphic of Case Study Results	222

LIST OF TABLES

Table 2.1	Hofstede's Scores for the Three Countries	13
Table 2.2	Cultural Constraints Corresponding to ACE	13
Table 2.3	Cultural Constraints Corresponding to GCE	13
Table 2.4	Cultural Constraints Corresponding to CSSE	13
Table 6.1	System of Systems Characteristics [Maier, 1996]	91
Table 6.2	Taxonomy for Describing a System of Systems [DeLaurentis, 2005]	92
Table 6.3	All View Products [DODAF, 2007b]	95
Table 6.4	Operational View Products [DODAF, 2007b]	95
Table 6.5	System View Products [DODAF, 2007b]	96
Table 6.6	Technical Standards View Products [DODAF, 2007b]	97
Table 6.7	Cohesion Example	130
Table 6.8	Degree of Reuse Example Data	133
Table 6.9	Degree of Reuse Calculation for SOSI f1	133
Table 6.10	Operational Information Exchange Matrix, OV-3	141
Table 6.11	Operational Rules Model, OV-6a	141
Table 6.12	Systems Data Exchange Matrix, SV-6	145
Table 6.13	Operational Activity to Systems Function Traceability Matrices, SV-5s	146
Table 6.14	Systems Rules Models, SV-10a's	147
Table 6.15	SOSI Architecture SV-6	149
Table 6.16	SOSI Architecture Rules Models, SV-10a	150
Table 7.1	ESG Operational Information Exchange Matrix OV-3	175
Table 7.2	ESG Operational Rules Model OV-6a	176
Table 7.3	Planning and Coordination SV-6	183
Table 7.4	Blue Force Tracking SV-6	183
Table 7.5	Process and Disseminate Intelligence Information SV-6	183
Table 7.6	Planning and Coordination SV-10a	184
Table 7.7	Blue force Trackin SV-10a	184
Table 7.8	Process and Disseminate Intelligence Information SV-10a	184
Table 7.9	Planning and Coordination SV-5	191
Table 7.10	Blue Force Tracking SV-5	191
Table 7.11	Process and Disseminate Intelligence Information SV-5	192
Table 7.12	P2P Systems Data Exchange Matrix, SV-6	197
Table 7.13	P2P Systems Rule Model, SV-10a	198
Table 7.14	P2P SOSI Architecture SV-5	204
Table 7.15	Cohesion SOSI P2P_1	209
Table 7.16	SOSI P2P_1 Coupling Results	209
Table 7.17	P2P Degree of Reuse Data	210
Table 7.18	P2P Degree of Reuse and Exclusiveness Results	210

SECTION 1

Executive Summary

1.1 Objectives

The objectives of this research effort were: (a) *to develop a process for conducting effects based operations and design an adaptive command and control architecture for the organization that executes this process; and (b) to develop an executable model of the organization suitable for the conduct of experiments using a model-based experimental paradigm.* The technical approach to both problems is based on work carried out under the Adaptive Architectures for Command and Control (A2C2) program and constitutes a major extension of that work in several ways: it addresses the changes that have been initiated by DoD in the design of architectures that now require the inclusion of services as a key enabler for net centric operations; it introduces a set of metrics and an approach for evaluating systems of systems; and it applies these results to an Expeditionary Strike Group. The Expeditionary Strike Group functions at both the tactical and the operational level. The perspective of the commander and the staff of an ESG is at the operational level of war; that of the component commanders can be at the tactical or the operational level. For example, operational assessment occurs at the ESG commander's level but also at the Marine Expeditionary Unit (MEU) Commander's level.

1.2 Tasks

The proposed research effort was organized in six tasks. The first four tasks represented the basic research effort; Task 5 was the outreach effort, while Task 6 was the documentation task.

- Task 1. On the basis of the definitions that articulate the Effects Based Operations (EBO) construct, develop a process for conducting EB planning, execution monitoring, and assessment. Take an architectural approach in describing the process for conducting Effects based Operations. Develop the Operational View of the architecture and use an executable model of the architecture to determine its properties.
- Task 2. Using the five stage decision maker model and the Lattice algorithm, design the family of organizational architectures that have embedded in them the process defined in Task 1. Since the behavior and performance characteristics of each member of the family of organizations will differ, develop a characterization of these architectures so that a selection can be made based on mission defined parameters.
- Task 3. Construct a scenario and a mission appropriate for an ESG. Use this scenario to identify the selection parameters and select the organizational structures that are appropriate for the defined mission. Conduct computational experiments to evaluate Measures of Performance and Measures of Effectiveness.

Task 4. Consider a scenario in which the organization needs to adapt. Use the morphing algorithm of Perdu and the evaluation approach of Handley¹ to determine the adaptive architecture for the organization. Conduct additional computational experiments to measure the performance and effectiveness of the adaptive organization.

Task 5. Continue to conduct an outreach program with Expeditionary Strike Groups and develop prototype tools that can be transitioned for experimental use by operators.

Task 6. Document the research results in technical reports in accordance with ONR requirements and in conference and journal papers.

In the three year period of the research project, a number of major changes in the architecture environment took place that led to some changes in the emphasis placed in these tasks. Consequently, the research tasks in the proposal were organized into different research units that covered the same scope of work. The results are presented in the six technical sections of this report.

As naval operations become other than conventional war – whether against transnational terrorist threats or conducting stabilization operations – the need to broaden the focus of models that support effects based planning and operations has become critical. One major weakness is the absence of socio-cultural attributes used in the models for course of action selection and effects based planning.

In Section 2, an algorithm for the design of multi-national organizations that takes into account cultural dimensions is presented. This is based on an extension of the work of Perdu and Handley as described in Task 4. The approach was illustrated through an example based on an Expeditionary Strike Group (ESG) conducting a Humanitarian Assistance/Disaster Relief mission (Tasks 3 and 5).

One of the key issues in Effects Based Operations is the ability to assess how effective a course of action would be (Task 1). The model driven experimentation process, developed further so that it can utilize executable models of DOD Architecture Framework compliant architectures, was used in conjunction with a critical experiment to explore procedures for effects based assessment. The results are documented in Section 3.

In Section 4, an approach based on Influence Nets that enables analysts to evaluate a complex situation in which an adversary is embedded in a society from which he is receiving support. A layered modeling approach is described that enables analysts to examine and explain how actions of the military and other entities may result in desired and undesired effects, both on the adversary and the population as a whole. Several techniques and associated metrics are used for comparing contemplated courses of action. This is an extension of Task 1.

¹ Both developed under earlier ONR grants

As the net centricity concept began to be integrated into the Global Information Grid, the Service Oriented Architecture paradigm became a key driver to architecture development. A new version of the DOD Architecture Framework, version 1.5, was released that enabled the use of services as a method for implementing required operational activities. In lieu of the original Task 4, a Task 4' was articulated in which the architecture design methodology was re-examined and then extended to accommodate the Service Oriented Architecture (SOA) construct. Similarly, it was recognized that in order to accomplish Task 4, an approach to assess or evaluate adaptive architectures was needed. A major basic research effort was undertaken to address adaptability in the context of a System of Systems that enables adaptability. This became Task 4". Section 5 describes a process for creating a DoDAF 1.5 compliant architecture that includes the description of the SOA aspects, a mapping and a process for converting that architecture to an executable model, and the use of that executable model in the evaluation of logical, behavioral, and performance aspects of the architecture. The concepts are illustrated with a detailed Case Study.

Sections 6 and 7 contain new work that extends substantially the scope of Task 3. A goal of an agile organization is the ability to adapt its structure to constantly changing operating environments so it can provide the multiple capabilities that enable mission accomplishment. A challenge for the system of systems (SOS) engineer is that while the SOS is being developed, the operating environment it was designed for changes. This situation causes significant uncertainty as to whether the SOS will meet the needs of the organization when finally deployed. To mitigate this uncertainty, SOS architectures need to be assessed for their ability to deploy in more than one configuration so that they can support adaptive organizations. Past architecture assessments and performance characteristics are primarily system focused and do not address the dynamics of the adapting organization and the consequences on the interacting constituent systems of the SOS. In this work, a first attempt is made to provide measures for assessing and comparing SOS architectures for their ability to adapt to the current operating environment and their ability to provide multiple capabilities concurrently.

A SOS is defined as being composed of individual Elements that can be organized into Nodes. Each Element belongs to one and only one Node. *Cohesion* is a measure of the relatedness of the Elements within a Node. *Coupling* is a measure of the interdependence among the Nodes. *Adaptability* is defined as the ability of a SOS to respond to changes in the allocation of Elements to Nodes; it is computed using the concepts of Coupling and Cohesion. The *Degree of Reuse* measures the extent to which Elements support multiple capabilities. *Agility* measures the ability of the SOS to execute multiple processes concurrently and adapt to changing situations. Agility is a function of Adaptability and Degree of Reuse.

The methodology provides the information required to assess the Adaptability and Agility of a proposed or actual SOS architecture. The process begins by identifying from the operational view of the architecture the capabilities that must be realized by the SOS alternatives. The SOS architecture describes how a particular subset of Elements organized into Nodes will realize the capabilities; it is the system view of the architecture. The SOS is transformed automatically into

an executable model using Colored Petri Nets; invariant analysis and simulation are used to compute Coupling and Cohesion while the Degree of Reuse is computed directly for each SOS. These three measures are then used to compute the Adaptability and Agility measures. Alternative architecture patterns are then compared in terms of their adaptability and agility. One advantage of the approach is that it can be applied early in the systems engineering process to help select preferred architecture alternatives.

In Section 7, a case study based on a complex mission for an Expeditionary Strike Group is presented to illustrate the application of the assessment methodology. It shows that different architecture types or patterns yield distinct values for the Adaptability and Agility measures that are consistent with the qualitative differences in the tested architectures.

SECTION 2

Computational Models of Multi-National Organizations

A. H. Levis, Smriti K. Kansal, A. E. Olmez, and Ashraf M. AbuSharekh

2.1 Introduction

A key objective in organization design is to relate structure to behavior. An executable model, i.e., a formal mathematical model with characteristics that are traceable to the static architecture designs, is used to determine the properties of the model and its performance characteristics. A wealth of theoretical results on discrete event dynamical systems, in general, and Colored Petri nets, in particular, can be applied to the executable model.

The problem of modeling multi-national organizations such as those found in military coalition operations has received renewed attention. Coalition partners may have differences in equipment or materiel, differences in command structures, differences in constraints under which they can operate, and, last but not least, differences in culture. The differences in equipment and in operational constraints can be handled easily in the existing modeling framework. Differences in command structures require some additional work to express them in structural and quantitative ways. The real challenge is how to express cultural differences in these, primarily mechanistic, models of organizations.

This work focuses on the ability to introduce attributes that characterize cultural differences into the organization design and use simulation to see whether these parameters result in significant changes in structure. The objective, therefore, is to relate performance to structural features but add attributes that characterize cultural differences. Specifically, the attributes or dimensions defined by Hofstede (2001) are introduced in the design process in the form of constraints on the allowable interactions within the organization. In Section 2.2, the modeling approach is described briefly since it has been documented extensively in the literature. In Section 2.3, the Hofstede dimensions are introduced and then applied to the organization design algorithm. In Section 2.4, an illustrative example is presented, followed by conclusions.

2.2 The Decision Maker Model and Organizational Design

The five-stage interacting decision maker model (Levis, 1993) had its roots in the investigation of tactical decision making in a distributed environment with efforts to understand cognitive workload, task allocation, and decision-making. This model has been used for fixed as well as variable structure organizations (Perdu and Levis, 1998). The five-stage decision maker (DM) model is shown in Figure 2.1.

The DM receives signals from the external environment or from another decision maker. The Situation Assessment (SA) stage represents the processing of the incoming signal to obtain the assessed situation that may be shared with other DMs. The decision maker can also receive situation assessment signals from other decision makers within the organization; these signals are then fused together in the Information Fusion (IF) stage. The fused information is then processed at the Task Processing (TP) stage to produce a signal that contains the task information necessary to select a response. Command input from superiors is also received. The Command Interpretation (CI) stage then combines internal and external guidance to produce the input to the Response Selection (RS) stage. The RS stage then produces the output to the environment or to other organization members. The key feature of the model is the explicit depiction of the interactions with other organization members and the environment.

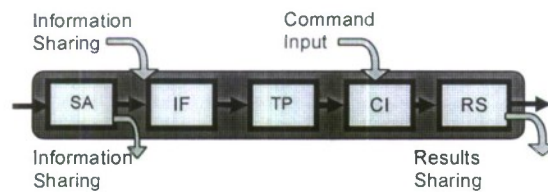


Figure 2.1: Model of the Five-Stage Decision Maker

These interactions follow a set of rules designed to avoid deadlock in the information flow. The representation of the interactions can be aggregated into two vectors e and s , representing interactions with the external environment and four matrices F , G , H and C specifying intra-organizational interactions (Fig. 2.2).

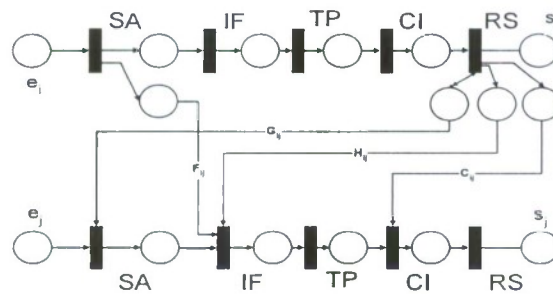


Figure 2.2: One-Sided Interactions Between DM_i and DM_j

The analytical description of the possible interactions between organization members forms the basis for an algorithm that generates all the architectures that meet some structural constraints as well as application-specific constraints that may be present. The most important constraint addresses the connectivity of the organization - it eliminates information structures that do not represent a single integrated organization.

Remy and Levis (1988) developed an algorithm, named the Lattice algorithm, that determines the maximal and minimal elements of the set of designs that satisfy all the constraints; the entire set can then be generated from its boundaries. The algorithm is based on the notion of a simple path - a directed path without loops from the source to the sink. Feasible architectures are obtained as unions of simple paths. Consequently, they constitute a partially ordered set. The algorithm receives as input the matrix tuple of dimension $n \{e, s, F, G, H, C\}$, where n is the number of organization members. A set of four different structural constraints is formulated that applies to all organizational structures being considered.

- R1 A directed path should exist from the source to every node of the structure and from every node to the sink.
- R2 The organizational structures should be acyclical.
- R3 There can be at most one link from the RS stage of a DM to each one of the other DMs; i.e., for each i and j , only one element of the triplet $\{G_{ij}, H_{ij}, C_{ij}\}$ can be nonzero.
- R4 Information fusion can take place only at the IF and CI stages. Consequently, the SA and RS stages of each DM can have only one input.

To introduce user-defined constraints that will reflect the specific application the organization designer is considering, appropriate 0s and 1s can be placed in the arrays $\{e, s, F, G, H, C\}$. The other elements will remain unspecified and will constitute the degrees of freedom of the design.

A feasible structure is one that satisfies both the structural and user-defined constraints. A maximal element of the set of all feasible structures is called a maximally connected organization (MAXO). Similarly, a minimal element is called a minimally connected organization (MINO). The design problem is to determine the set of all feasible structures corresponding to a specific set of constraints. The Lattice algorithm generates, once the set of constraints is specified, the MINOs and the MAXOs that characterize the set of all organizational structures that satisfy the requirements. This methodology provides the designer of organizational structures with a rational way to handle a problem whose combinatorial complexity is very large. Having developed a set of organizational structures that meets the set of logical constraints and is, by construction, free of structural problems, we can now address the problem of incorporating attributes that characterize cultures.

2.3 Modeling Cultural Attributes

Hofstede (2001) distinguishes dimensions of culture that can be used as an instrument to make comparisons between cultures and to cluster cultures according to behavioral characteristics. Culture is not a characteristic of individuals; it encompasses a number of people who have been conditioned by the same education and life experience. Culture, whether it is based on nationality or group membership such as the military, is what the individual members

of a group have in common (De Mooij, 1998). To compare cultures, Hofstede originally differentiated them according to four dimensions: *uncertainty avoidance (UAI)*, *power distance (PDI)*, *masculinity-femininity (MAS)*, and *individualism-collectivism (IND)*. The dimensions were measured on an index scale from 0 to 100, although some countries may have a score below 0 or above 100 because they were measured after the original scale was defined in the 70's. The hypothesis here is that these dimensions may affect the interconnections between decision makers working together in an organization. Organizations with low power distance values are likely to have decentralized decision making characterized by a flatter organizational structure; personnel at all levels can make decisions when unexpected events occur with no time for additional input from above. In organizations with low scores on uncertainty avoidance, procedures will be less formal and plans will be continually reassessed for needed modifications.

The trade-off between time and accuracy can be used to study the affect of both power distance and uncertainty avoidance (Handley and Levis, 2001). Messages exchanged between decision makers can be classified according to three different message types: information, control, and command ones. Information messages include inputs, outputs, and data; control messages are the enabling signals for the initiation of a subtask; and command messages affect the choice of subtask or of response. The messages exchanged between decision makers can be classified according to these different types and each message type can be associated with a subjective parameter. For example, uncertainty avoidance can be associated with control signals that are used to initiate subtasks according to a standard operating procedure. A decision maker with high uncertainty avoidance is likely to follow the procedure regardless of circumstances, while a decision maker with low uncertainty avoidance may be more innovative. Power distance can be associated with command signals. A command center with a high power distance value will respond promptly to a command signal, while in a command center with a low power distance value this signal may not always be acted on or be present.

Cultural constraints help a designer determine classes of similar feasible organizations by setting specific conditions that limit the number of various types of interactions between decision makers. Cultural constraints are represented as interactional constraint statements. An approach for determining the values of these constraints has been developed by Olmez (2006). The constraints are obtained using a linear regression on the four dimensions to determine the change in the range of the number of each type of interaction that is allowed.

$$dY = c + \alpha(PDI) + \beta(UAI) + \gamma(MAS) + \delta(IND)$$

where Y is #F or #G or #H or #C

Example: #F ≤ 2, #G = 0, 1 ≤ #H ≤ 3, #C = 3

C-Lattice Algorithm. This is an extension of the Lattice algorithm that allows cultural constraints to be imposed as additional structural constraints, R5-R8, on the solution space. For the cultural constraint example given above, they become:

R5: The number of F type interactions must be between 0 and 2

R6: The number of G type interactions must equal 0

R7: The number of H type interactions must lie between 1 and 3

R8: The number of C type interactions must equal 3

The flowchart in Fig. 2.3 explains the generation of the culturally constrained solution. MAXOs and MINOs are generated using the same algorithm described in Remy and Levis (1988). The “Build Lattices” step checks if a MINO is contained within a MAXO. If it is, then the MINO is connected to that MAXO and forms part of a lattice. For each lattice, we check the MINO to see if it violates the cultural constraints. For example, if the number of F type interactions in the MINO is two and cultural constraint allows only one, then the MINO does not satisfy the cultural attributes and since the MINO is the minimally connected structure in that lattice, no other structure will satisfy the constraints. Hence the lattice can be discarded. If the MINO does pass the boundary test, then simple paths are added to it to satisfy the cultural constraints R5 to R8. The corresponding minimally connected organization(s) is now called the C-MINO(s) (culturally bound MINO). Similarly, by subtracting simple paths from the MAXO, C-MAXO(s) can be reached. The step “Build C-Lattices” connects the C-MINOs to the C-MAXOs. The advantage of using this approach is that the designer does not have to know the cultural attributes at the start of the analysis. He can add them at a later stage. This also enables him to study the same organization structure under different cultures, which will be useful in our coalition scenario.

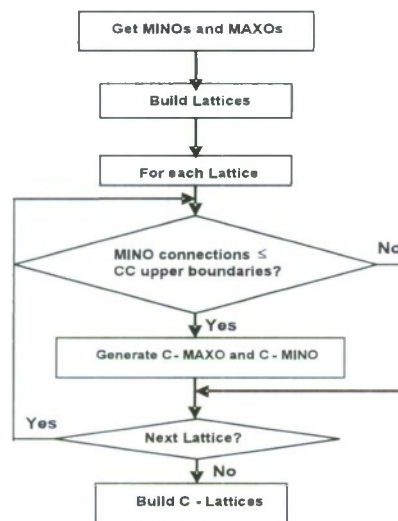


Figure 2.3: Flowchart for Culturally Constrained Solution

2.4 Coalition Modeling Using CAESAR III

The proposed computational approach for the design of coalition operations is illustrated using a hypothetical example in which an emergency situation in an island nation requires rapid

humanitarian assistance and disaster relief as well as securing military assets. The alternative architecture designs and the associated simulations to evaluate performance were carried out using a new application called CAESAR III developed in System Architectures Lab at GMU.

The scenario depicts a situation in which anarchy has risen on an island due to a recent earthquake that caused substantial damage. The infrastructure and many of the government buildings are destroyed in the island's capital. The US maintains a ground station that receives data from space assets. It is concerned about the rising tensions, as there has been opposition to its presence on the island. As a result, the US decides to send an Expeditionary Strike Group (ESG) to the island to provide timely Humanitarian Aid/ Disaster Relief (HA/DR) to three sectors of the island and to counteract the effects of any hostile attacks which may impede the operations of the HA/DR mission and the security of the ground station. As the ESG is away for the first critical day of the operation, countries A and B offer help to support the mission and agree to take part in a Coalition Force that would be commanded remotely by the ESG commander. Since they are close to the island, both countries can deploy elements in a matter of hours, while the ESG rushes to the island.

A team of five units carries out the HA/DR mission. The team is organized in the divisional structure and each unit under the team has its sub-organizations and staff to perform the tasks allocated to it. The five units are: (1) ESGC: Commander; (2) MEUC: Commander of the Marine Expeditionary Unit; (3) ACE: Air Combat Element with its Commander and sub-organizations; (4) GCE: Ground Combat Element with its Commander and sub-organizations; and (5) CSSE: Combat Service Support Element with its Commander and sub-organizations.

It is assumed that country A can provide support as ACE, GCE and CSSE while country B can only provide support as GCE and CSSE. The roles of ESGC and MEUC remain with the US. The countries are able to provide rapid assistance in coordination with each other and the design question becomes the allocation of different tasks to partners in this ad-hoc coalition.

This is a multi-level design problem in which interactions between different decision making units need to be determined both at the higher level (Level-1) as well as at the lower level (Level-2). Level-1 interactions are interactions between culturally homogenous subunits, while the Level-2 problem consists of designing the internal structure of these homogenous subunits on the basis of a defined set of interactional constraints and culture. The structure of the ESG imposes user constraints to design the Level-1 organization. Figure 2.4 shows the block diagram of this organization as designed in CAESAR III; the matrices describing the interactions are shown below.

Figure 2.5 shows the result of running the lattice algorithm on level-1 organization. The solution space contains one MINO, Fig. 2.6, and one MAXO, Fig. 2.7. The designer can pick a structure from this space and use it to design the sub-organizations at level-2.

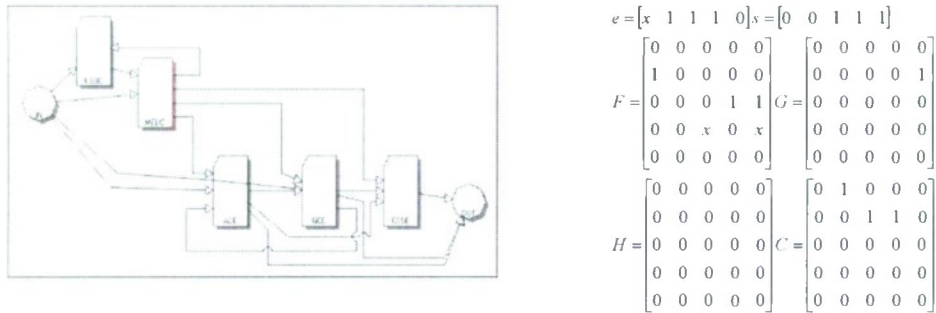


Figure 2.4: Level-1 Organizational Block Diagram.

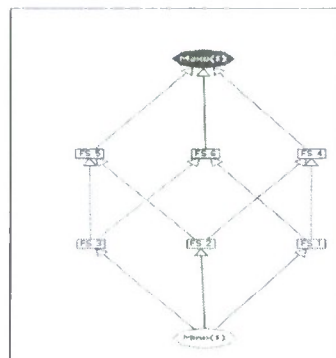


Figure 2.5: Solution Space For Level-1 Organization Design as Seen in CAESAR III

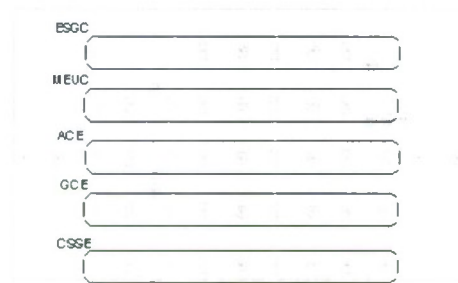


Figure 2.6: MINO of Level-1 Design

Level-1 design is free of cultural constraints. However Level-2 design uses the C-Lattice algorithm to include cultural attributes to form the various coalition options. The sub-organizations of ACE, GCE and CSSE are designed using CAESAR III. Figures 8, 9 and 10 show the respective block diagrams along with the matrices specifying the user constraints. Since the US always performs the roles of ESGC and MEUC, these sub-organizations are not decomposed further.

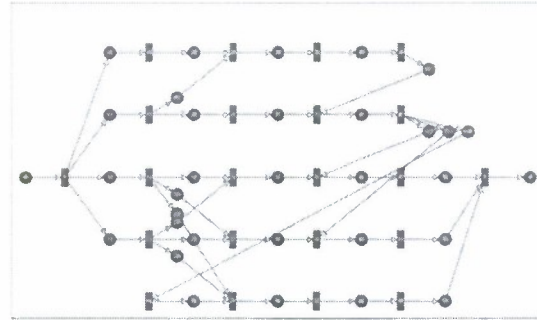
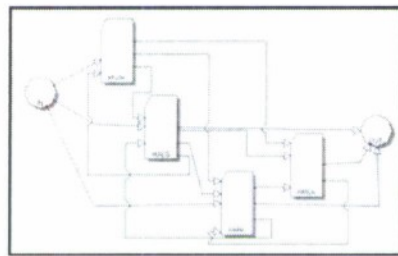


Figure 2.7: MAXO of Level-1 Design

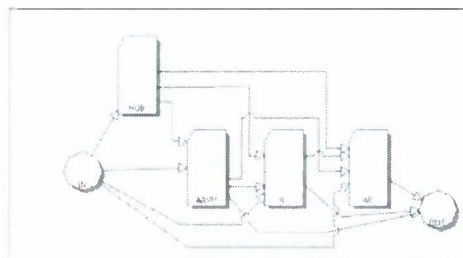


$$e = [x \ 1 \ 1 \ 0] s = [0 \ x \ 1 \ 1]$$

$$F = \begin{bmatrix} 0 & 0 & 0 & 0 \\ x & 0 & 0 & x \\ 0 & x & 0 & x \\ 0 & 0 & 0 & 0 \end{bmatrix} \quad G = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

$$H = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & x & 0 & x \\ 0 & 0 & x & 0 \end{bmatrix} \quad C = \begin{bmatrix} 0 & 1 & x & x \\ 0 & 0 & 1 & x \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

Figure 2.8: Block Diagram for ACE



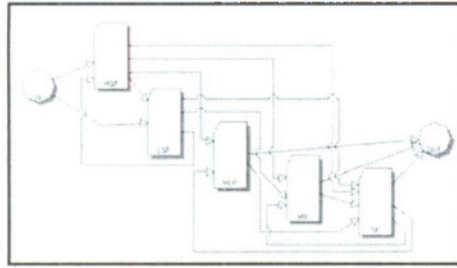
$$e = [1 \ x \ x \ x] s = [0 \ 1 \ 1 \ 1]$$

$$F = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \quad G = \begin{bmatrix} 0 & x & x & x \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

$$H = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & x & x \\ 0 & 0 & 0 & x \\ 0 & 0 & 0 & 0 \end{bmatrix} \quad C = \begin{bmatrix} 0 & x & x & x \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

Figure 2.9: Block Diagram for GCE

Table 2.1 gives the Hofstede's scores for US, Country A and Country B. Using a multiple linear regression model, these scores are converted into limits to be placed on allowable interactions based on culture. These are imposed as additional structural constraints on the solution space of the sub-organizations. The cultural constraints for the three sub-organizations are shown in tables 2.2, 2.3 and 2.4. Maximum indicates the limit placed on the number of interactions by user constraints.



$$e = [1 \ 1 \ 0 \ 0 \ 0] s = [0 \ 0 \ 1 \ 1 \ x]$$

$$F = \begin{bmatrix} 0 & x & 1 & 0 & 0 \\ x & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix} \quad G = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

$$H = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & x \\ 0 & 0 & 0 & 0 & x \\ 0 & 0 & x & x & 0 \end{bmatrix} \quad C = \begin{bmatrix} 0 & 1 & 0 & x & x \\ 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

Figure 2.10: Block Diagram for CSSE

Table 2.1. Hofstede's Scores for the Three Countries

Country	PDI	IND	MAS	UAI
US	40	91	62	46
A	38	80	14	53
B	66	37	45	85

Table 2.2. Cultural Constraints Corresponding to ACE

Country	#F	#G	#H	#C
Maximum	$0 \leq F \leq 4$	0	$0 \leq H \leq 3$	$2 \leq C \leq 5$
US	$3 \leq F \leq 4$	0	$2 \leq H \leq 3$	3
A	2	0	$2 \leq H \leq 3$	3
B	2	0	1	$4 \leq C \leq 5$

Table 2.3. Cultural Constraints Corresponding to GCE

Country	#F	#G	#H	#C
Maximum	0	$0 \leq G \leq 3$	$0 \leq H \leq 3$	$0 \leq C \leq 3$
US	0	2	$2 \leq H \leq 3$	2
A	0	2	$2 \leq H \leq 3$	1
B	0	$2 \leq G \leq 3$	2	$2 \leq C \leq 3$

Table 2.4. Cultural Constraints Corresponding to CSSE

Country	#F	#G	#H	#C
Maximum	$1 \leq F \leq 3$	$\square$	$0 \leq H \leq 4$	$3 \leq C \leq 5$
US	$2 \leq F \leq 4$	$\square$	$3 \leq H \leq 4$	$\square$
$\square$	$\square$	$\square$	$3 \leq H \leq 4$	$\square$
$\square$	$\square$	$\square$	$\square$	$4 \leq C \leq 5$

Using the C-Lattice algorithm, the solution space for each sub-organization is computed for each culture and a suitable structure is selected by the user. These structures are then used to form the different coalition options and analyze the performance. In view of the limited space, the complete solution spaces are not shown here. Figures 2.11-12.3 show the structures selected

by the user for each country for CSSE. A similar approach can be used to select different structures to be used for ACE and GCE.



Figure 2.11: GCE Structure Selected for US



Figure 2.12: GCE Structure Selected for Country A



Figure 2.13: GCE Structure Selected for Country B

Once the structure is selected, CAESAR III exports it as a Colored Petri net to *CPN Tools* where it can be simulated to analyze performance. For the given scenario, based on the availability of support from the two countries, eight coalition options are possible, excluding the homogeneous option of all US. The five sub-organizations are combined together using Level-1 MINO and the eight options were simulated to study performance in terms of tasks served. The following assumptions were made. Each process (transition) needs 50 units of processing time. Each additional incoming link increases this time by 50 units. The reasoning is that the additional input(s) will require more processing. Hence, structures that have more interactions will take

more time to process the tasks, which will affect the overall performance. Figure 2.14 shows the results of this analysis for all combinations. The x-axis shows the percentage of tasks **un-served**.

Based on these results, the US-US-US-B-A coalition structure performs best. Most options with country B in the CSSE role perform badly. This is because country B needs a high number of command relationships and the structure of CSSE allows for this to occur, thereby increasing the processing delay. User constraints on GCE allow for very similar cultural constraints for all countries; changing the ordering in this role does not change the performance very much. Similar results were obtained when the coalition options were simulated using a Level-I MAXO organization.

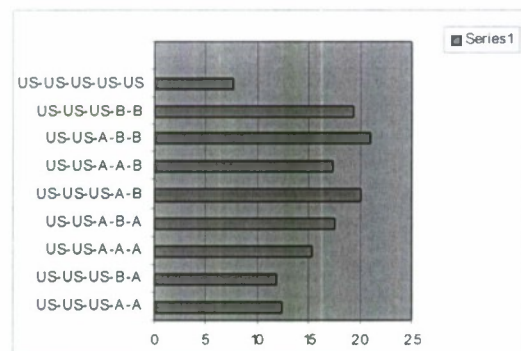


Figure 2.14: Percent of tasks un-served for coalition options.

2.5 Conclusion

A previously developed methodology for the computational design of information processing and decision making organizations has been enhanced to include cultural constraints that affect the choice of organizational structures. While the Hofstede cultural dimensions have been used, other cultural metrics can be used to derive the cultural constraints. A simple example illustrates the approach for designing coalition organizations and analyzing their performance. The results indicate that culture does affect the structure and working of organizations thereby affecting the overall performance. This could aid in the allocation of different tasks to partners in an ad-hoc coalition.

2.6 References

1. Handley H, Levis, A H (2001) Incorporating heterogeneity in command center interactions. *Information Knowledge Systems Management* 2(4).
2. Hofstede G (2001) *Culture's Consequences: Comparing Values, Behaviors, Institutions, and Organizations Across Nations*, 2nd edn. Sage, CA .
3. Levis A H (1993). A Colored Petri Net Model of Command and Control Nodes. In: Jones R (ed.) *Toward a Science of Command Control and Communications*. AIAA Press, Washington, DC.
4. De Mooij M (1998) *Global Marketing and Advertising: Understanding Cultural Paradoxes*. Sage, CA

5. Olmez, A E (2006) Exploring the Culture-Performance Link in Heterogeneous Coalition Organizations. PhD Thesis, George Mason University, Fairfax, VA.
6. Perdu D M and Levis A H (1998) Adaptation as a morphing process: A methodology for the design and evaluation of adaptive organization structures. *Computational and Mathematical Organization Theory* 4(1): 5-41
7. Remy P and Levis A H (1988) On the Generation Of Organizational Architectures Using Petri Nets. In: G Rozenberg (ed) *Advances in Petri Nets 1988*, Springer-Verlag, Berlin.

SECTION 3

Using Architectures to Support Experimentation

Lee W. Wagenhals and Alexander H. Levis

3.1 Introduction

As new technology and new environments for military operations have evolved over the past 10 years, DoD has been attempting to formulate, evaluate, and implement new command and control concepts. The notion is to build plans based on the effects that will lead to accomplishing the goals and objectives and then to link actions to those effects through known or plausible cause and effect relationships. This shifts the emphasis from focusing on targets to higher level effects that actions that impact those targets may have. Effects Based Operations (EBO) has evolved as the new concept for planning and executing operations. Evolving with this concept are the techniques and procedures needed to develop courses of action and the associated detailed plans. Once plans are made and execution starts, it is necessary to continually assess how well the plans are working. With EBO, this is more than just determining whether targets have been affected as planned (e.g. checking off targets serviced), but also collecting and assessing information to see if the actions are indeed leading to the overall desired effects. The difficulty is that Effects Based Assessment (EBA), particularly at the higher operational level, requires a different approach than the one that has been used in the past, and the techniques and procedures, along with the supporting information processing systems, have not been established. This has created a need to provide a new capability for command and control, but the concepts for supporting EBA at the operational level are not in place. In short, the problem is complex and unprecedented, and the needs of the user are ill structured.

In a concurrent project¹, the System Architectures Laboratory participated in a critical experiment (CE) in which a spiral approach had been selected consisting of a repetitive sequence of development and experimentation to evolve the concepts, techniques, and procedures along with the systems that will support them for Effects Based Assessment (EBA).

Because the needs of the users are ill structured and because DoD has mandated the use of architectures, an architecture based approach was adopted for the critical experiment. Furthermore, a model driven experimentation technique was selected to conduct the various spirals of the experiment.

This section describes the implementation of the architecture-based model-driven experimentation. Specifically, the section illustrates how an executable model of an architecture

¹ AFRL/RI's Dynamic Air and Space Execution and Assessment (DASEA)

has been used to guide the design of both computational and human-in-the-loop experiments to test and verify a new concept and supporting processes and systems for Effects Based Assessment. A repetitive, or spiral, approach was used to test and refine, at increasing levels of specificity, the proposed processes that will be used by operators and the supporting systems to conduct the assessments. Since initially there was no operational concept for operational level EBA, one had to be postulated. Then this initial concept was refined using a DOD Architecture Framework (DODAF) compliant Operational View (OV) of the operational concept. The OV was used to design a human-based table-top experiment to examine and evaluate the processes defined in the architecture. Lessons learned were used to refine the architecture. An initial Systems View of the architecture was created based on the OV to help guide the design and development of systems that would support the processes defined in the OV. This systems view also guides the design of the next spiral that broadens the experiment with the use of humans and the system.

The rest of the section is organized as follows. Section 3.2 provides the highlights of the model driven experimentation process showing the relationships between the architecture, executable models and model-based computational and human-based experimentation. Section 3.3 provides the detail of the first spiral that followed the process including the development of the operational concept, its refinement using the DODAF compliant architecture views, and the development of the discrete event executable model of the architecture views. It also describes how the executable model was used to conduct computational experiments and to verify the correctness of the architecture and thus of the new concepts. Section 3.4 summarizes the findings of the architecture-based model driven experimentation approach.

3.2 Architecture-Based Model Driven Experimentation

Model Driven Experimentation (MDE) is a concept that has evolved over the past 20 years [Levis and Vaughan, 1999]. For the last 15 years, the GMU System Architectures Lab has participated in and refined MDE with a research team set up by ONR consisting of the Naval Postgraduate School and several universities [Handley et al., 1999]. Nine experiments have been conducted (both human-in-the-loop and computational ones) using MDE. The focus of these experiments has been on the design of command and control organizations and the understanding of how different organizational structures can effect behavior and performance, particularly in a changing environment where the ability to adapt is important.

Within the Critical Experiment project, the concept for organizational experimentation was expanded as follows: Given (1) an operational concept, a process for carrying out the operational concept, an organizational design, and a system that supports that organization in carrying out the process, and (2) a set of hypotheses or propositions about those items that need to be evaluated, develop a rigorous process to design and conduct experiments that that will test the hypotheses and use the results of the experiments to refine the operational concept, process, organization, and system. The unprecedented, complex, and ill-structured nature of the problem led to the conclusion that an architecture modeling paradigm used in conjunction with the MDE

that had been developed for organizational experimentation would be appropriate. Using the DODAF views facilitates the experimental design. The OV provides a rigorous framework for describing the operational concept, the organizational structure, and the operational process. Executable models can be derived from the OV products and used to conduct computational experiments focusing on the proposed process and organizational structure. The DODAF Systems View (SV) describes the arrangement and interconnections of a set of systems and their components that can support the processes and organizations described in the Operational View. Executable models of the Systems View also can be derived from the SV products and evaluated experimentally. The SV can then be used to drive the design of the actual systems that will support the process that will be used in the experiments. Figure 3.1 shows the basic process.

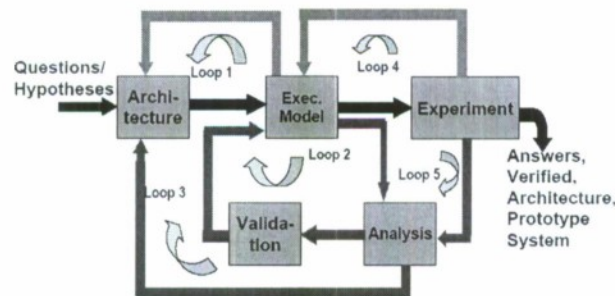


Figure 3.1: Architecture Based MDE Process

The process starts with the formulation of questions, propositions, or hypotheses to be examined in the experiment. In the case of DASEA, the initial hypothesis was that it is possible to develop a process that would enable useful operational level assessment of progress in the execution of an Effects Based Plan. Next, an operational concept is needed to support the development of an architecture. The hypothesis and the operational concept then are used to help create an initial architecture model of the process or system that will be the subject of the experiment. A critical element of this step is the determination of the boundary between the process and system that are being used and the environment. The environment contains the things that interact with the process, i.e., the sources of inputs and the sinks that receive the outputs. The understanding of these interactions will be needed to develop the scenario that will be used to conduct the experiment and collect the data needed to answer the questions or test the hypothesis. Once the model of the architecture has been completed, it can be converted into an executable model. The procedures for doing this have been described in [Wagenhals et al., 2000 and 2003]. Once the executable model is created, it can be used to verify the architecture, detecting errors and flaws in the architecture design (Loop1). These findings also can be used to refine the scenarios that will be used in the experiments. After the architecture and the scenario have been verified, the executable model can be used in a series of computational experiments generating preliminary analyses about the hypotheses. These analyses may result in changes and improvements to the executable model (Loop 2). The final results may also result in a refinement of the architecture which may be used to drive the design of systems that support the

processes (Loop 3). The primary reason for developing the model, both architecture and executable, is to “tune” the experiment before it is conducted. In general, these experiments are conducted with human subjects (operators). These experiments using humans can be much more realistic than those done computationally, but they can be quite costly to conduct. Usually, data about a limited number of vignettes of the overall scenario can be obtained. In many cases, the actual experiment will deviate from the planned experiment due to changes in personnel, equipment, or other factors. Despite the limited nature of the human-based experiment, the results can be used to test the accuracy and predictability of the executable model (Loop 4). This is done by setting the conditions that were actually used in the experiment and comparing the executable results with those of the experiment. Changes may be necessary in the executable model and such changes should be reflected in the architecture (Loop 5). Once the executable model has been verified, it can be used to explore computationally many other vignettes. The data collected from both the human-based and the model-based experiments is analyzed to generate the answers to the questions and to address the hypotheses. When used in a spiral approach, the verified architecture and executable model along with the results of the entire spiral are used to start the next spiral in which the MDE process is repeated.

3.3 Application of Model Driven Experimentation

A plan composed of eight steps was developed for the first spiral of the architecture-based model-driven critical experiment.

1. Postulate an operational concept for Operational Level Effects Based Assessment; develop an understanding of the factors and unknowns involved.
2. Use this understanding to define the hypothesis to be evaluated in the spiral.
3. Refine the operational concept using an architecture description. The DODAF 1.0 Operational View was be used.
4. Convert the operational view of the architecture into an executable model to verify the architecture.
5. Develop, conduct, and analyze a table top experiment based on the operational concept and its architecture.
6. Refine the architecture and its executable model based on the findings from step 5.
7. Plan, conduct, and analyze a computational experiment using the executable model to verify the soundness and completeness of the architecture description. Adjust the architecture description based on the findings.
8. Develop an initial Systems View of the architecture that is congruent to the Operational View and use it to guide the design of technology and systems that will be used in the next spiral.

The first step in setting up the experiment was to define an operational concept for effects based assessment capability that is the subject of the experiment. This included defining the boundary between the capability under investigation and the environment in which that capability is used. The capability being investigated is the ability to dynamically conduct operational level EBA based on actions and activities that have taken place, so-called post execution EBA. It is assumed that an Effects Based Plan (EBP) has been developed that is designed to achieve some operational level objectives based on attainment of the commander's intent. The plan is based on an analysis that shows that accomplishing a set of tactical tasks and actions will cause direct effects to occur, and these in turn will cause the higher level effects to be achieved. The achievement of the higher level effects will result in the attainment of the commander's operational level objectives. The desired EBA capability uses the EBP and an incoming stream of data and reports about the execution of that plan and observations about effects to continually determine progress being made toward achieving the Air Component commander's intent. Note that in this operational concept the process of creating and selecting the EBP and the execution of that plan are outside the scope of the EBA capability. Those two activities continually provide inputs to the capability and the capability in turn dynamically assesses the outcomes (or effects) of executed air & space operations with regards to the attainment, or progress towards attainment, of commander's intent.

Once the operational concept was defined, Step 2 in designing the experiment was to establish the experimental hypothesis or proposition to be evaluated. For the first spiral the hypothesis was: Given an effects-based plan that meets commander's intent, data relating to execution outcomes, and an operational-level, post-execution EBA process; it is possible to produce an assessment of the progress towards attaining commander's intent. Note that this is a rather abstract hypothesis. We wanted to first show that the operational concept contained the process and data flows that could be used to create the operational level EBA before exploring the quality and timeliness of that EBA. The latter concerns will be explored only after the first spiral successfully shows that the operational concept and the process derived there from worked.

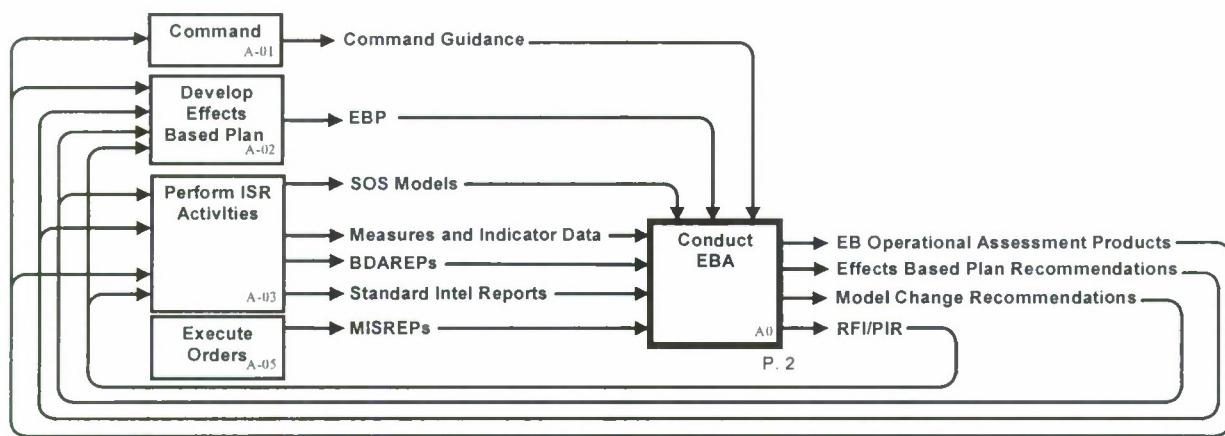
In Step 3, the operational concept needed to be refined, and this was done by creating an architecture description of the operational concept. Because this is a DoD effort, the architecture was created in compliance with the DODAF 1.0. At this point, only an Operational View was developed. The OV is a description of the tasks and activities, operational elements, and information exchanges required to accomplish DoD missions. It conveys these elements using a combination of products including activity, data, and rule models, dynamic descriptions, organization descriptions, and descriptions that identify the activities performed by organizational elements and the information that is exchanged between those elements to carry out the operational concept. The structured analysis methodology was chosen over the object oriented approach because of the familiarity with this methodology by the program manager.

One of the key concepts described in the OV is the set of activities and their relationships. This concept maps directly to the structured analysis approach which is based on functional (or activity) decomposition. One was created using four levels of decomposition. The first level was the context level with the overall activity being "Conduct Operational Level Effects Based Assessment". It was decomposed into three activities, A1, Process Incoming Information, A2, Assess Progress Toward Objectives, and A3, Generate Recommendations and Reports. Each of these was further decomposed to provide more detail into the activities that would be used to carry out the operational concept.

An Activity Model (OV-5) was created based on the functional decomposition using the IDEF0 formalism. The analysis started with an External System Diagram that shows the system (in this case the overall EBA process) in the context of the systems (in this case the organizations) that it interacts with (Figure 3.2). It helped define the system boundary. Note the explicit expression of the purpose and the viewpoint on the diagram.

The A0 page (first level of decomposition) is shown in Figure 3.3. The main inputs (the various messages and data that will be received giving progress reports and observations about effects) are shown coming into A1. The controls are the Effects Based Plan (EBP), System of System Models (SOS Models), EBA Standard Operating Procedures, and Command Guidance. The process produces the four types of output products, the three shown exiting A3 and the Requests for Information (RFI) from A1 and A2. In addition to the Activity Model, other OV products were produced including the Operational Concept Graphic, the Operational Node Connectivity Description, and the Operational Event Trace Diagram. The analysis that led to these architecture artifacts yielded a much more detailed description of the operational concept.

In Step 4, the operational architecture view description was converted to an executable model of the OV using the procedures described in Wagenhals et al., [2000]. The Colored Petri Net methodology implemented in CPNTools® Version 1.4 was used [www.daimi.au.dk]. Colored Petri Nets are the most general form of discrete event dynamical models. They have a graph theoretic basis which allows the analysis of properties of the model and they are executable, so that they can be used in simulation. Figure 3.4 shows the CPNTools® model page that corresponds to the IDEF0 A0 page of Figure 3.3. Note the correspondence of CPN model to the IDEF0 model: the three activity boxes of the IDEF0 are transformed into the transitions represented as boxes in the CPN model and each ICOM flow is modeled as an arc leaving the output transition and connect to a place where tokens that represent the output data can reside and then an arc going from that place to the appropriate receiving transition. The transitions on the page are substitution transitions meaning that they are decomposed on a lower level page. Thus there is a one to one correspondence between the pages of the IDEF0 model and the CPN model.



Purpose: to describe EBA process in order to develop technology to support it and the DASEA CE
Viewpoint: Operational Assessors

Fig. 3.2 External System Diagram

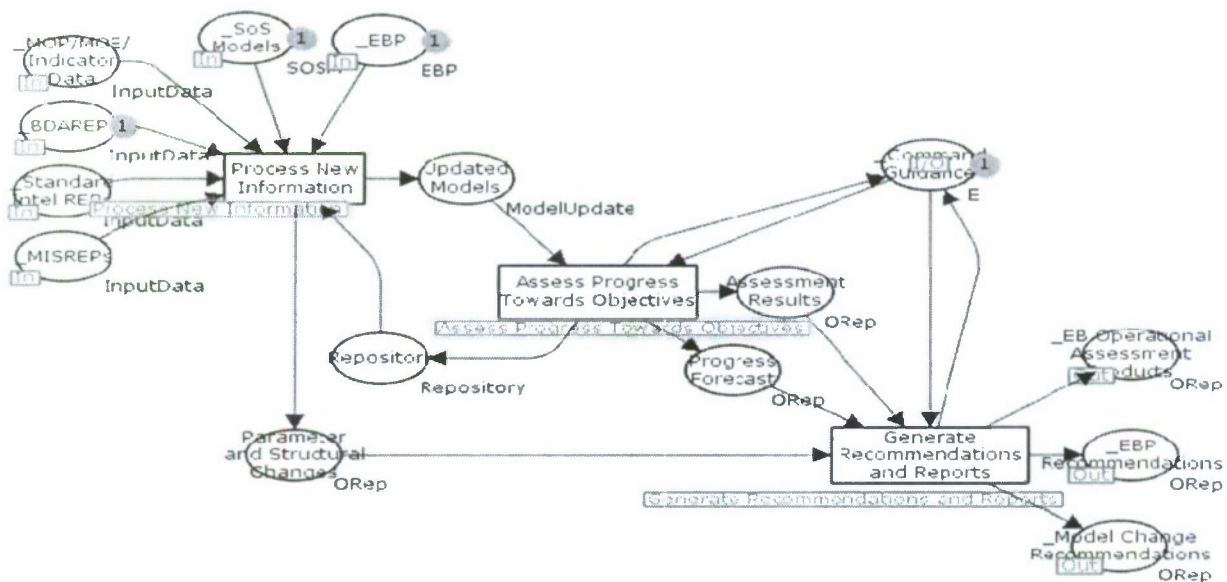


Fig. 3.3 A0 Page

Running this executable model in simulation with the various inputs verified that the process described in the Operational View indeed could produce the EBA products described in the Activity Model of the OV. This gave indication that the IDEF0 activity model was sound. This was an initial iteration of Loop 1 of Figure 1. However, the model was still at a high level of abstraction, and it was unknown what would happen if a team of assessors actually followed the operational concept.

Step 5 was to design a human-based experiment in which a small group of subject matter experts attempted to follow the process defined by the architecture using real input and control artifacts. To do this, a scenario was created including an EBP, a SOS Model, and a set of input messages. Particular attention was paid to the latter to evaluate the process through a set of progressively more complicated inputs. Three separate “threads” were designed, each expected to take about 4 hours to complete. All three threads used the same EBP and SOS Model, and each thread had enough content to allow the humans to be able to produce at least some of the output products, especially updates on the progress toward achieving commander’s intent. The first thread consisted of a set of messages whose content unambiguously matched the expected information found in the EBP and the SOS Model. The challenge was to match the content of the input information to the appropriate elements of the EBP and then produce the update to the assessment. The second thread contained the same inputs as the first thread, along with additional inputs that changed or updated the input data from the first thread. The goal was to create the situation where the humans had to aggregate or combine results to produce the output products. The third thread contained the same inputs as the first thread, but also contained additional inputs that did not exactly match the expected information contained in the EBP. This, the most challenging thread, was designed to determine how the humans would resolve such ambiguities.

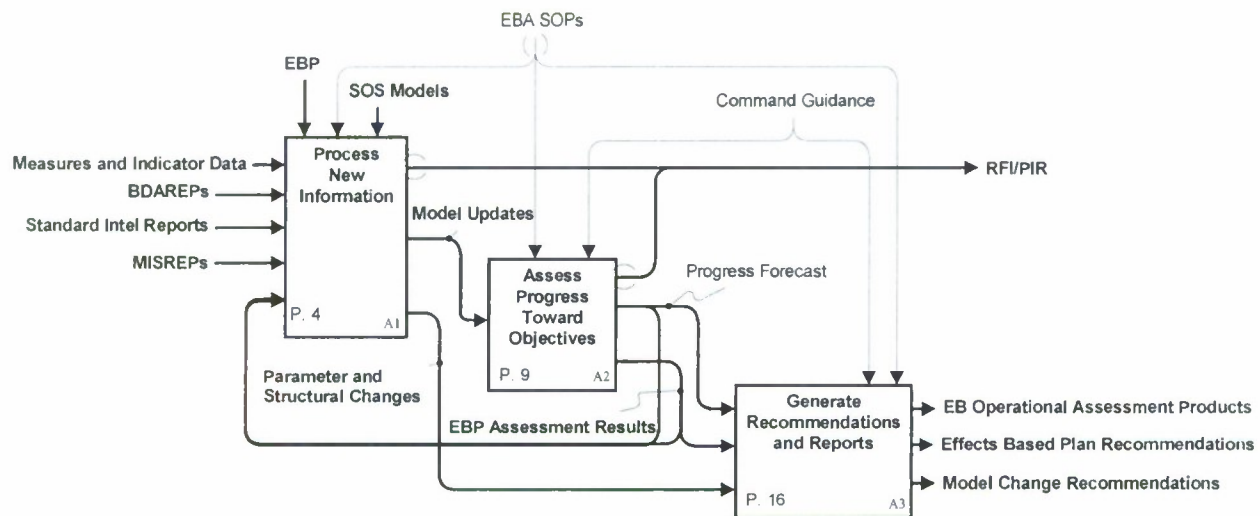


Fig. 3.4 Color Petri Net Executable Model, A0 Page

This human experiment took several weeks to design, three days to conduct, and several more days to analyze. Many comments and observations were made by the subject matter experts including suggestions for adjustments to the process that had been defined in the architecture. With the changes that were made on the fly during the threads, the humans were able to produce the key outputs including updates to the progress being made toward achieving commander’s intent. Thus this initial experiment demonstrated that it was possible to produce an

operational level EBA given the EBP, SOS Model, the process, and various sets of input information which was the hypothesis being evaluated.

In Step 6 the results of this human based experiment and the observations and suggestions for changes were used to refine the architecture and then the executable model (see Loops 3 & 4 of Figure 3.1). In this case, the refinements were small.

The human based experiment had provided valuable insight into the operational concept and the process defined in the architecture and resulted in some refinements of that architecture. Due to the time limitations, only a few threads through the process were evaluated. Therefore in Step 7 of the spiral the refined executable model was used to conduct a computational experiment over an entire range of inputs and thereby verify that the process and its OV description were sound and complete. The Colored Petri Net executable, synthesized from the refined architecture description, was used as the experimental test bed for this computational experiment. Two types of evaluation were accomplished with the executable model; (1) simulation using each and every legitimate type of input to review the process flow through the model and determining the types of output produced for each input and (2) state space analysis to identify all the combinations of sequences through the process and to identify potentially erroneous ones.

The focus of the computational experiment was on the input data sets. An analysis of the content of the input data indicated that there were a total of 180 valid types of inputs to this executable model. Each input represented a particular set of characteristics that can be contained in the data in any incoming message. It was decided to examine all 180 possible inputs using simulation. Multiple runs were made for each of the 180 legitimate input data types. Approximately 600 simulation runs were made with each run taking less than two seconds to complete.

As expected many of the input types could result in more than one set of output products. This is because the process allows for choices to be made as the process unfolds. Different choices result in different paths through the process and potentially different outputs for the same type of input. Overall, the simulation runs confirmed an analysis of the model indicating there are 11 different types of output products that can be generated depending on the nature of the input. These outputs include measures of effectiveness comparisons, progress toward commander's intent updates, target system analyses, EBP change recommendations, and various briefings associated with these outputs. Most inputs generated one or more of these outputs. A review of each input and the resultant output showed that in the final version of the executable, every legitimate input produced the desired outputs.

While each simulation run of the executable model shows *particular sequence or trajectory of processing for a given input*, state space analysis shows *all possible trajectories* for a given input. State space analysis [Kristensen et al., 1998] provides a detailed look at all possible sequences of states that can occur given a specific input. It can determine several important properties of the state space of a CPN model given a specific initial state. These include statistics on the state space such as the total number of states and transitions between states, and

liveness properties such as the number and identify of final states, and the number and identity of CPN transitions that never fired in any sequence. The analysis determines boundedness properties that identify the minimum and maximum number and type of tokens that occur in each place. It also captures the marking (distribution of tokens) for any state including the final states so that these can be examined.

State Space analysis is used to see if it is possible for an input to generate an undesired sequence or output. Because State Space analysis can take more time than running simulations, there was concern that it would take too long to conduct State Space analysis on all 180 inputs. Fortunately, by reviewing the process using simulation, it was discovered that it was possible to group the inputs into sets that would result in the same behavior. This meant that only one example from each group needed to be examined using the state space analysis to check the complete behavior of the architecture. Five groups were identified. For groups 1 and 2 (the least complex) CPNTools® was able to generate completed State Spaces with no unexpected final states. The first group had a single final state and the second two final states. Examination of these final states indicated that the process had completed successfully

The state space analysis for the more complex groups 3, 4, and 5 resulted in state space explosion meaning the state space analysis algorithm did not generate a complete state space after a reasonable amount of time (30 minutes). To work around this problem the executable model was divided into two parts. The first part included only processes A1 and A2 representing the processing of the input data and the assessment analysis. The second part was A3 (Generate Recommendations and Reports) by itself. This partitioning of the model allowed the state space analysis algorithm to generate complete state spaces. First, the state space for the model that contained A1 and A2 was generated and its final state(s) were examined. The state space analysis for A1 and A2 for the most complex input had 42 states and two final states. These final states contained one and five data elements, respectively. These were used as the initial input to the model of A3 and the state space analysis was run in it. For the case of the five inputs, it was not possible to provide all inputs at once because state space explosion occurred. Instead, the state spaces for combinations of the five inputs were examined and showed that there are no unexpected or undesirable final states for even the most complex of inputs to the process.

The overall conclusion from the computational experiments was that the operational view of the architecture was sound and complete. It demonstrated the truth of the basic hypothesis although it did not prove that the hypothesis is always true.

With the soundness of the operational view established, Step 8 of the process was to develop a systems view of the architecture that can be used to guide the development of systems that support the OV processes so that the second spiral of the Critical Experiment can be conducted. The second spiral and later spirals will again rely on the use of the architecture based model driven experimentation approach. These more complex spirals will use a combination of humans and new systems that are based on the system view of the architecture. The ultimate goal is to produce technology and systems that can support a sound operational level EBA process.

3.4 Conclusions

The architecture-based model-driven experimentation methodology has been successful in guiding the Critical Experiment through a series of spirals. It appears to be particularly relevant to situations where operational concepts and requirements are ill structured and the desired capability is complex and unprecedented. It follows a layered approach starting with the most abstract description of the problem or capability and, through a succession of steps, a continual refinement of that description unfolds until a final design is developed and evaluated. It is important not to go into too much detail early in the process. It must be shown that the abstract concepts work before going into more detail. The development of the operational concept is critical in the process; without one the rest of the process cannot be undertaken. In addition, a clear demarcation of the boundary of the system or capability must be defined. The process relies on a combination of static and executable architecture models and human-based experiments whose results feed one another. The development of the architecture description requires a few key subject matter experts. The conversion to the executable model is a straight forward process, and new tools are being developed to support the automation of this process which will make this step even easier. The subsequent analysis of the executable model allows a rigorous and complete or nearly complete exploration of the set of input conditions that can be expected. The human based experiments are critical, but demand the most resources and are the most time consuming. They are critical because they provide a depth of understanding that cannot be obtained with the architecture models by themselves. While they yield valuable information, they generally only relate to a relatively small portion of what is potentially a large number of probable vignettes that will be faced by the real system that will ultimately be developed and fielded. The bottom line is that the use of executable models of the architecture in conjunction with the more traditional human-based experiments can provide greater insight into the complex interactions that exist in these systems and allow developers and operators to make informed choices early to guide the development and create new technologies that will effectively support the operators.

3.5 References

- H. A. H. Handley, Z. R. Zaidi, and A. H. Levis, "The use of Simulation Models in Model Driven Experimentation," *Systems Engineering*, 1999, Vol. 2, No. 2, pp. 108-128.
<http://wiki.daimi.au.dk/cpntools/cpntools.wiki>
- L.M. Kristensen, S. Christensen, K. Jensen: *The Practitioner's Guide to Coloured Petri Nets*. International Journal on Software Tools for Technology Transfer, 2 (1998), Springer Verlag, 98-132.
- L. W. Wagenhals, I. Shin, D. Kim and A. H. Levis. "C4ISR Architectures II: A Structured Analysis Approach for Architecture Design," *Systems Engineering*, Vol. 3, No. 4, Fall 2000
- Lee W. Wagenhals, Sajjad Haider, and A. H. Levis, "Synthesizing Executable Models of Object Oriented Architectures," *Systems Engineering*, Vol. 6, No. 4, 2003.

A. H. Levis and W. S. Vaughan, "Model Driven Experimentation," *Systems Engineering*, 1999,
Vol. 2, No. 2,

SECTION 4

Course of Action Analysis in a Cultural Landscape Using Influence Nets

Lee W. Wagenhals and Alexander H. Levis

4.1 Introduction

Two challenges are addressed: (a) the need to understand how actions taken by the military or other elements of national power may affect the behavior of a society that includes an adversary and non adversarial elements, and (b) the need to be able to capture and document data and knowledge about the cultural landscape of an area of operations that can be used to support the understanding of the key issues, beliefs, and reasoning concepts of the local culture so that individuals that are new to the region can quickly assimilate this knowledge and understanding.

The first challenge relates to capabilities that enable the analysis needed to conduct focused effects based planning and effects based operations. Models to support Effects Based Operations developed to date relate actions to effects on the adversary [1]. Such models can be quite effective in informing the comparison of alternative courses of action provided the relationships between potential actions and the effects are well understood. This depends on the ability to model an adversary's intent and his reactions and identifying his vulnerable points of influence. But as the nature of Blue's military operations goes well beyond the traditional major combat operations, there is the need to anticipate the effects of actions not only on the adversary (Red), but also on the local population which may support or oppose that adversary. Such support may depend in part on the actions taken by Blue.

The second challenge involves the need for new personnel to rapidly assimilate the local knowledge needed to analyze the local situation and to analyze and formulate the effects based plans and operations. Data about a culture exists in many forms and from many sources including historical reference documents, observations and reports by intelligence analysts, and unclassified (and unverified) sources such as the internet. The data is often incomplete and partially incorrect and includes contradictions and inconsistencies. Analysts, particularly those new to an area of operation, who are responsible for formulating courses of action, are hard pressed to quickly develop the necessary understanding of the cultural factors that will affect the behavior of the adversary and the society in which it is embedded.

A case study based on a particular province in Iraq has been used to examine and test an approach to these challenges. The case study demonstrated the development of a model of an adversary and the culture that can be used to assess various courses of action designed to achieve several high level effects. A timed influence net (TIN) modeling technique was used that enables analysts to create executable (probabilistic) models based on knowledge about the cultural environment that link potential actions with their timing to effects. Such models capture the rationale for courses of action and explain how various actions can achieve effects. Given a set of potential actions, the model is then used to determine the course of action that maximizes the likelihood of achieving desired effects as a function of time.

The rest of this section is organized as follows. Section 4.2 gives a brief formal description of a TIN and describes a process that can be used for course of action analysis. Section 4.3 describes the case study and how a specific objective along with detailed data about the cultural environment was used to create and analyze a TIN. The rationale and thought processes that were used to determine the content of the TIN are described first, followed by a description of how the TIN was used in a layered analysis process to examine various courses of action to determine their impact on the overall effects over time. Section 4 provides some observations and comments.

4.2 Timed Influence Nets

Several modeling techniques are used to relate actions to effects. With respect to effects on physical systems, engineering or physics based models have been developed that can predict the impact of various actions on systems and assess their vulnerabilities. When it comes to the cognitive belief and reasoning domain, engineering models are much less appropriate. The purpose of affecting the physical systems is to convince the leadership of an adversary to change its behavior, that is, to make decisions that it would not otherwise make. However, when an adversary is imbedded within a culture and depends upon elements of that culture for support, the effects of physical actions may influence not only the adversary, but the individuals and organizations within the culture that can choose to support, be neutral, or oppose the adversary. Thus, the effects on the physical systems influence the beliefs and the decision making of the adversary and the cultural environment in which the adversary operates. Because of the subjective nature of belief and reasoning, probabilistic modeling techniques such as Bayesian Nets and their influence net cousin have been applied to these types of problems. Models created using these techniques can relate actions to effects through probabilistic cause and effect relationships. Such probabilistic modeling techniques can be used to analyze how the actions affect the beliefs and decisions by the adversary.

Influence Nets (IN) and their Timed Influence Nets (TIN) extension are abstractions of Probabilistic Belief Nets also called Bayesian Networks (BN) [2, 3], the popular tool among the Artificial Intelligence community for modeling uncertainty. BNs and TINs use a graph theoretic representation that shows the relationships between random variables. These random variables

can represent various elements of a situation that can be described in a declarative statement, e.g., X happened, Y likes Z, etc.

Influence Nets are Directed Acyclic Graphs where nodes in the graph represent random variables, while the edges between pairs of variables represent causal relationships. While mathematically Influence Nets are similar to Bayesian Networks, there are some key differences. BNs suffer from the often intractable task of knowledge elicitation of conditional probabilities. To overcome this limitation, INs use CAST Logic [4, 5], a variant of Noisy-OR [6, 7], as a knowledge acquisition interface for eliciting conditional probability tables. This logic simplifies knowledge elicitation by reducing the number of parameters that must be provided. INs are appropriate for modeling situations in which the estimate of the conditional probability is subjective, e.g., when modeling potential human reactions and beliefs, and when subject matter experts find it difficult to fully specify all conditional probability values.

The modeling of the causal relationships in TINs is accomplished by creating a series of cause and effect relationships between some desired effects and the set of actions that might impact their occurrence in the form of an acyclic graph. The actionable events in a TIN are drawn as root nodes (nodes without incoming edges). Generally, desired effects, or objectives the decision maker is interested in, are modeled as leaf nodes (nodes without outgoing edges). In some cases, internal nodes are also effects of interest. Typically, the root nodes are drawn as rectangles while the non-root nodes are drawn as rounded rectangles. Figure 1 shows a partially specified TIN. Nodes B and E represent the actionable events (root nodes) while node C represents the objective node (leaf node). The directed edge with an arrowhead between two nodes shows the parent node promoting the chances of a child node being true, while the roundhead edge shows the parent node inhibiting the chances of a child node being true. The inscription associated with each arc shows the corresponding time delay it takes for a parent node to influence a child node. For instance, event B, in Fig. 1, influences the occurrence of event A after 5 time units.

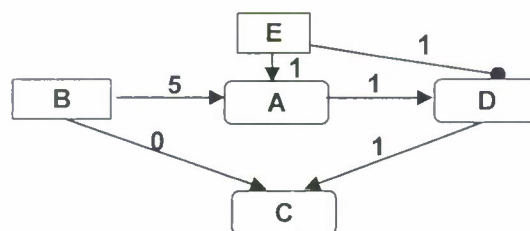


Fig 4.1. An Example Timed Influence Net (TIN)

Formally, a TIN is described by the following definition.

Definition 4.1: Timed Influence Net (TIN)

A TIN is a tuple $(V, E, C, B, D_E, D_V, A)$ where

V : set of Nodes,

E : set of Edges,

C represents causal strengths:

$E \rightarrow \{ (h, g) \text{ such that } -1 < h, g < 1 \}$,

B represents Baseline / Prior probability: $V \rightarrow [0,1]$,

D_E represents Delays on Edges: $E \rightarrow Z^+$ (where Z^+ represent the set of positive integers),

D_V represents Delays on Nodes: $V \rightarrow Z^+$, and

A (input scenario) represents the probabilities associated with the state of actions and the time associated with them.

$A: R \rightarrow \{ ([p_1, p_2, \dots, p_n], [t_{11}, t_{12}], [t_{21}, t_{22}], \dots, [t_{n1}, t_{n2}]) \}$

such that $p_i = [0, 1]$, $t_{ij} \rightarrow Z^+$ and $t_{i1} \leq t_{i2}$, $\forall i = 1, 2, \dots, n$ and $j = 1, 2$ where $R \subset V$

(where Z^+ represent the set of nonzero positive integers)

The purpose of building a TIN is to evaluate and compare the performance of alternative courses of actions. The impact of a selected course of action on the desired effects is analyzed with the help of a probability profile. Consider the TIN shown in Fig. 1. Suppose the following input scenario is decided: actions B and E are taken at times 1 and 7, respectively. Because of the propagation delay associated with each arc, the influences of these actions impact event C over a period of time. As a result, the probability of C changes at different time instants. A probability profile draws these probabilities against the corresponding time line. The probability profile of event C is shown in Fig. 4.2.

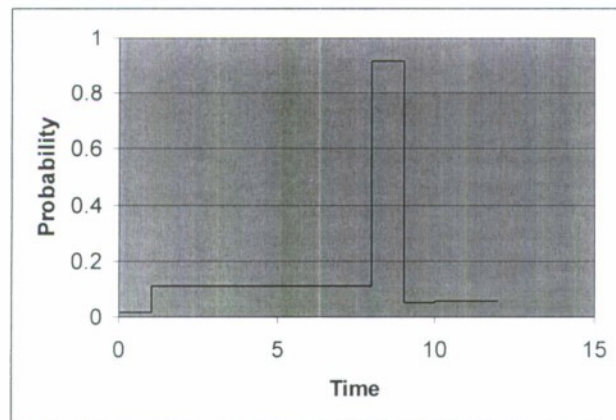


Fig 4.2. Probability Profile for Node C

To construct and use a TIN to support effects based operations, the following process has been defined.

1. Determine the set of desired and undesired effects expressing each as a declarative statement that can be either true or false. For each effect, define one or more observable indicators that the effect has or has not occurred.
2. Build an IN that links, through cause and effect relationships, potential actions to the desired and undesired effects. Note that this may require defining additional intermediate effects and their indicators.
3. Use the IN to compare different sets of actions in terms of the probability of achieving the desired effects and not causing the undesired effects.
4. Transform the IN to a TIN by incorporating temporal information about the time the potential actions will occur and the delays associated with each of the arcs and nodes.
5. Use the TIN to experiment with different timings for the actions to identify the “best” COA based on the probability profiles that each candidate generates. Determine the time windows when observation assets may be able to observe key indicators so that assessment of progress can be made during COA execution.
6. Create a detailed execution plan to use the resources needed to carry out the COA and collect the information on the indicators.
7. Use the indicator data to assess progress toward achieving the desired effects.
8. Repeat steps 2 (or in some cases 1) through 7 as new understanding of the situation is obtained.

In building the IN, the modeler must assign values to the pair of parameters that show the causal strength (usually denoted as g and h values) for each directed link that connects pairs of nodes. Each non-root node has an associated baseline probability that must be assigned by the modeler (or left at the default value of 0.5). It represents the probability that the random variable will be true in the absence of all modeled influences or causes. The CAST logic is based on a heuristic that uses these quantified relationships and the baseline parameter to compute the conditional probability matrix for each non-root node. Finally, each root node is given a prior probability, which is the initial probability that the random variable associated with the node (usually a potential action) is true.

When the modeler converts the IN into a TIN (step 4), each link is assigned a corresponding delay d (where $d \geq 0$) that represents the communication delay. Each node has a corresponding delay e (where $e \geq 0$) that represents the information processing delay. A pair (p, t) is assigned to each root node, where p is a list of real numbers representing probability values. For each probability value, a corresponding time interval is defined in t . In general, (p, t) is defined as

$$([p_1, p_2, \dots, p_n], [[t_{11}, t_{12}], [t_{21}, t_{22}], \dots, [t_{n1}, t_{n2}]]),$$

where $t_{i1} < t_{i2}$ and $t_{ij} > 0 \forall i = 1, 2, \dots, n$ and $j = 1, 2$

The last item is referred to as an input scenario, or sometimes (informally) as a course of action.

To analyze the TIN (Step 5), the analyst selects the nodes that represent the effects of interest and generates probability profiles for these nodes. The probability profiles for different courses of action can then be compared.

4.3 Case Study

A case study was used to demonstrate a capability to address the two challenges described in the introduction. The challenge was to create (demonstrate) a capability to allow rotating and in-country forces to easily and quickly access data and knowledge about the cultural landscape of their area of operations that can be used to support their understanding of the key issues, beliefs, and reasoning concepts of the local culture. The specific need that the case study addressed was stated as follows: given a military objective and a set of desired effects derived from statements of commander's intent, develop and analyze alternative courses of actions (COAs) that will cause those desired effects to occur and thus achieve the military objective. The use of TINs was the approach taken. Specifically, the case study demonstrated the use of a TIN tool called Pythia that has been developed at George Mason University. This demonstrated the use of the tool to create knowledge about an adversary and the population that potentially supports or resists that adversary and the use of the TIN to analyze various COAs.

A scenario was chosen based on the problem of suppressing the use of Improvised Explosive Devices (IEDs) in a specific province of Iraq, denoted as province D. Specifically, it is assumed that IED incidents have increased along two main east-west routes between the capital town C of the province and a neighboring country M. Both roads are historically significant smuggling routes.

There were hundreds of documents about Iraq in general and D province in particular that were reviewed to get a better understanding of the situation. The province includes substantial fractions of Kurdish, Shia, and Sunni populations as well as other minorities. It was noted that the northern route was in the predominantly Kurdish region and the southern route was in a predominantly Shia region. A dynamic tension existed between these regions particularly with regard to the flow of commerce because of the revenue the flow generates. It was noted that some revenue was legitimate, but a significant amount was not and was considered covert. Increased IEDs in one region tended to suppress the trade flow in that region and caused the flow to shift to the other. Consequently, each region would prefer to have the IEDs suppressed in its region, but not necessarily in the neighboring region. The IED perpetrators needed support from the local and regional populations as well as outside help to carry out their attacks. The support was needed for recruiting various individuals to help manufacture the IEDs and to carry out the operations necessary to plant them and set them off. It was postulated that improving the local economy and the quality of the infrastructure services would reduce the local and regional

support to the insurgents. Of course this required effective governance and willingness on the part of the workers to repair and maintain the infrastructure that in turn requires protection by the Iraqi security and coalition forces.

With this basic understanding, the following steps were taken to create the TIN. First the overall key effects were determined to be 1) IED attacks are suppressed on routes A and B (note these were modeled as separate effects because it may be possible that only one of the routes may have the IED attacks suppressed), 2) Covert economic activity improves along each of the two routes. 3) Overall Overt economic activity increases in the region. 4) Insurgent fires are suppressed, 5) Local support for the insurgents exist and 6) Regional support for the insurgents exists. Nodes for each of these effects were created in the Pythia TIN modeling tool. It was noted that suppression of IED attacks on one route could have an inverse effect on the covert economic activity on the other, but each could improve the overall overt economic activity. The suppression of the insurgent fires positively affected both covert and overt economic activity.

The next step was to identify the key coalition force (Blue) actions that would be evaluated as part of the potential overall COA. To be consistent with the level of model abstraction the follow high level actions were considered: Blue coalition forces (CF) exercise their standard Tactics, Techniques, and Procedures (TPPs) (including patrols, searches, presence operations, and the like). Blue Coalition Forces actively conduct surveillance operations. Blue CF actively conduct Information Operations. Blue CF continue to train the local Iraqi security forces and police. Blue CF broker meetings and discussions between various Iraqi factions (Green).

Of course, it is not possible to just connect these actions to the key effects, and therefore several other sub-models were constructed and then linked together to produce the final model. These models include a model of the process the insurgents must use to conduct IED operations, a sub-model for the infrastructure and economic activity, and a sub model of the political and ethno-religious activities. In addition, it was recognized that the region was being influenced by outside sources, so these also were added to the model.

The sub model of the insurgent IED activities was based on the concept of how the insurgents develop an IED capability. They must have the IEDs, the personnel to carry out the IED operation, the communication systems to coordinate the operation and the surveillance capability to determine where to place the IED and when to set it off. Each of these in turn requires additional activities. For example, the personnel must be trained and in order to get the personnel they must be recruited. The IEDs must be manufactured, and this requires material and expertise. Furthermore, the insurgents must be motivated to use their capability. Much of this capability relies on support for the local and regional population and funding and material from outside sources. The nodes and the directed links between them were added to the TIN model to reflect the Insurgents' Activities.

The economic and infrastructure sub-model included nodes for each of the main essential services: water, electricity, sewage, health, and education. It also included financial institutions (banks, etc.) and economic activities such as commerce and retail sales of goods. The nodes for

the economic and infrastructure aspect of the situation were linked to the local and regional support as well as to the overall effect on the overt economic activity.

Of course, the economic and infrastructure services will not function properly without the support of the Political and Ethno-Religious entities in the region. Thus a sub-model for these factors was also included. To do this, three facets of the region were considered: the religious activities including Shia, Sunni, and Kurdish (who are either Shia or Sunni) groups, political party activities (Shia, Sunni, and Kurdish), and the Shia, Sunni, and Kurdish activities within the government structure including the civil service and the police and law enforcement institutions. The nodes for all of these activities were created and appropriate links were created between them. Links were also created to other nodes in the model such as local and regional support of the insurgents, economic activity and infrastructure development.

Finally, the outside influences were added to the model. These include external support for the insurgents, anti-coalition influences from neighboring countries, and external financial support for the local government and the commercial enterprises of the region. All of these nodes were modeled as actions nodes with no input links. With this model design, analysts could experiment with the effects of different levels of external support, both positive and negative, on the overall outcomes and effects.

The complete model is shown in Figure 4.3. The model has 62 nodes, including 16 nodes with no parents, and 155 links.

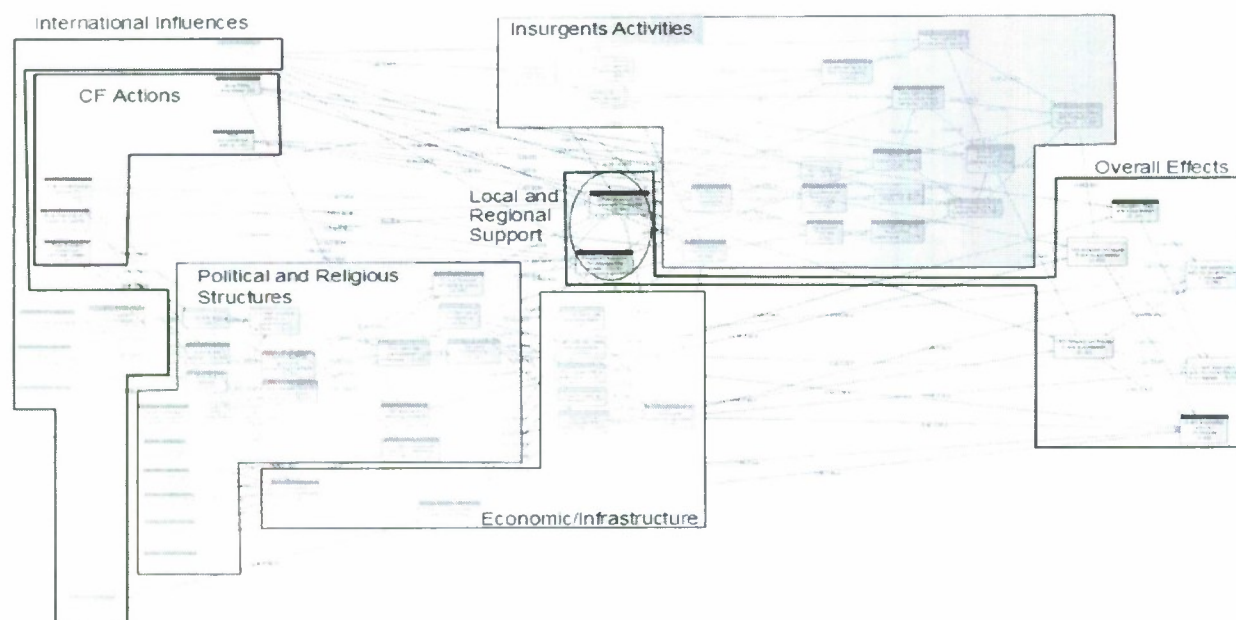


Fig. 4.3 Complete TIN Model

Once the structure of the models was completed, the next step was to assign the values to the parameters in the model. This was done in two steps. First, the strengths of the influences (the g and h parameters on each link) and the baseline probability of each node were selected. This may seem like a daunting task given the subjective nature of the problem and the number of links and nodes. However, TINs and the Pythia tool limit the choices that can be made for these parameters. For each link, the model determines the impact of a parent node on a child node first if the parent is true and then if the parent is false. The choices range from very strongly promoting (meaning nearly 100%), strong (quite likely, but not 100%), moderate (50% or greater, but less than strong), slight (greater than 0% but not likely), or no effect. The modeler can also select a similar set of inhibiting strengths ranging from very strongly inhibiting to no effect. The second set of parameters is the baseline probabilities of the node. These are set to a default value of 0.5 meaning that the probability of the node being true is 0.5 given no other influences or causes (we don't know). In many cases, the default value was selected.

At this point it is possible, if not prudent, to perform some analysis on the model to observe its behavior. We will describe this in detail shortly. The final step in creating the TIN model was to assign the temporal parameter values to the nodes and the links. The default value for these is 0. With all values set to 0 the model is identical to an ordinary Influence Net. The process for assigning the time delay values is similar to that for assigning the strengths of the influences and the baseline probabilities. For each link, the modeler determines how long it will take for the child node to respond to a change in the probability of the parent node. In some cases the change is instantaneous, so the default value of 0 is appropriate. In others, a time delay may be expected. Part of this process requires that the modeler establish the time scale that will be used in the model and thus what actual time length of one unit of delay is. Any unit of measure can be selected from seconds to days, weeks, months or even years. In this particular model each time delay unit was set to be one week. In setting the time delay of the arcs, it may also be useful to set the time delay of the nodes. Again the default value for this delay is 0. This delay represents processing delay. It reflects the concept that if there is a change in one or more of the parent nodes, once the child node realizes that the change has occurred, there may be some time delay before it processes this new input and changes its probability value.

Once the complete TIN was created, a validation of the model was undertaken. This was done by consulting with several subject matter experts who had been in the region and were familiar with the situation. Each node and link was checked to see if the node and the relationships to and from that node made sense. In short, we were confirming that the overall structure of the model made sense. Several suggestions were made and the changes were incorporated. Once the structure had been vetted, then the parameters were checked. This was done link by link and node by node. First the strengths of the influences were checked, then the baseline probabilities, and finally the time delays.

Once the TIN model was finished and validated, two levels of analysis were accomplished to demonstrate the utility of the approach. The first level is the logical level. This can be done

without using the parameters because it only requires the structure of the model. At this level of analysis the model shows the complex causal and influencing interrelationships between Blue CF, the external influence, the religious and political factions, the adversary (Red), and the local and regional population (Green). This particular model shows that while Blue CF has some leverage, there are many other outside influences that also can affect the outcome of any actions that Blue may take. The model identifies these influences and how they may help inhibit the progress that is made as a result of Blue CF actions. Furthermore, the model shows relationships between the actions and activities of major religious and ethnic groups and effects on government activities (police, judiciary, public works and service, etc.). It shows the impact of the adequacy of government and public services on support of the insurgency. It captures the IED development, planning, and employment processes and the impact of the other activities, the status of public services, and coalition interventions on those processes. Finally the model captures interaction of IED attack suppression on two major trade routes (suppressing one route increases attacks on the other). In short, the model has captured Blue's understanding of a very complex situation and can help articulate concepts and concerns involved in COA analysis and selection.

The second level of analysis involves the behavior of the model. It is divided into a static quantitative and a dynamic temporal analysis. The static quantitative analysis requires the structure of the model and the non temporal parameters to be set. The temporal, time delay parameters should be set to the default value of 0. This analysis enables one to compare COAs based on the end result of taking the actions in the COA. In the Province D model, four major COAs were assessed as shown in Fig. 4.4. This table has four parts, an Action stub in the upper left corner, the Action or COA matrix to the right of the Action stub, an Effects stub below the Action stub, and the Effects matrix adjacent to the Effects stub. In the COA matrix, the set of COAs that have been evaluated are listed with an X showing the actions that comprise the COA. The Effects matrix shows the corresponding effects as the probability of each effect.

Actions	Situation (COA) 1	Situation (COA) 2	Situation (COA) 3	Situation (COA) 4
International Interference	X	X	X	X
External Financial Support		X	X	X
CF TTPs and Surveillance		X	X	X
CF IO, training, brokering			X	X
Iraqi political and religious group participation				X
EFFECTS				
Local and Region Support for Insurgents Exists	0.97	0.92	0.26/0.36	0.22/0.14
IED Attacks Suppressed on Route A / B	0.17/0.15	0.31/0.34	0.67/0.68	0.85/0.74
Insurgent's fires suppressed	0.14	0.65	0.9	0.93
Public services adequate	0.12	0.39	0.39	0.55
Overt Economic Activity Increasing	0.02	0.08	0.31	0.89
Covert Economic Activity Increasing along routes A and B	0.37	0.50	0.56	0.57

Fig. 4.4 Static Quantitative COA Comparison

COA 1 was a baseline case in which only international interference and support to the insurgency occurs. There is no action from the Blue CF, no external financial support to the infrastructure and the economy, and the religious and political factions are not participating in the governance of the area. The overall effects are shown in the lower part of the matrix. The results for this COA are very poor. There is support for the insurgency and it is very unlikely that the IED attacks will be suppressed on either route. With an ineffective local government, the basic services are inadequate which encourages the support to the insurgency and there is little chance for economic increase.

COA 2 represents the case where external financial support is provided and the coalition forces are active both in presence operations and in conducting surveillance. However, Information Operations, training of Iraqi forces and workers, and brokering of meetings and agreement between Iraqi factions are not occurring. In addition, the political and religious groups are not participating in positive governance and support to civil service. In this case, there is some improvement compared to COA 1, but still there are many problems. Local support for the insurgents is still very strong, although there is some suppression of the IED attacks and insurgent fires due to the activities of the coalition forces. As a result there is some improvement in public services and an increase in covert and overt economic activity, due in part to the reduction in IED attacks and insurgent fires.

The third COA contains all of the actions of COA 2 plus the addition of coalition force information operations, training of Iraqi security and police forces as well as civilian infrastructure operations and significant brokering of meetings and agreements between the various Iraqi agencies and factions. The result is a significant improvement in the suppression of the IED attacks and insurgent fires due to the improved capabilities of the Iraqi security and police forces and the significant drop in the local and regional support of the insurgents. There is also a significant improvement in the covert and overt economic activity. However, there is little change in the adequacy of the public services, due primarily to the lack of effective participation of the Iraqi governance function.

The last COA has all actions occurring. In addition to the activities of the previous three COAs, COA 4 includes the active participation of the Iraqi religious and political groups in the governance activities. It results in the highest probabilities of achieving the desired effects. While there is still some likelihood of local and regional support for the insurgents (0.22 and 0.14, respectively), many of the IED attacks are suppressed as are the insurgent fires. The result is significant increases in overt economic activity and moderate increase in the covert economic activity. Public services are still only moderately adequate, with room for improvement.

While the static quantitative analysis provides a lot of insight into the potential results of various COAs, it does not address the questions of how long it will take for the results to unfold or what should the timing of the actions be. The dynamic temporal analysis can provide answers to these types of questions.

Having created the TIN model with the time delay information, it is possible to experiment with various COAs and input scenarios. Fig. 4.5 shows an example of COA and input scenarios that illustrate such an experiment. The second column of the Table in Fig. 5 shows a summary of the input nodes that were used in the experiment. They are divided into two types, those listed as Scenario and those listed as COA Actions. The scenario portion contains actions that may take place over which limited control is available. These set the context for the experiment. The second group contains the actions over which control exists, that is the selection of the actions and when to take them is a choice that can be made. The last column shows the scenario/action combinations that comprise the COA/Scenario to be examined. The column provides a list of ordered pairs for each Scenario Action or COA Action. Each pair provides a probability (of the action) and a time when that action starts. For example, the listing for the second scenario actions is [0.5, 0] [1.0, 1] which means that the probability of Country M and Country L interfering is 0.5 at the start of the scenario and changes to 1.0 at time = 1. In this analysis, time is measured in weeks.

The entries under the column labeled "COA 4a" mean that the scenario/under which the COA being tested is one in which there is immediate and full support for the insurgency (financial, material, and personnel) from international sources, and it is expected to exist throughout the scenario. The same is true for support from Country S. Countries M and L are modeled with the probability of providing support at 0.5 initially, but it immediately increases to 1.0 at week 1. All of the COA actions are assumed to not have occurred at the start of the scenario, thus the first entry of each is [0, 0]. The coalition force (Blue) actions start at week 1 with a probability of 1.0, meaning that all of the elements of Blue actions start at the beginning. With regard to religious activities, the Kurds begin at week 1 with probability 1.0. The Shia and Sunni have a probability of 0.5 starting at week 10 and then increase to 1.0, becoming fully engaged at week 20. In terms of political activity, the Kurds and Shia become fully active at week 1. The Shia become more likely to be active at week 10, fully active at week 20, then become less likely to be active at week 30 (probability 0.5) and then become fully active again at week 40. Finally, the External Financial support begins at week 26.

	Action	COA 4a: List [p, t]
Scenario	Int'l Support to Insurgents	[1.0, 0]
Actions	Interference by countries M and L	[0.5, 0], [1.0, 1]
	Interference by country S	[1.0, 0]
COA	Blue TTPs activated	[0, 0], [1.0, 1]
Actions	Blue Surveillance, IO, Training, Brokering	[0, 0], [1.0, 1]
	Shia and Sunni Religious Activity	[0, 0], [0.5, 10], [1.0, 20]
	Kurd Religious Activity	[0, 0], [1.0, 1]
	Kurd and Shia Political Activity	[0, 0], [1.0, 1]
	Sunni Political Activity	[0, 0], [1.0, 20], [0.5, 30], [1.0, 40]
	International Investment	[0, 0], [1.0, 26]

Fig. 4.5 Dynamic Temporal Analysis Input

To see what the effect of this input scenario on several key effects, the model is executed and the probabilities of the key effects as a function of time are plotted as shown in Fig. 4.6. In the figure, the probability profiles of four effects are shown: IEDs are suppressed on Routes A and B and Local and Regional support for the Insurgents exists. Figure 4.6 shows that the probability of suppression of the IED attacks on the two routes increases significantly under this scenario. This means that the number of IED attacks should decrease, more on Route A than on Route B. The improvement can be expected to occur more rapidly along Route A than along Route B by about 35 weeks or 8 months. Route A is the northern route that is controlled by the Kurds and Route B is the southern route controlled by the Shia and Sunni. This can be attributed to the rapid and steadfast political and religious activities of the Kurds as opposed to the more erratic activities of the others as modeled in the input scenario (Fig. 4.5). Also note that it is expected to take 80 to 100 weeks (nearly 2 years) for the full effect to occur. Figure 4.6 also shows a significant decline in support for the insurgents both by the local and the regional populace with the local support decreasing more as the situation with respect to governance and services improves.

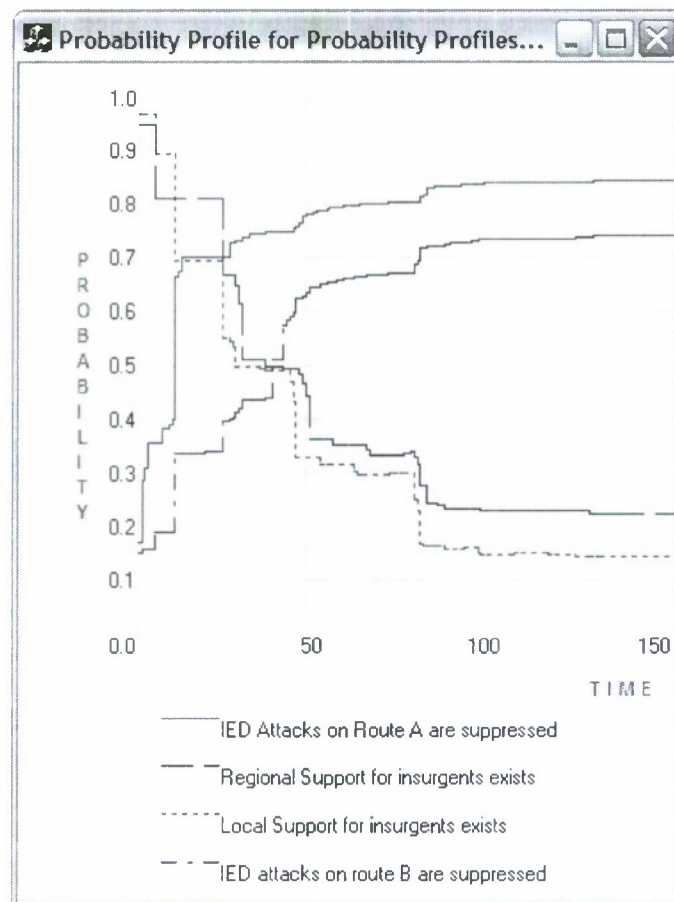


Fig. 4.6 Probability Profiles of Scenario (COA) of Fig. 4.5

Of course it is possible to examine the behavior of any of the nodes in the model, by plotting their probability profiles. This can increase the understanding of the complex interactions and dependencies that in the situation that have been expressed in the TIN model. The TIN model provides a mechanism to experiment with many different scenarios and COAs. Questions like what will happen if some of the Blue CF actions are delayed or what will happen if the Shia or Sunni decide not to participate after some period of time can be explored. By creating plots of the probability profile of key effects under different scenarios, it is possible to explore the differences in expected outcomes under different scenarios. This can be illustrated by changing the input scenario. Suppose that it is believed to be possible to get other countries or external organizations to reduce the support to the insurgents by some means, for example diplomatic or military action. It is postulated that we could reduce the likelihood of such support to about 50% but it will take 6 months to do this. The results can be modeled by changing the input scenario of Fig. 4.5. In this case the first line of Fig. 4.5 is changed from [1.0, 0] to [1.0, 0] [0.5, 26]. All of the other inputs remain the same. Fig. 4.7 shows a comparison of effect of this change on the suppression on IED attacks along Route B. The reduction in international support for the insurgents at week 26 can cause a significant improvement in the suppression of the IED attacks along Route B (and a corresponding improvement along Route A, not shown). The improvement begins about 6 months after the reduction in international support or about 1 year into the scenario. Thus, decision makers may wish to pursue this option.

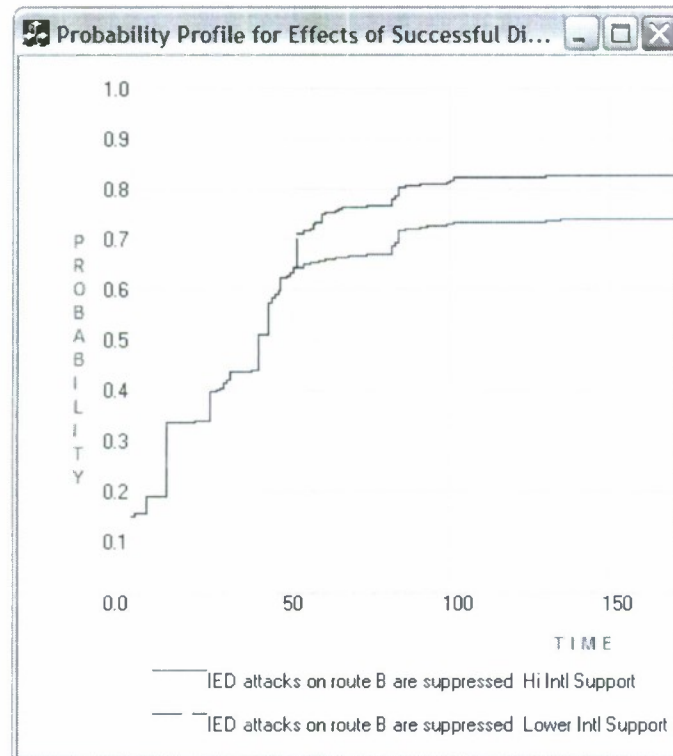


Fig. 4.7 Comparison of the Effect of Different Scenarios

4.4 Observations and Comments

Creating TIN models of situations appears to help address the two challenges described in the beginning of this paper. It provides a representation of knowledge about a situation that is derived from an understanding of the capabilities of an adversary and the interactions and dependencies of that adversary with the local and regional social, religious, and economic condition. Once created, the TIN model can be used to conduct computational experiments with different scenarios and COAs. In a sense, it provides a mechanism to assess various COAs based upon comparisons of the change in the probability of key effects over time.

It is important to emphasize that the purpose of these models is to assist analysts in understanding the potential interactions that can take place in a region based on actions taken by one or perhaps many parties. It is not appropriate to say that these models are predictive. They are more like weather forecasts, which help us to make decisions, but are rarely 100% accurate and are sometimes wrong. To help deal with this uncertainty, weather forecasts are continually updated and changed as new data becomes available from the many sensors that make a variety of observations in many locations. Since these models cannot be validated formally, the appropriate concept is that of credibility. Credibility is a measure of trust in the model that is developed over time through successive use and comparison of the insights developed through the model and the occurrence of actual events and resulting effects.

We believe that the techniques described in this paper can make an important contribution to a variety of communities that need to evaluate complex situations to help make decisions about actions they may take to achieve effects and avoid undesired consequences. The approach offers at least three levels of analysis, a qualitative evaluation of the situation based on the graph that shows the cause and effect relationships that may exist in the environment, and two levels of quantitative evaluation. The first level of quantitative analysis is static, and shows, a coarse way, what the likelihood of different effects occurring are given different sets of actions. The second quantitative level is dynamic, and shows how the scenario may play out over time. The relevant aspect is that the approach allows the inclusion of diplomatic, information, military, and economic (DIME) instruments and highlights their cumulative effects.

This modeling approach can provide analysts with a rich vehicle for explanation and computational experimentation with COAs so that important recommendations can be made to the decision makers. The models can be used to illustrate areas of risk including undesired effects, and risks associated *with the amount of time* it will take to achieve desired effects. It should also be noted that these models are not likely to be created on a one time basis. It can be expected that the understanding of the situation will continue to evolve requiring updates or even new models to be created. Perhaps the best contribution is that the technique offers a standard way to analyze and describe very complex situations.

ACKNOWLEDGMENT The invaluable advice that Dr. Susan Numrich, Institute for Defense Analyses, provided throughout the conduct of this work is gratefully acknowledged.

4.5 References

- [1] L.W. Wagenhals, A. H. Levis, and Maris McCrabb, "Effects Based Operations: a Historical Perspective for a Way Ahead," Proc. 8th Int'l Command and Control Research and Technology Symposium, NDU, June 2003
- [2] Jensen, F. V., *Bayesian Networks and Decision Graphs*, Springer-Verlag, 2001.
- [3] Neapolitan, R. E., *Learning Bayesian Networks*, Prentice Hall, 2003.
- [4] Chang, K. C., Lehner, P. E., Levis, A. H., Zaidi, S. A. K., and Zhao, X., "On Causal Influence Logic", Technical Report, George Mason University, Center of Excellence for C3I, Dec. 1994.
- [5] Rosen, J. A., and Smith, W. L., "Influence Net Modeling with Causal Strengths: An Evolutionary Approach", Proceedings of the Command and Control Research and Technology Symposium, Naval Post Graduate School, Monterey CA, Jun. 1996.
- [6] Agosta, J. M, "Conditional Inter-Causally Independent Node Distributions, a Property of Noisy-OR Models", In Proceedings of the 7th Conference on Uncertainty in Artificial Intelligence, 1991.
- [7] Drudzel, M. J., and Henrion, M., "Intercausal Reasoning with Uninstantiated Ancestor Nodes", In the Proceedings of the 9th Conference on Uncertainty in Artificial Intelligence, 1993.

SECTION 5

Service Oriented Architectures, the DoD Architecture Framework 1.5, and Executable Architectures

Lee W. Wagenhals and Alexander H. Levis

5.1. Introduction

The Department of Defense (DoD) Net Centric Warfare (NCW) concept is key to the transformation of DoD capabilities in the information age. Developing both operational concepts and systems that support those concepts based on ubiquitous information and data sharing across traditional boundaries is at the heart of NCW and its enabler called Net Centric Operations (NCO). This implies a shift from platform orientation based on tightly coupled or large-scale monolithic systems to a spectrum of integration techniques that include loosely coupled systems-of-systems. The DoD views architectures as the mechanism for designing solutions for this transformation, and the use of Service Oriented Architectures (SOAs) has been selected as an approach for achieving many of the goals of this transformation.

To support this transformation, DoD has issued a major revision in its DoD Architecture Framework that enables the inclusion of services-based architectures. The result of this transformation and DoDAF decisions is that there is much to analyze and many choices to be made. The behavior and performance (e.g., quality of service) of the information sharing approaches supported by SOA have not been proven within the DoD environment. It is well known that the dynamic behavior of these systems is complex. Any engineering approach, including those that are architecture based, requires an ability to determine stakeholder needs (requirements) and techniques for evaluating potential solutions based on the projected capabilities of the design to meet those requirements. DoDAF 1.5 as an architecture description specification relies on static pictures (diagrams) and tables. These are capable of describing the behavior of the architecture only in a limited way. If architectures are the mechanism for designing solutions and the solutions are complex, there is a strong need for architecture evaluation techniques that go beyond static diagrams and examine behavior and performance in detail. Converting the architecture description into an executable model and applying evaluation processes to that model can support this expanding analysis and evaluation need.

By intent, the DoDAF does not specify or provide a process for designing or evaluating architectures. The methodologies, tools, techniques, and processes for design and evaluation need to be selected and executed by the practitioners that will be creating and analyzing

architectures to support the transformation concepts. Such processes and techniques have been developed, but the majority of the DoD community effort has focused primarily on the creation of architecture descriptions without rigorous behavior and performance evaluation, because that is what has been the required deliverable in most cases. Explicit evaluation processes are much less evident.

The objective of this paper is to describe and illustrate the processes and techniques that can support end-to-end design, analysis, and evaluation of architecture descriptions, particularly in light of the shift in direction from designing large scale, tightly coupled or monolithic systems to the more loosely coupled constructs needed to support the NCO vision. Section 5.2 summarizes the background and challenges facing architectures including the motivation for architectures and a discussion of some of the issues with services, SOA, integration, and coupling. Section 5.3 introduces an overall end-to-end process for generating architecture descriptions and supporting evaluations using executable models. Three sub processes are described. The first is a process for creating DoDAF compliant architecture descriptions that contain all the necessary information needed to derive the executable model from the architecture description. Two variants are discussed, one using object orientation with UML as the architecture description language, and the other using Structured Analysis. Second, the techniques and processes for converting the architecture description into an executable model are discussed with Colored Petri Nets serving as the mathematical framework for the executable model. Finally, analysis and evaluation needs and the techniques to address them using the executable model are described. Section 5.4 describes a recently completed case study that was tailored to illustrate the processes for creating and analyzing a DoDAF 1.5 compliant architecture that incorporates NCO concepts. This case study gives a detailed description of how a DoDAF 1.5 compliant architecture can be created, the type of analysis that can be done based on that architecture description, the process for conversion of the architecture to an executable model, and a process for analysis of the architecture by using the executable model to address questions that cannot be answered by the architecture description alone. Section 5.5 concludes the paper with observations and challenges.

5.2. Background And Challenges

In 1998, the Department of Defense released and approved the Command, Control, Communications, and Computers, Intelligence, Surveillance, and Reconnaissance Architecture Framework (C4ISR AF) Version 2.0. Motivated by growing interoperability challenges, DoD viewed architectures as the means to analyze interoperability solutions rather than attempting to deal with detailed design descriptions which can change very rapidly. The goal was to standardize the manner in which DoD organizations represented the descriptions of architectures and to provide a common framework for coping with uncertainty, change, and complexity in requirements, missions, organizational structures (e.g., joint and coalition operations), and technology. Organizations were directed to create architectures to support the analysis of requirements and capabilities, budgetary needs, and acquisition plans and processes.

As military challenges expanded, the need to be able to respond to a variety of situations by assembling coalitions of different components that may be geographically dispersed became a major driver for a transformation of military capability. The notion of composing forces using a “plug and play” construct where component systems can plug into an Internet-like Global Information Grid (GIG) was postulated and NCW was established as the overarching concept. Net Centric Warfare was focused on generating combat power by networking the warfighting enterprise, and making essential information available to authenticated, authorized users when and where they need it. This ability is at the core of net-centricity and essential to achieving Net Centric Operations (NCO). Migrating to the NCO concept poses several challenges. It implies a shift in policy from a need to know to a need to share. It means carefully reexamining the tradition of building large scale monolithic system constructs that do excellent jobs in their own right, but pose challenges when an attempt is made to interoperate many of these systems. Developing more loosely coupled constructs is considered essential to the information sharing goals. To address these challenges, DoD developed and issued a Data Strategy and released the Net Centric Operations in Warfare (NCOW) Reference Model Version 1.1. The NCOW RM supports the concept of services and SOA as a means for achieving the goals of NCO. As the NCOW RM was being developed, DoD worked to update the C4ISR Architecture Framework Version 2.0 by releasing the DoD Architecture Framework (DoD ADF) Version 1.0 in 2004. This version made only slight changes to the basic construct of its predecessor. However, in April 2007, DoD released the DoDAF Version 1.5 which included important changes that respond to the transformation to NCO.

The DoDAF provides the guidance and rules for developing, representing, and understanding architectures based on a common denominator across DoD, Joint, and multinational boundaries. The DoDAF is intended to ensure that architecture descriptions can be compared and related across programs, mission areas, and the enterprise. While the DoDAF provides a standardized format for describing architectures, it does not provide a procedure for developing the artifacts and data that are used in the description. DoDAF 1.5 is a transitional version that responds to the DoD’s migration towards NCW. It applies essential net-centric concepts in transforming the DoDAF and acknowledges that the advances in enabling technologies—such as services within a SOA—are fundamental to realizing the Department’s Net-Centric Vision. DoDAF 1.5 maintains the standard views of its predecessors, the Operational, System, and Technical Standards Views, so as to maintain backward compatibility with the DoDAF 1.0, but it extends the System View now calling it the Systems and Services View. Each view is composed of standardized products. Within the Systems and Services View two products include extensions to support the description of services and SOA constructs. These views are the Systems and Services Functionality Description (SV-4a and b), and the Operational Activity to System and Services Functionality Traceability Matrices (SV-5a, b, and c). Each of the other SV products includes techniques for explicitly representing services in addition to systems. There is a considerable amount of flexibility in describing services and SOA in the DoDAF 1.5.

There are many organizations that are designing and implementing systems using SOA, and there are many SOA variants. SOA is an approach to defining integration- architectures based on the concept of service. SOA is not the implementation of a specific technology. A service is a collection of applications, data, and tools with which one interacts via message exchange. The services are (1) defined using a common language and are listed in a registry, (2) distributed across the network, but are computer/platform independent, and (3) independent of the communication protocol they utilize. Web Services is one example of services that is focused on the use of browsers to access and provide data and implement processes, but there are other concepts that link together services to support processes. SOAs allow organizations to communicate data without intimate knowledge of each other's IT systems. As DoD migrates from the past point-to-point approach for data exchange to a service approach, it has defined a set of core infrastructure services for the GIG. These comprise the Net-Centric Enterprise Services (NCES), which also are the Assistant Secretary of Defense (Networks and Information Integration) program for creating them. Other non-NCES services are expected to be developed under other programs.

There are many definitions for SOA. One definition is as follows [Hurwitz et al. 2007: 27]. "A SOA is an architecture for building business applications as a set of loosely coupled black-box components orchestrated to deliver a well-defined level of service by linking together business processes." According to this definition, SOA is for building business applications (that is applications to support business processes), not all software. SOA is a black box component architecture, hiding complexity, and enabling reuse of existing applications via "adaptors." In other words, one can encapsulate existing applications and provide an adaptor that provides a standard interface. SOA components are loosely coupled (simplicity and autonomy). Each component carries out a small range of simple services. Components can be combined in a variety of ways. Perhaps the key concept is that SOA components are orchestrated to link together business processes. This orchestration concept can deliver very complex process services and can adapt to maintain specified levels of service. It provides the flexibility, but also increases complexity in terms of both components and messages.

There is a lot of "stuff" going on in a SOA. It isn't enough just to make a set of adaptors for existing applications to make the processes work. SOA requires the creation of several software components, some that make up or support the business processes and others to ensure that they work properly and reliably. The SOA Registry contains reference information about where the components of the SOA are located (an electronic catalog for components) and detailed specification about how to interface with each service. Governance processes must be established to ensure the specifications are published and maintained in the Registry. A Workflow Engine is needed to define the business processes that connect people to people, people to processes, and processes to processes. These process descriptions also are placed in the Registry. Whenever a business process is needed, a Service Broker connects the needed services together using the information in the Registry. An SOA Supervisor ensures that all of the platforms that support the SOA (the plumbing) are running in a consistent and predictable

manner, providing the required service levels. The supervisor monitors all of the running business processes and takes corrective action, if the quality of service is not being met. Finally, an Enterprise Service Bus (ESB) may be required to transport the plethora of messages that passes between the software components so that the end-to-end message passage occurs reliably. Indeed, a full SOA implementation is very dynamic.

The introduction of SOA as the solution to the DoD Data Strategy for assured, secure, authenticated information sharing and the need to incorporate services and SOA in the architecture descriptions create greater analysis and evaluation challenges than those faced prior to SOA. The architecture description mandated as the mechanism for describing solutions for transformation provides a static representation of highly dynamical systems, but quality of service, including performance, is a major requirement and concern for many DoD systems and capabilities. We therefore need to go beyond the typical architecture descriptions to a more complete examination of the logic, behavior, and performance of proposed systems.

Figure 5.1, which was first presented in Wagenhals, Haider, and Levis [2003: 281], provides a framework for a process for achieving a reliable architecture description with rigorous evaluation. In addition to the architecture description and the evaluation outputs, Figure 1 shows the feedback that occurs from the development of the executable model and its use both in evaluation and in the refinement of the architecture design. This diagram provides a high level view of a process for building and evaluating concepts with the help of an executable model of an architecture description. Three processes need to be addressed: (1) a process for creating the architecture description (DoDAF does not specify any process), (2) a process for converting the architecture description to the executable model and (3) a process for using the executable model for analysis and evaluation.,

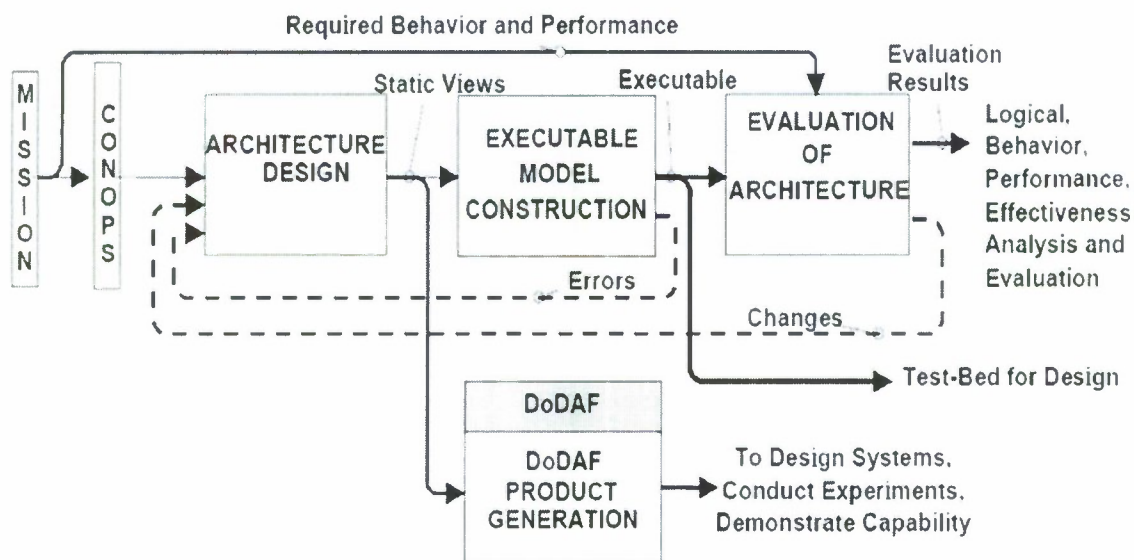


Fig. 5.1: Architecture Design and Evaluation.

The architecture creation process relies on one of two prevalent methodologies, Object Orientation and Structured Analysis. Either methodology can produce all the information needed for conversion to the executable model [Wagenhals et al., 2000] and [Wagenhals, Haider, and Levis, 2003], but care must be taken to follow procedures that ensure all the needed data are captured. A key concept is that all elements of the executable model must be traceable to elements in the architecture description. As more is learned about the behavior and performance of the architecture from the creation of the executable model and the detailed analysis of behavior, any changes detected using the executable model are used to modify the architecture description. Discrete event dynamical system models are appropriate for the executable model, and the Colored Petri Net (CPN) is a sufficiently general and rigorous model [<http://wiki.daimi.au.dk/cpntools/cpntools.wiki>].

Other modeling approaches such as State Machines, Queuing Models, Automata, etc. can be used, but CPNs subsume all of these. CPNs were chosen because they are graph theoretic, executable, and enable both simulation and analysis of properties. They are rigorous in the way they handle concurrent and asynchronous events. Once created, the executable model can be used to support logical, behavioral, and performance evaluations as will be described in Section 5.4.

5.3. Process Descriptions

Wagenhals et al. [2000] developed and described a process for creating the DoDAF Operational and System Views using Structured Analysis. A process for using Object Orientation and the Unified Modeling Language (UML) was described and illustrated in [Wagenhals, Haider, and Levis, 2003]. A model of this UML process is shown in Fig. 5.2. The process evolves through six stages. Stage 0 initiates the effort and includes the articulation of the purpose and scope of the architecture, as well as the identification of background documentation needed to create the architecture. Stage 1 focuses on developing the operational concept. Organizations and their relationships are defined in Stage 2 along with an initial sketch of the system nodes and links of the Systems and Services View. Stage 3 involves a full analysis of the Operational View. If Object Orientation is used, both structure and behavior diagrams are developed to understand and describe the operational activities carried out by organizations and the information that needs to be generated and exchanged. If Structured Analysis is used, this analysis is accomplished using activity, data, rule, and dynamics models. Stage 3 also turns to the Systems and Services View by developing mappings from the operational activities to the systems, services and system functions. In Stage 4, summary Operational View products are generated, and the detailed analysis effort shifts to the Systems and Services View. The same Object Oriented or Structured Analysis techniques are used, but the focus is on system components and their functions along with system data that is exchanged. In the last stage, Stage 5, the architect extracts data and concepts from the Stage 4 analysis and generates system and service interface descriptions, the communications infrastructure description, the system and service performance parameters documentation, and the system, service, and technology

evolution descriptions. Both Structured Analysis and Object Orientation based on UML can produce a complete architecture description conformant to the DoDAF products.

Both will describe the same operational activities, information exchanges, operations nodes, etc. in the Operational View, and the same systems, services, nodes, interfaces, data exchanges, and communications systems, in the Systems and Services View. When using Structured Analysis, the key models are activity models (IDEF0 or Data Flow Diagrams), data models (e.g., IDEF1X or Entity Relationship Diagrams), rule models, and dynamic models (e.g., state charts and event traces). If Object Orientation is used, structural diagrams (class, component, and deployment diagrams) and behavior diagrams (activity, state machine, and sequence diagrams) are developed. Some of the UML products look different than products developed using Structured Analysis models, but the concept content is the same.

In following the process described in Figure 2, the development of the architecture must adhere to certain design constraints in order for the architecture data to be converted into the CPN executable model using the techniques described in Wagenhals et al. [2000: 230– 236] and Wagenhals, Haider, and Levis [2003: 275– 278]. This means that both the Operational and the Systems and Services Views must be designed to carry out the operational concept.

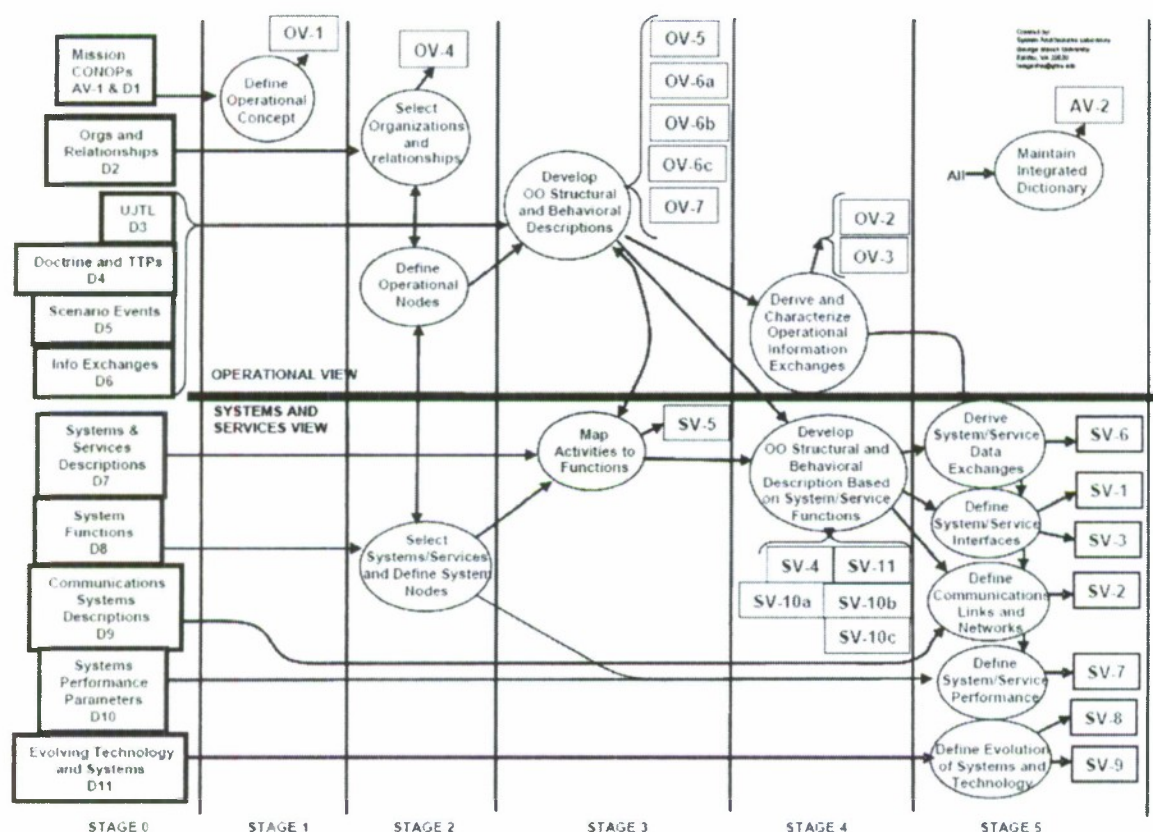


Fig. 5.2: DoDAF Architecture Design Process.

The views must include all activities or functions and their relationships, define all of the information or data exchanges (messages), and express the logic of each activity or function. This architecture data are used to produce the CPN with the functions being modeled as transitions and the relationships between the functions using directed arcs and places. The color set declarations for the tokens in each place are obtained from data descriptions, and the logic descriptions are used as arc inscriptions or guard functions. These architecture data are contained in both Structured Analysis and UML models, although the format is different. This means that a Structured Analysis derived CPN will not appear to be identical to one derived from a UML description of the same architecture, but since the same functions, relationships, data, and logic will be described, the behavior of the two should be identical. Concordance must be maintained wherein consistency and completeness are assured across all of the data and the descriptions that are created using the various UML or Structured Analysis diagrams.

Once the executable model has been created, it can be used to address, in part, the following layered questions: (1) Is the architecture logically correct? (2) Does the architecture exhibit the desired behavior? (3) Are the instantiations of the architecture in the Systems and Services View consistent with the Operational View? (4) Do instantiations of this architecture exhibit the desired performance characteristics? (5) Do systems built in conformance to the architecture provide the desired capability? (6) Can we analyze alternatives?

The construction of the executable model, especially of the one based on the operational view, provides the basis for checking the logical consistency and correctness. The first step is to validate the logic of the model. The static views describe the structure, data, and rules that manipulate that data to accomplish tasks. We need to verify that the combination of rules, data, and structure “works,” e.g., the rules are consistent and complete. This can be accomplished by executing the model to be sure that it runs properly. In a sense, we are “debugging” the architecture. Any errors found must be corrected in the appropriate static views to preserve traceability.

We can execute the model using notional inputs to determine whether activities do indeed use data specified by the information exchange requirements. “Flaws” can result in either an incorrect response or a deadlock. We can test the sequence of events; i.e., does the executable model produce the sequences specified by the sequence diagrams? And we can see whether the execution of activities is a correct implementation of the operational concept.

Once we verify that the executable model runs properly we can examine the behavior of the architecture; this is an examination of the functionality of the architecture. The behavior of the executable model and the behavioral diagrams should correlate. This behavior evaluation has several facets: Does the architecture produce all the correct output for a given stimulus? Does the information arrive at the right functions in the right sequence, i.e., are the inputs processed in the required way? The behavior of the architecture can be compared to the user’s requirements.

The behavioral correctness can be approached from two perspectives: the operational perspective and the systems and services perspective. For the operational perspective, scenarios are developed and executed to determine whether the desired behaviors (as reflected in the state charts or event traces) are obtained. What is of particular interest here is the identification of undesired behaviors or the possibility of undesired states. Note that state transition descriptions (OV-6b and SV- 10b) and the event trace descriptions (OV-6c and SV- 10c) capture only a few of the desired behaviors. A real system may exhibit many more behaviors and the use of an executable model is one way of determining them. If CPNs are used for the executable model, then algorithms based on invariants can be used to relate the structure of the model to its behavior [Valraud and Levis, 1991].

Once behavioral correctness has been established, performance can be examined. With DoDAF architectures, performance of the implementation of an architecture can be evaluated only through the use of the executable model derived from the architecture data in the systems and services view. The performance parameters of the systems and services used to implement the architecture are obtained from the Systems/Services Performance Matrix (SV-7). Scenarios need to be developed that are consistent with the use cases. Data collected from simulation can be used to compute relevant Measures of Performance (MOPs). CPNs offer more than just simulation to support the analysis and evaluation. CPNs in general (and CPNTools [2008] in particular) allow behavioral properties to be verified by analysis without resorting to simulation. State Space Analysis is an analysis technique that provides a variety of properties about a CPN [Kristensen, Christensen, and Jensen, 1998: 122–129]. State Space Analysis techniques have been implemented in CPNTools. While each simulation run of the executable model shows particular sequence or trajectory of processing for a given input set, State Space Analysis shows all possible trajectories for a given input. State Space Analysis provides a detailed look at all possible sequences of states that can occur given a specific input set. Thus, it can be used to see if it is possible for a set of inputs to generate undesired sequences or outputs. In addition, State Space Analysis can determine several important properties of the state space of a CPN model. These include statistics such as the total number of states and transitions between states, liveness properties such as the number and identity of final states, and the number and identity of CPN transitions that can never fire in any sequence. The analysis determines boundedness properties that identify the minimum and maximum number and type of tokens (a CPN representation of an instance of data, e.g., messages) that occur in each place. It also captures the marking (distribution of tokens) for any state including the final states so that these can be examined. Figure 5.3 illustrates the relationships that can exist between the Architecture Description and its executable model as the latter is used both in simulation and State Space Analysis.

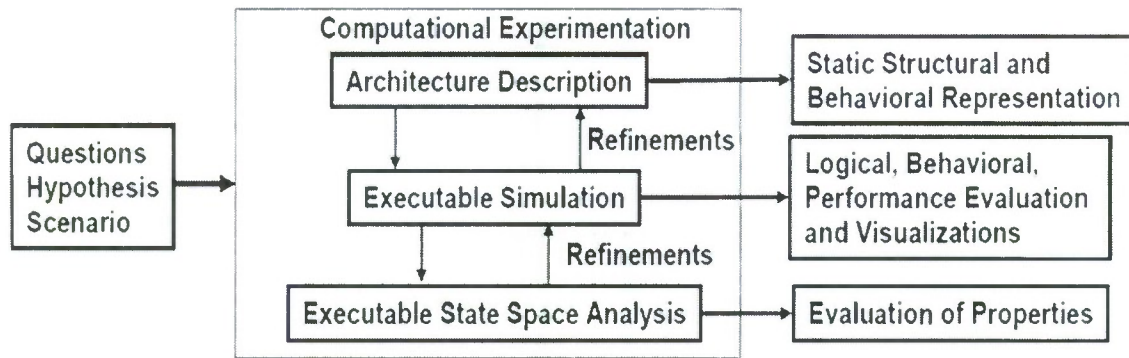


Fig. 5.3: Architecture Evaluation with Executable Model.

It is important to understand that building the executable model does not provide in itself an evaluation. The logical, behavioral, and performance aspects described above outline the steps of a process for evaluation, and the executable model becomes an important tool in that process. Each step requires more effort and additional information. One advantage of this staged approach is that one does not need to enter the details of the systems, a very laborious and costly undertaking, until the previous stages are completed satisfactorily.

5.4. Case Study

A Case Study has been created to demonstrate the process for completing a full DoDAF 1.5 compliant architecture. It is based on a hypothetical operational concept for a new Theater Ballistic Missile Defense (TBMD) system called the Airborne TBM Intercept System (ATIS). It is not an accurate description of such a system—it has been created for the express purpose of illustrating the architecture design process, especially the case where incorporating new information technology (e.g., net centric concepts and a modified interceptor missile) in existing large legacy systems provides a new capability. It was assumed that the architecture will be used to inform decision-makers about the nature of a new TBMD system and some of the tradeoffs involved in building one.

All DoDAF 1.5 products were produced using the previously described process; both Object Orientation (using UML) and Structure Analysis were used to illustrate the techniques for both methodologies. An Executable model using CPNs and the CPNTools software was created for the Operational View. Logical, behavior, and high level performance evaluations were accomplished using the simulation capabilities of the tool. We describe the process for creating the DODAF 1.5 All View 1 product, plus all of the Operational and System and Services View products. We will illustrate the conversion to the CPN and the use of the CPN to perform the analysis.

As shown in Fig. 5.2, the first part (Stage 0) of the process involves the collection of data and information that is pertinent to the architecture. This includes any description of the operational concept, potential operational activities from authoritative sources such as the Universal Joint Task Lists [2002], doctrine, tactics, techniques, and procedures, etc. It also includes information about systems, services, and communications networks. During Stage 0, it is imperative that the

architect or architecting team establish the purpose and the scope of the architecture description. The purpose defines the questions that the architecture description will answer and the time frame of the architecture. In the case study, the purpose was to develop an understanding of the arrangement and interoperation of organizations and systems that support the concept of operations for ATIS. The architecture is designed to determine if the operational concept can be made to work and to assess the impact of evolving this system into the Net Centric Environment including its evolution to incorporate Net Centric Enterprise Services and create special services of its own. An additional goal is to be able to assess the ability of the proposed system to destroy incoming TBMs based on the capabilities of Adversary A and B to launch them. Since we have knowledge about the individual TBMs, but do not know exactly how many TBMs each adversary has or how many can be launched at one time, we need to bound the problem and define how many ATIS assets will be needed to give us the capability to defeat them. Supplementing the purpose is a point of view; in this case, it is that of the ATIS Commander who would understand all of the operating procedures and know the basic systems. The scope includes a time frame between 2010 and 2015. All of this initial analysis, i.e. purpose, scope, and reference data will be cataloged in the All View Overview and Summary Information (AV-1). After the architecture has been completed the results of the analysis also will be included in the AV-1. I

In Stage 1 of the basic process, the architect develops the operational concept and creates both a graphic and a textual description. This becomes the OV-1, Operational Concept Graphic product. Figure 5.4 show the Operational Concept graphic. Note that the elements of the graphic represent types of operational nodes not systems. Of course, in a system like ATIS, the operational concept will rely heavily on systems (Radar, Interceptors, Command Centers, etc.), but it is important not to constrain the design of the material solution by specifying exact systems in the operational concept. Indeed, the concept could support a nontraditional airborne laser "interceptor" where the "missile" is a laser shot. This OV-1 would not need to be modified significantly to support this material solution. Note that the boundary of the ATIS architecture has been depicted in the graphic to distinguish what the ATIS architecture is composed of as well as the entities that will be external to ATIS but that will interact with it. The textual description explains that the operational concept is based on the use of existing legacy systems that will be given modifications to support the concept of intercepting Tactical Ballistic Missiles using modified interceptors and interceptor missiles.

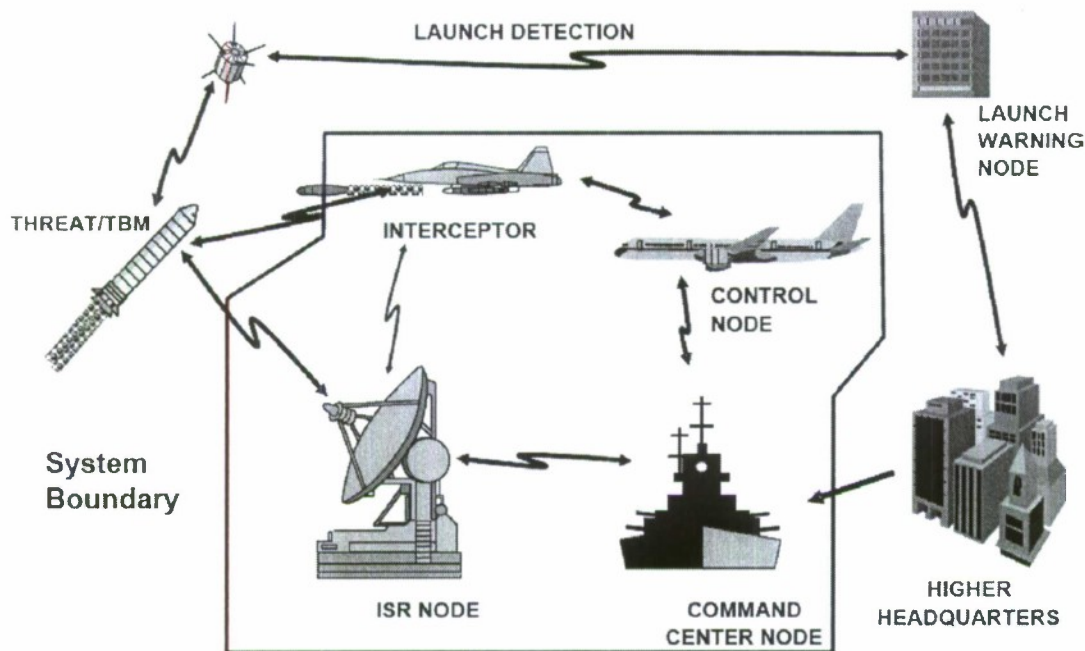


Fig. 5.4: OV-1, Operational Concept Graphic.

In Stage 2 of the basic process, the architect determines the organizations that will execute the operational concept and the relationships that will exist between those organizations. Figure 5.5 shows the OV-4, the Organization Relationship Diagram. The UML format is shown. If Structured Analysis is used, the resulting diagram would be very similar. It would not have the UML symbology for the relationships between the organization, but rather different styles of lines to show the different types of relationship. Understanding these relationships is key to most military command and control architectures.

The organizations will provide the operators who are responsible for performing the operational activities that will be defined in other OV products. These operational activities will be allocated to what are called operational nodes in the Operational Views. At this point it is possible to determine what the operational nodes will be. The designation of operational nodes may be based on the organizations that will carry out the operational activities at those nodes or on logical groupings of operational activities that will be carried out by one or more organizations. In the case study, four ATIS operational nodes were created: a Command Node, a Sense Node, a Control Node, and an Intercept Node. In the case study, these operational nodes have a one-to-one mapping to the organizations, but that will not always be the case. In selecting organization, the architect also must be aware of the assets (systems) that the organizations possess and use. This understanding of the organization along with the operational concept can be used to create an initial sketch of the systems and services view SV-1. While this is only a sketch, it can start the process of creating the SV products. In the case study it was assumed that existing systems such as Radars, Command Centers, Control Centers, and Interceptors would be used as the systems. Figure 5.6 shows an initial sketch of those Systems using the UML deployment diagram. A similar sketch can be created without using UML.

The process now shifts to Stage 3, which has two foci. The first is on a detailed analysis and description of the operational view. The effort describes the operational activities and their relationships, the operational information, and the dynamic behavior of the operational view. The second focus is on the System and Services View and mapping the operational activities to the systems, system functions, and services that will support those activities.

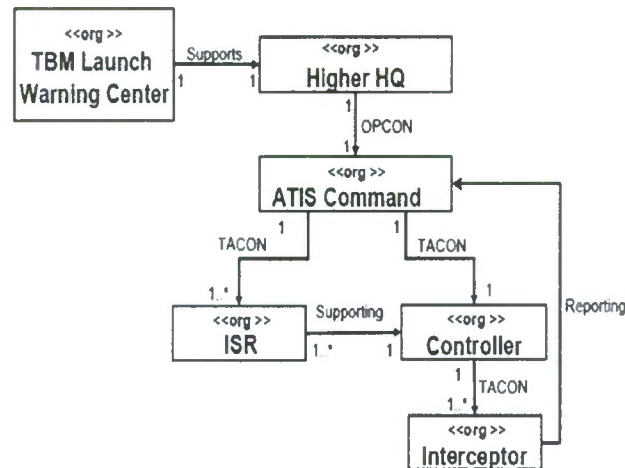


Fig. 5.5: OV-4 Organization Relationship Diagram

The Operational View models and products of Stage 3 include the Activity Model (OV-5), the Logical Data Model (OV-7), and the behavior descriptions including the Operational Rules Model (OV-6a), the Operational State Transition Description (OV-6b) and the Operational Event Trace Description (OV-6c). These products can be represented using either UML diagrams or the various models that are traditionally used in the structured analysis methodology (activity models, data, models, rule models, and dynamic models).

Figure 5.7 shows a UML activity diagram that was created for the case study. The activities were derived from a functional decomposition that was created to represent the type of information that is in the Universal Joint Task List [UJTL, 2002]. Only the leaf level tasks were used to create the diagram. If the architect allocates the operational activities to the operational nodes, then the activity diagram can be created using swim lanes, one swim lane for each operational node. Figure 5.8 shows this version of the OV-5. Note that there may be more than one activity diagram for OV-5. In the case study, there were two, one for Adversary type A and one for type B.

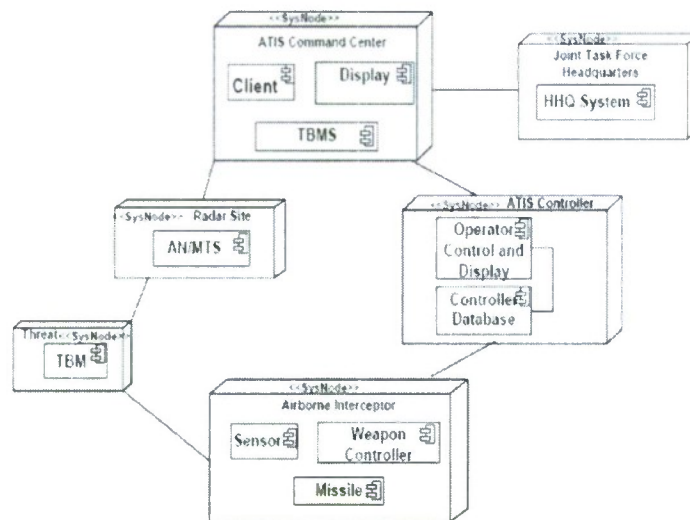


Fig. 5.6: Initial Sketch of System Nodes and System.

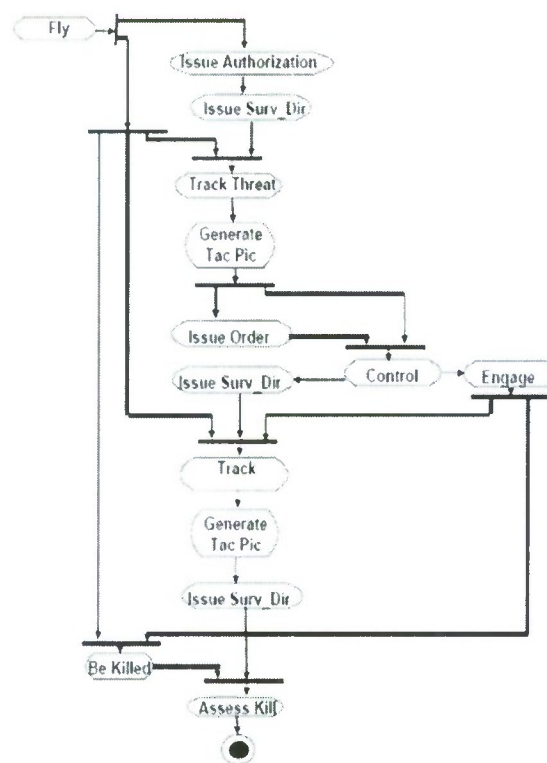


Fig. 5.7: Basic UML Activity Diagram (OV-5).

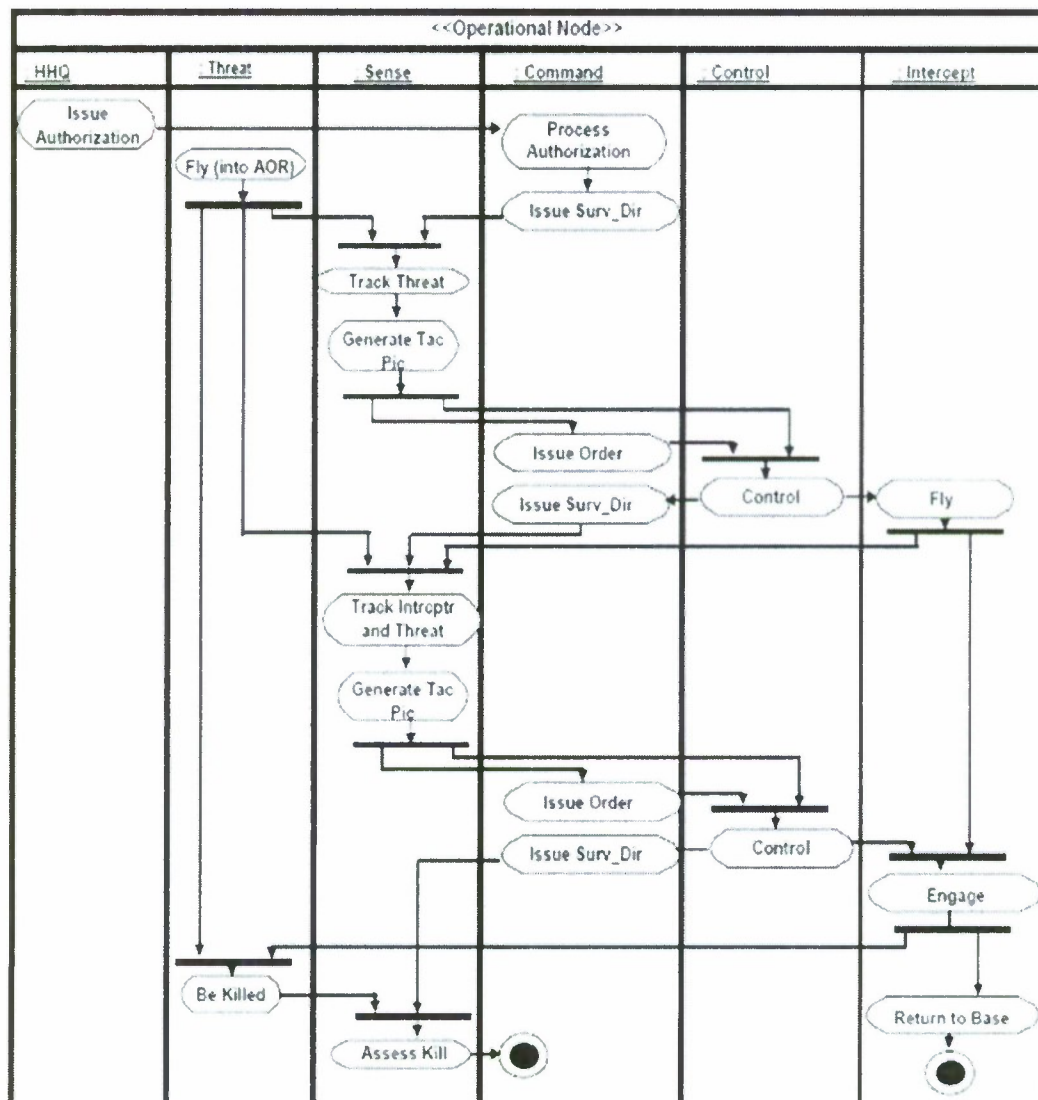


Fig. 5.8: Activity Diagram OV-5 with Swim Lanes.

Once the activity diagrams with swim lanes have been created, it is a simple matter to convert them to UML sequence diagrams that can be used for OV-6c. Figure 5.9 gives an example from the case study. Note that each arrow that crosses a swim lane in the OV-5 becomes a message between the life lines of the UML sequence diagram. We have provided labels for these messages.

In UML it is easy to convert a sequence diagram to a communications diagram. These have more of a structure like appearance than the sequence diagrams. They describe links between the objects' life lines. Each link will be an instance of an association that exists between the

classifiers of each object. Figure 5.10 shows the UML communications diagram that corresponds to the sequence diagram of Fig. 5.9.

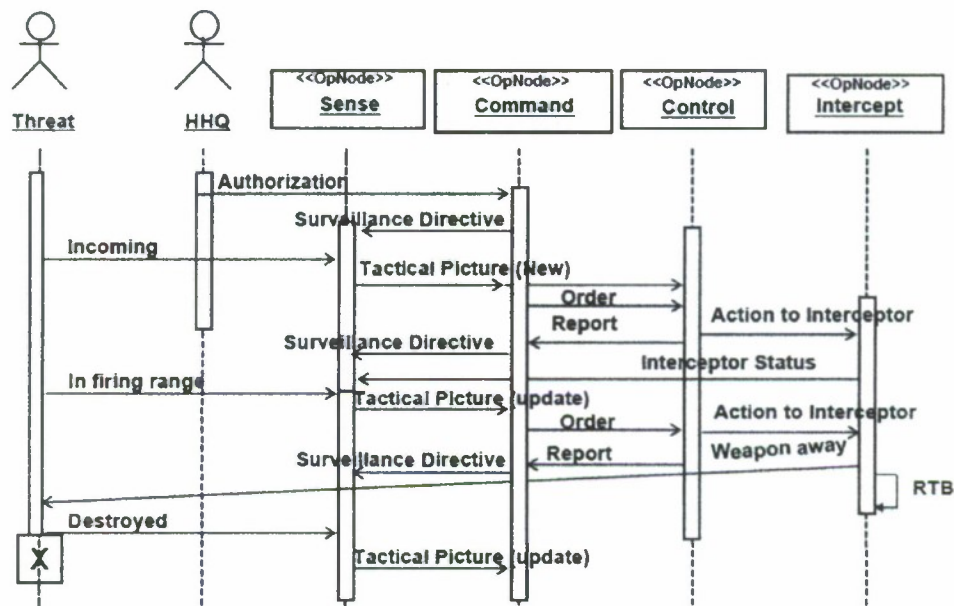


Fig. 5.9: UML Sequence Diagram used for OV-6c.

UML does not have a “rule model” per se. OV-6a is a description of operational rules and must be consistent with the other diagrams. The rules can be created for each operational node. By using the activity diagram with swim lanes, it is straightforward to create rules in the Structured English form of “IF (conditions), Then (Condition or Action), Else, (Condition or Action).” By looking at the arcs that cross swim lanes into an operational node and the activities and their output that are a result of those inputs, one can describe the behavior using Structured English. For example, the rules for the Sense node in the case are:

Rule 1: If Surveillance Directive (Track Threat) and Threat Status (Incoming) then Track Object and Generate Tactical Picture (new).

Rule 2: If Surveillance Directive (Track Intercept) and Threat Status (Incoming) then Track Object, Associate Threat ID, and Int. ID and Generate Tactical Picture (engaged).

Rule 3: If Surveillance Directive (Assess Kill) then Track Objects, Perform Kill Assessment, and Generate Tactical Picture (Killed).

These rules indicate some of the attributes that the operational information exchanges must have. These operational information exchanges and attributes will be described in the logical data model (OV-7). With UML it is possible to create state machine diagrams to describe the behavior of instances of classifiers. These can be used to provide OV-6b. In the case study, four state machine diagrams were created, one for each operational node. One must define the various

states for the operational node and then describe the events (usually the arrival of a message or the completion of an activity or task) that cause the transitions between the states. The behavior described in each state machine diagram should match the flows in the activity diagram with swim lanes and the sequence diagrams. Figure 5.11 shows the state machine diagram for the Sense operational node.

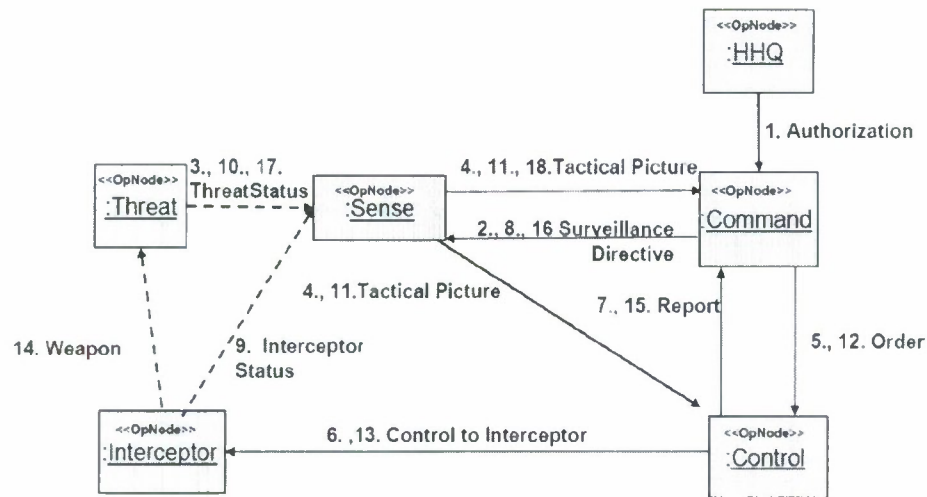


Fig. 5.10: UML Communications Diagram.

Each element of the UML behavior diagram represents an instance of a classifier. A UML class diagram can be created based on the behavior descriptions that created for the OV-5 and -6 series. Figure 5.12 shows the UML class diagram for the case study. Note that classes have been created for each operational node. The operational activities are represented as operations of the classes. Association classes are used to describe the operational information that is exchanged between operational nodes. The association classes enable the architect to describe the attributes of each operational information exchange. Attributes have been included for each operational node. These attributes represent operational information that each operational node knows or stores in order to carry out the operational concept.

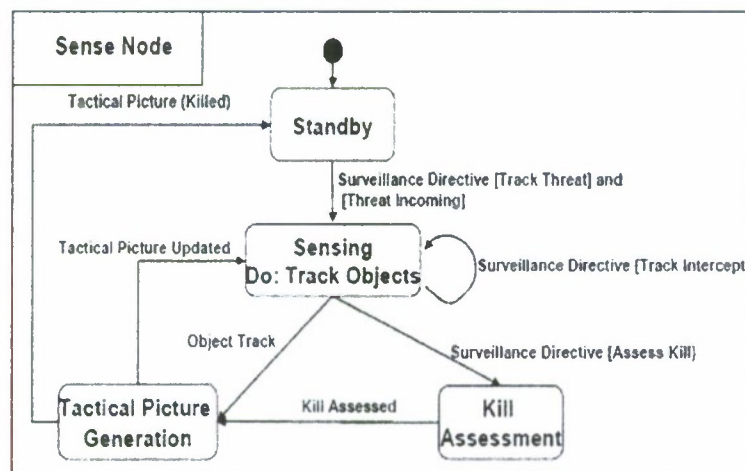


Fig. 5.11: State Machine Diagram for the Sense Node (OV-6b).

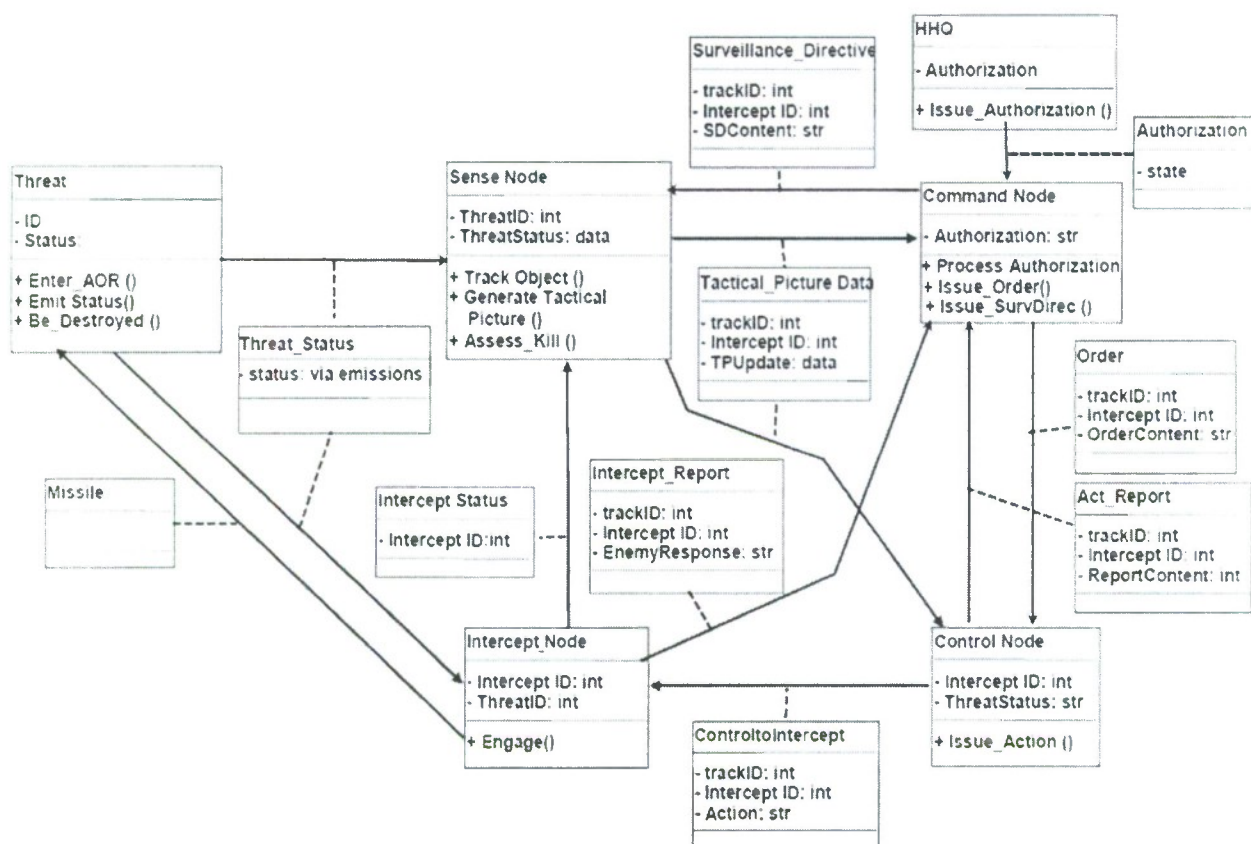


Fig. 5.12: Class Diagram.

The class diagram can be used as the Logical Data Model (OV-7) by hiding the operations. It can also serve the basis for the Operational Node Connectivity Description (OV-2). In this case the attributes are hidden, operations are shown, and each association class is given an operational information exchange identification number. These identification numbers will be used in the Operational Information Exchange Matrix (OV-3). The case study OV-3 is shown in Fig. 5.13. It shows each operational information exchange and the operational node and activity that produces it and the operational node and activity that receives it. OV-3 reflects the analysis shown in OV-5, the -6 series, and -7.

The OV products that have been shown are based on UML diagrams. If the Structured Analysis methodology is used, the same architecture data would have been created, and all of the OV products except for OV-5 and OV-7 would be the same. OV-5 and -7 would have a different appearance because different modeling languages would be used. Figures 5.14 and 5.15 respectively show the OV-5 and OV-7 based on IDEF0 and IDEF1X modeling languages. Structured Analysis is based on functional decomposition; therefore, the IDEF0 would follow the activity decomposition provided by the UJTL. Figure 14 shows the decomposition of the context (A-0) page; the full model had three levels. Note that the IDEF1X description of the operational information exchanges and their attributes is the same as described in the OV-7 based on UML.

		Description	Producer		Consumer		Performance	Security
Needline ID	Info Exchange ID	Name	Op Node	Activity	Op Node	Activity	Timeliness	Protection
1	1.1	Threat Status (Radar Return)	Threat	N / A	Sense	Sense (Track Threat / Assess Kill)	Real Time	none
2	2.1	Threat	Threat	N / A	Intercept	Engage	Real Time	none
3	3.1	Authorization	HHQ	N / A	Command	Process Authorization	5 seconds	Secure
4	4.1	Tactical Picture	Sense	Generate Tactical Picture	Command	Issue Intercept Order	5 Seconds	Secure
5	4.2	Tactical Picture	Sense	Generate Tactical Picture	Control	Control	5 Seconds	Secure
6	5.1	Surveillance Directive	Command	Issue Surveillance Directive	Sense	Sense (Track Threat / Assess Kill)	5 seconds	Secure
7	6.1	Order	Command	Issue Order	Control	Control	5 Seconds	Secure
8	7.1	Report	Control	Control	Command	Issue Surveillance Directive	5 Seconds	Secure
9	8.1	Control to Interceptor	Control	Control	Intercept	Engage	5 Seconds	Secure
10	9.1	Interceptor Report	Intercept	Engage	Control	Control	5 Seconds	Secure

Fig. 5.13: OV-3 Operational Information Exchange Matrix.

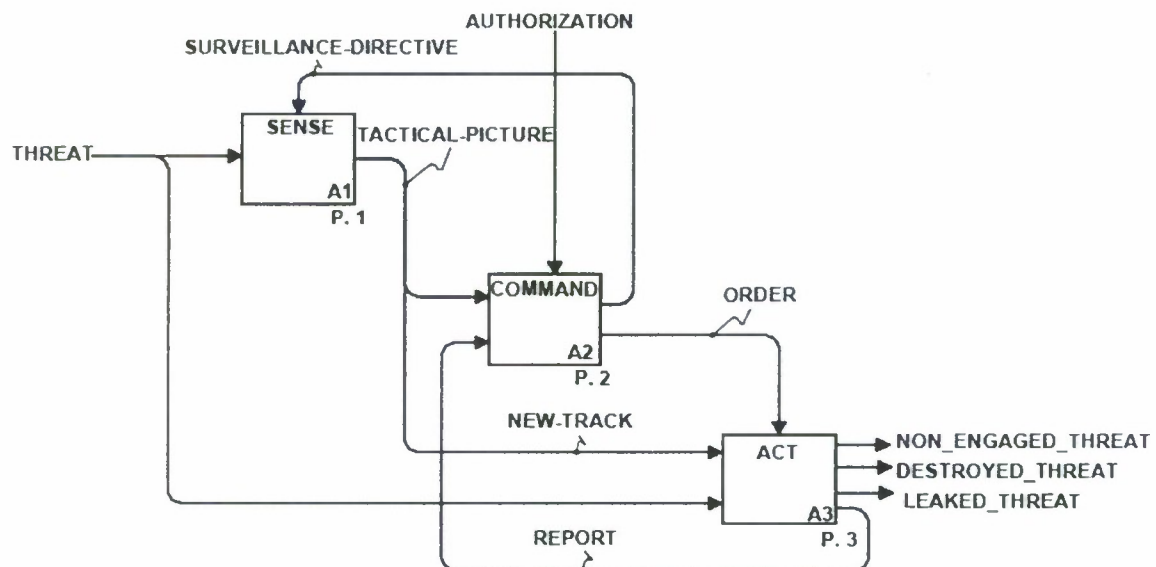


Fig. 5.14: OV-5 Based on IDEF0.

As Stage 3 nears completion, it is possible to start the design of the Systems and Services View. This work should not begin until the Stage 3 of the Operation View is complete because the Systems and Services View shows how the material resources will support the operational view. The first step is to develop the mapping between the operational activities and the system functions, the systems, and any services. These mappings are described in the SV-5 products. Producing these mappings is a systems engineering activity involving trade-offs between different potential configurations of systems and services to support the operational view. In the case study, it was assumed that the architecture would rely on as many legacy systems as possible. In addition, one of the questions to be explored with the architecture was the impact of employing services within the architecture. The systems and their system functions were fabricated for the case study. The basic systems have been shown in the initial sketch of Figure 5.6. For services it was assumed that three of the Net Centric Enterprise Services being developed by DoD will be available. In addition, it was assumed that a Global Ballistic Missile Warning Service would be available. Two ATIS specific services were postulated: a tactical picture service capable of providing tactical picture of the TBM engagements and a special kill assessment service that could support the determination of success of each TBM engagement.

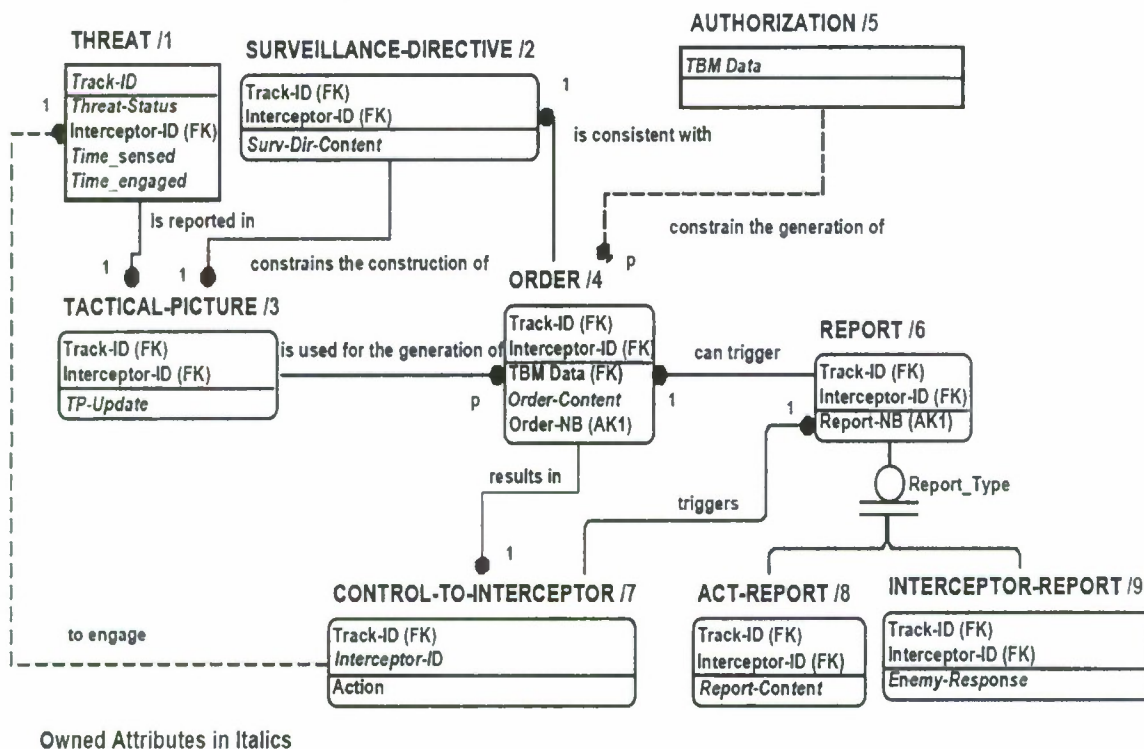


Fig. 5.15: OV-7 Based on IDEF1X

Figure 5.16 shows a matrix that was generated to map the operational activities to existing systems and the system functions they provide. Figures 5.17, 5.18, and 5.19 show the three DoDAF 1.5 SV-5 products. SV-5b and c are new products in DoDAF 1.5. SV-5b maps capabilities defined as a grouping of operational activities to system and their system functions. SV-5c maps the capabilities to services. Note that the latter two SV-5 products allow the use of a

stop light color coding system to describe how well each system or service supports the operational activity and thus the capability. It is important that these products be provided with a time stamp or time window as the ability of a system or service to support a capability may change over time. In the case study example the SV-5b and c were developed assuming an initial operational capability in 2010. Thus some of the system functions and the services will not be fully capable in that time frame. Instead of a stop light system, a grey scale has been used to reflect the readiness of the system function or service with light, medium, and dark grey meaning good, partial, and non-satisfactory capability, respectively. The SV-5b and c products are shown in Figs. 5.18 and 5.19.

Activities	System Functions	System
Track Objects	Search Sector Detect Objects Identify Objects Track Objects	Missile Tracking and Surveillance RADARs (AN/MTS)
Generate Tactical Picture	Generate Tactical Picture Data	Missile Tracking and Surveillance RADARs
	Create Tactical Picture Maintain Threat Status Display Tactical Picture	TBMS (Tactical Picture Service) Display
Assess Kill	Detect Objects Identify Objects Track Objects Assess Status	Missile Tracking RADAR Kill Assessment (Weapon Effects Analysis) Service
Process Authorization	Parse Authorization Relay Authorization Display Authorization	TBMS Display
Issue Surveillance Directive	Generate Surveillance Directive Display Surveillance Directive Select Surveillance Directive	TBMS Display Operator Client
Issue Intercept Order	Generate Intercept Directive Display Intercept Directive Select Intercept Directive	TBMS Display Operator Client
Control	Display Local Tactical Picture Plan Intercept Process Messages	Operator Control and Display System
Engage	Find, Fix, Track Objects Target TBM Assess Boost Launch Interceptor Detonate	Interceptor Sensors Weapon Control System Interceptor Missile

Fig. 5.16: Mapping from Operational Activities to Systems and their Functions.

Once the mapping from operational activities to systems and services has been postulated, the architect moves to a detailed analysis of those systems, services, and functions in Stage 4. The techniques are similar to the ones used for the operational view. When using UML, the architect will create a set of behavior and structure diagrams. The behavior diagrams consist of the activity, sequence, communications, and state machine diagrams using the same techniques

that were used for the Operational View. Instead of class diagrams for structure diagrams, component and deployment diagrams will be used to focus on the systems and their interfaces. If the Structured Analysis methodology is used, then activity models (e.g. IDEF0 or Data Flow Diagrams), data models (e.g. IDEF1X or Entity Relationship Diagrams), rule models, and dynamic models such as state transition diagrams, and sequence diagrams will be used. Instead of focusing on operational activities, the organizations or operational nodes that will perform them, and the operational information exchanges, the System and Service View analysis is focused on system nodes, systems and their function, or services, and system data that is exchanged. We will first show some of the products produced using UML. Every diagram will not be shown; we omit the diagram if the diagram creation technique is similar to that used in the Operational View.

System Functions	Operational Activities							
	Track Object A11	Generate Tactical Picture A12	Assess Kill A13	Process Authorization A21	Issue Surveillance Directive A22	Issue Order A23	Control A31	Engage A32
Search Sector	■							
Detect Objects	■		■					
Track Object	■		■					
ID Object	■		■					
Generate Tactical Picture Data	■							
Create Tactical Picture		■						
Store Track Data	■	■						
Display Tactical Picture		■						
Assess Kill			■					
Parse Authorization				■				
Relay Authorization				■				
Display Authorization				■				

Fig. 5.17: SV-5a, Operational Activity to Systems Function Traceability Matrix (partial).

We start with the activity diagram which will be used to produce the SV-4. As we did with the operational view, we will use swim lanes. In the Systems and Services View, the swim lanes will be created for the instances of the components that represent the systems or services rather than operational nodes as was done for the Operational View. The activities in the activity diagram will be the functions that the systems or services perform. Figure 5.20 shows part of an

activity diagram that is based on systems and their functions. This activity diagram can be used for SV-4a, the Systems Functionality Description.

System	System Functions	TBM Tracking Capability					TBM Intercept Capability		
		Track Object	Generate Tactical Picture	Assess Kill	Process Authorization	Issue Surveillance Directive	Issue Order	Control	Engage
		A11	A12	A13	A21	A22	A23	A31	A32
Missile Tracking Radar AN MTS	Search Sector	G							
	Detect Objects	G		Y					
	Track Object	G		Y					
	ID Object	G		Y					
	Generate Tactical Picture Input	G							
	Store Track Data	G	G						
	Assess Kill			R					
TBMS	Create Tactical Picture		G						
	Display Tactical Picture		G						
	Parse Messages				G				
	Display Authorization				G				
	Generate Surveillance Directive					G			
	Display Surveillance Directive					G			
	Generate Intercept Order						G		
	Display Intercept Order						G		
Control Operator Control and Display System	Display Local Tactical Picture							G	
	Plan Intercept							Y	
	Process Messages							G	
Interceptor Aircraft	Find, Fix, Track Objects								Y
	Assess Boost								Y
	Target TBM								Y
	Launch Interceptor								Y
Interceptor Missile	Fly and Detonate								Y

Fig. 5.18: SV-5b, Operational Activity to Systems Traceability Matrix.

A similar diagram can be created showing services as components. Such an activity diagram would be used as the SV-4b, Services Functionality Description. Both activity diagrams can be converted to sequence diagrams in the same manner as was done for the Operational View. To focus on the service aspect of the ATIS architecture, a sequence diagram was created that showed the major services (but not the Machine-to-Machine Messaging Net Centric Environment Service). This diagram is shown in Fig. 5.21. It shows the major systems and the services as component life-lines. The service life lines are labeled with the stereotype “<Service>.” Figure 5.21 shows the sequence of data messages that are exchanged to carry out the operational concept when services are incorporated. This diagram can be presented as SV-10c, but DoDAF 1.5 says that this type of diagram also can be used as an SV-4.

Services	TBM Tracking Capability					TBM Intercept Capability		
	Track Object A11	Generate Tactical Picture A12	Assess Kill A13	Process Authorization A21	Issue Surveillance Directive A22	Issue Order A23	Control A31	Engage A32
Missile Tracking Service	Y	Y						
Tactical Picture Service		R						
Kill Assessment (Weapons Effects) Service			R					
NCES Content Delivery Service		R	R					
NCES Discovery Service		R	R					
NCES M2M Messaging Service (part of NCES EBS)			R	R	R	R	R	R

Fig. 5.19: SV-5c, Operational Activity to Services Traceability Matrix.

DoDAF 1.5 states that the SV-4b should include a Service Specification. DoDAF 1.5 provides a minimum set of data each Service Specification should have. Figure 22 shows the case study Service Specification. Only the services that will be part of the ATIS arc included.

As we illustrated with the Operational View, it is easy to convert the sequence diagram, once it has been created, to a communications diagram. While these diagrams are not part of the Systems and Services View products, they can lead to products such as the SV-1, the Systems and Services Interface Description. Figure 23 shows the communications diagram for the case where no services are provided and the interfaces are point-to-point. It was derived from the sequence diagram for SV-10c, the Systems Event-Trace Description (not shown).



Fig. 5.22: SV-4b Service Specification.

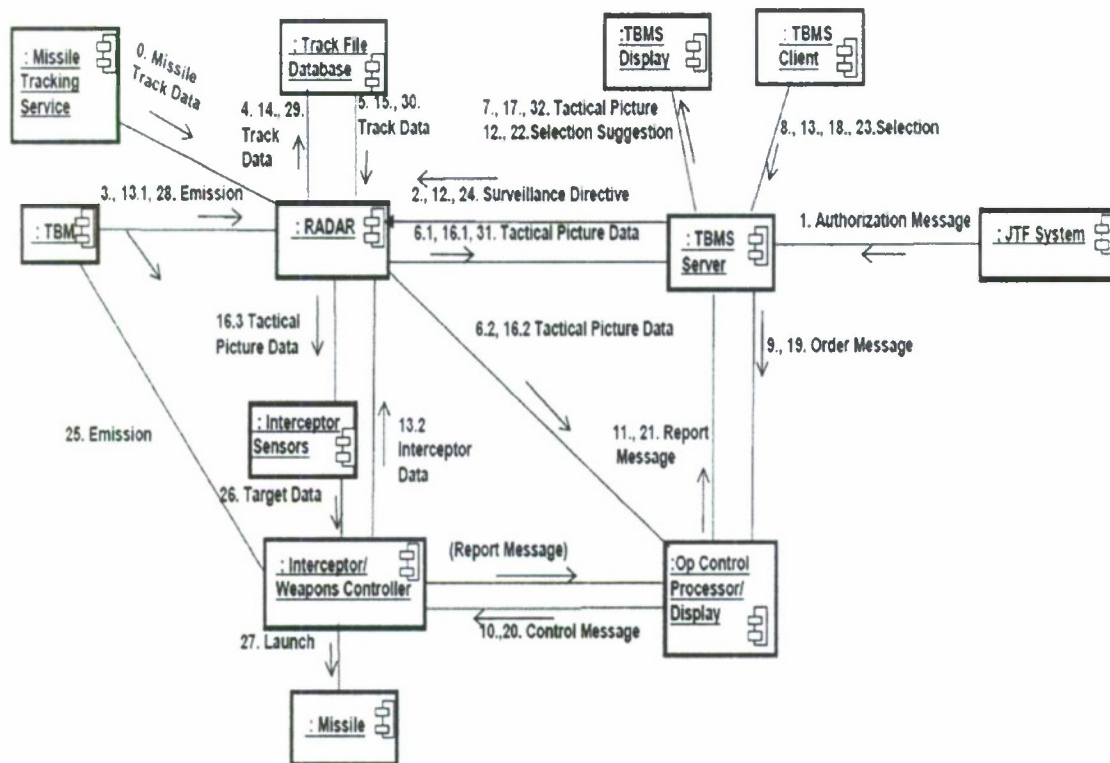


Fig. 5.23: Systems Communications Diagram.

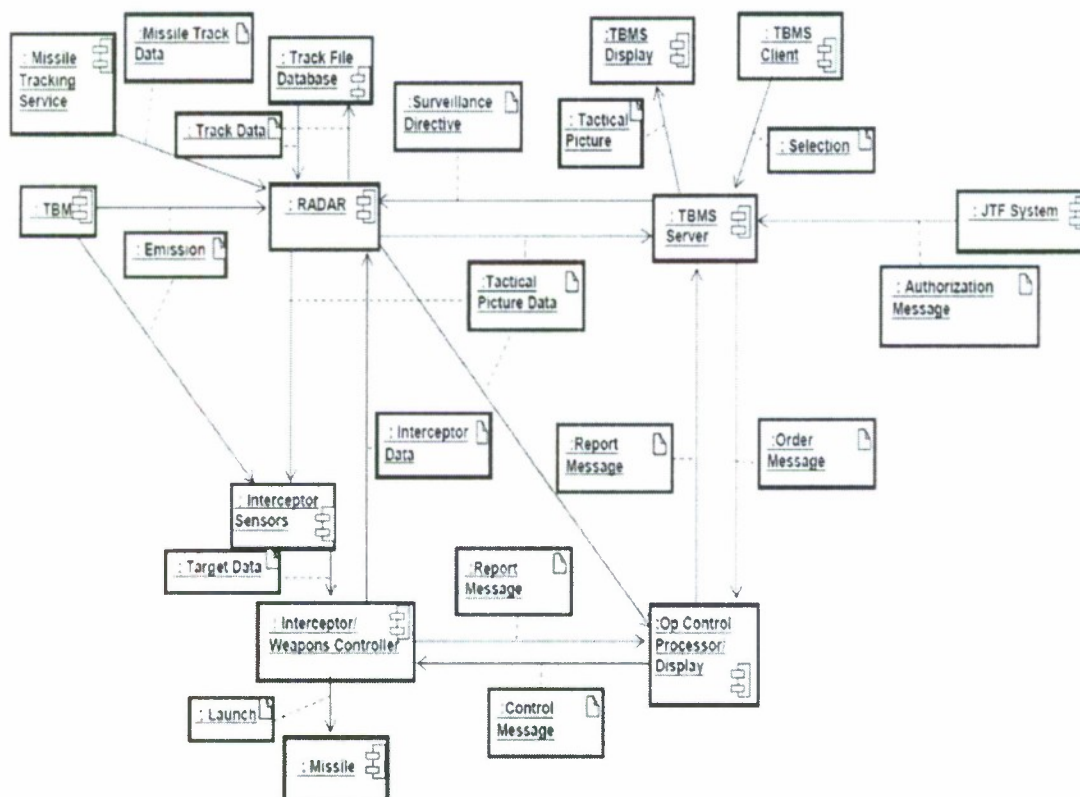


Fig. 5.24: Component Diagram (with only the missile tracking service).

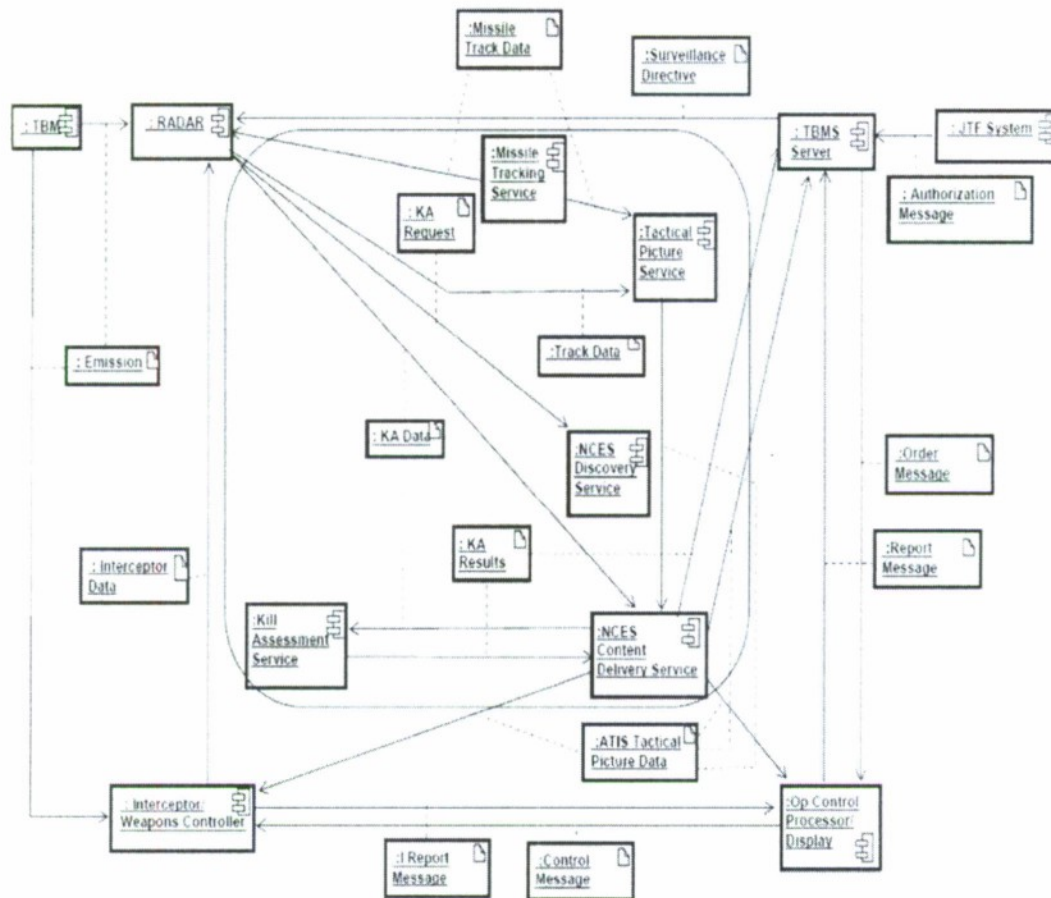


Fig. 5.25: Component Diagram (with services).

In Stage 5, the results of Stage 4 are summarized in a set of Systems and Services View products. Figures 5.28 and 5.29 show the two versions of the SV-1 (Systems/ Service Interface Description). These are UML deployment diagrams and were derived from the component diagrams shown in Figs. 5.24, 5.25, and 5.26. Note that a key interface designation has been added to each interface. These diagrams are analogous to the OV-2, Operational Node Connectivity Description. The details of the system data exchanges will be described in SV-6, the Systems Data Exchange Matrix, which is analogous to the OV-3. Details about each interface are provided in the SV-3 product.

The case study SV-1 shows two concepts. The first SV-1 shows the ATIS without services (other than the TBM Warning Service). The Radar sends and receives system data messages directly to and from the Command Center systems. This design is tightly coupled, and system may need to be designed that way given the time critical nature of the system. Adding services reduces coupling but increase complexity. The radar sends track data to the tactical picture service that converts into the ATIS tactical picture and posts it in the NCES Content Delivery Service. Users can subscribe to this content, and the users can include the other ATIS system nodes. Furthermore, the ATIS Tactical Picture Service may receive data from non-ATIS sensors that could enhance the tactical picture. The Kill Assessment Service is decoupled from the Radar

(but it is shown as part of the Radar system, although it could be located elsewhere). Instead of tightly coupled to the Radar, it can be open to other inputs and processes that may enhance the overall kill assessment product. In a complex SOA environment, the equivalent of a service broker might link together all of the services and data providers prior to any actual TBM intercept. The details of this type of SOA have not been included in the architecture description.

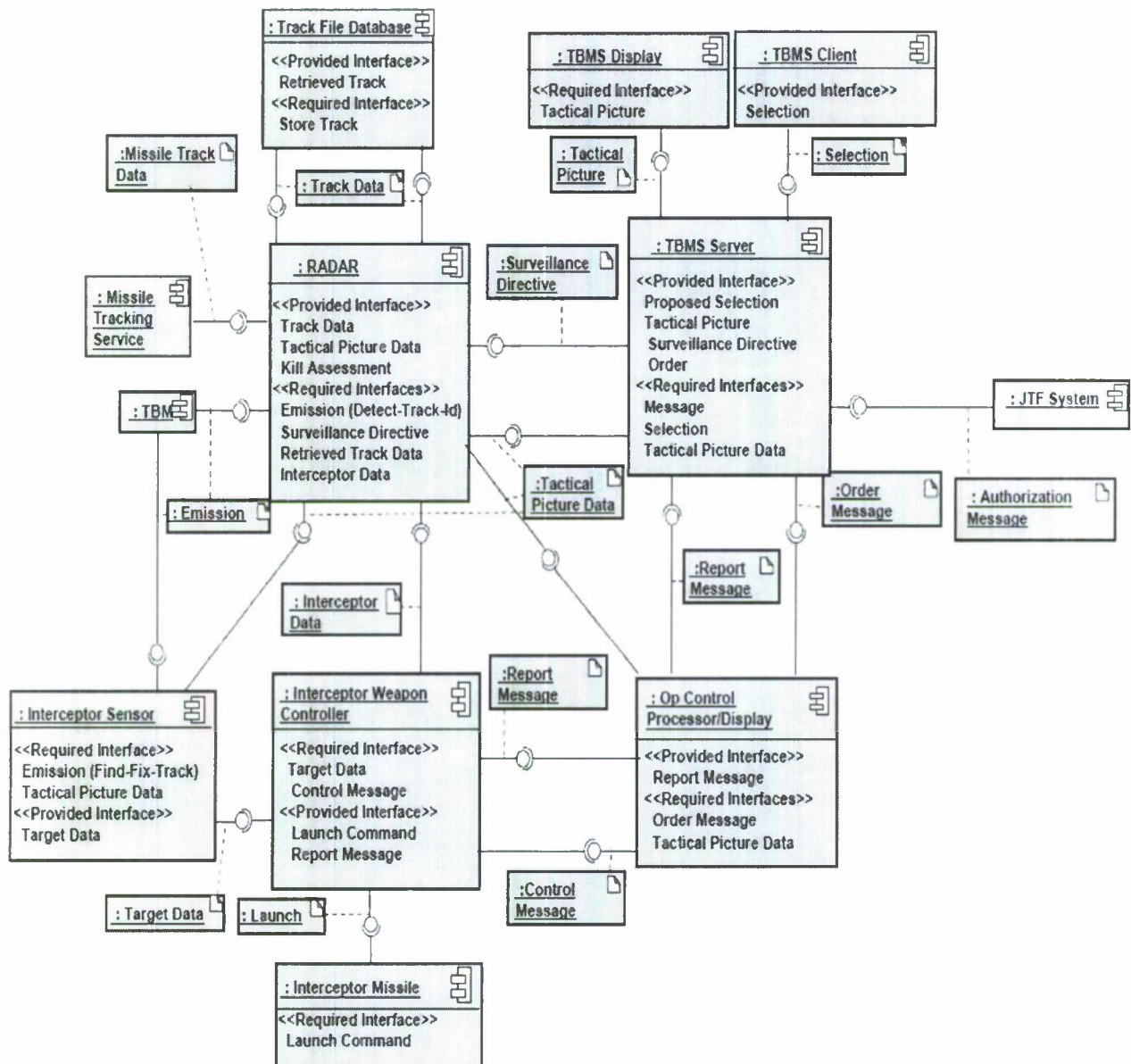


Fig. 5.26: Component Diagram (with interfaces).

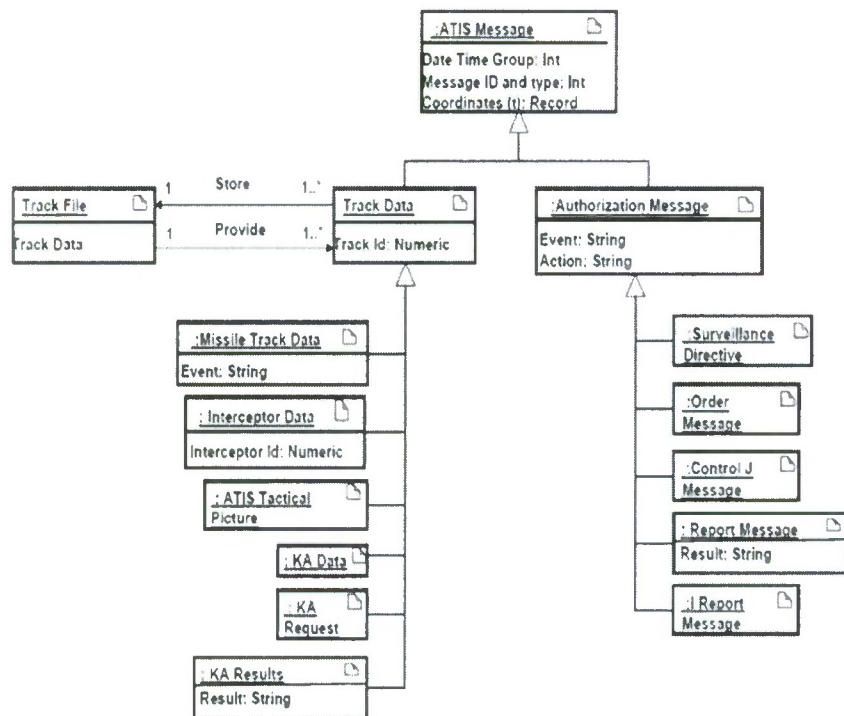


Fig. 5.27: SV-11, Physical Schema.

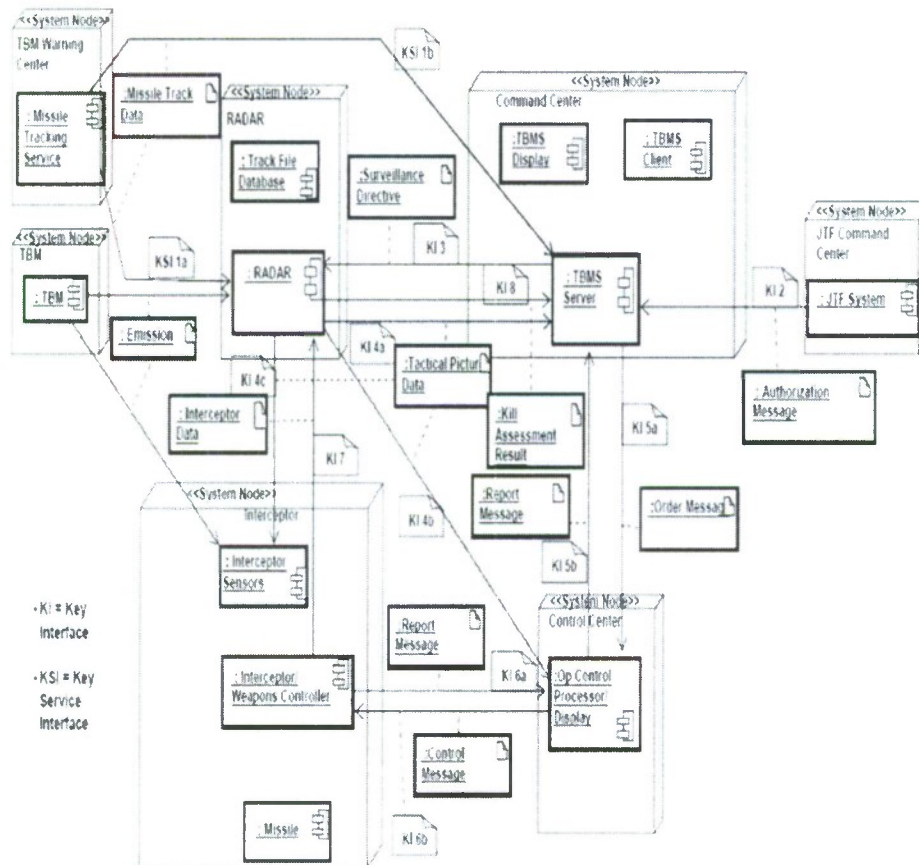


Fig. 5.28: SV-1, Systems Interface Description.

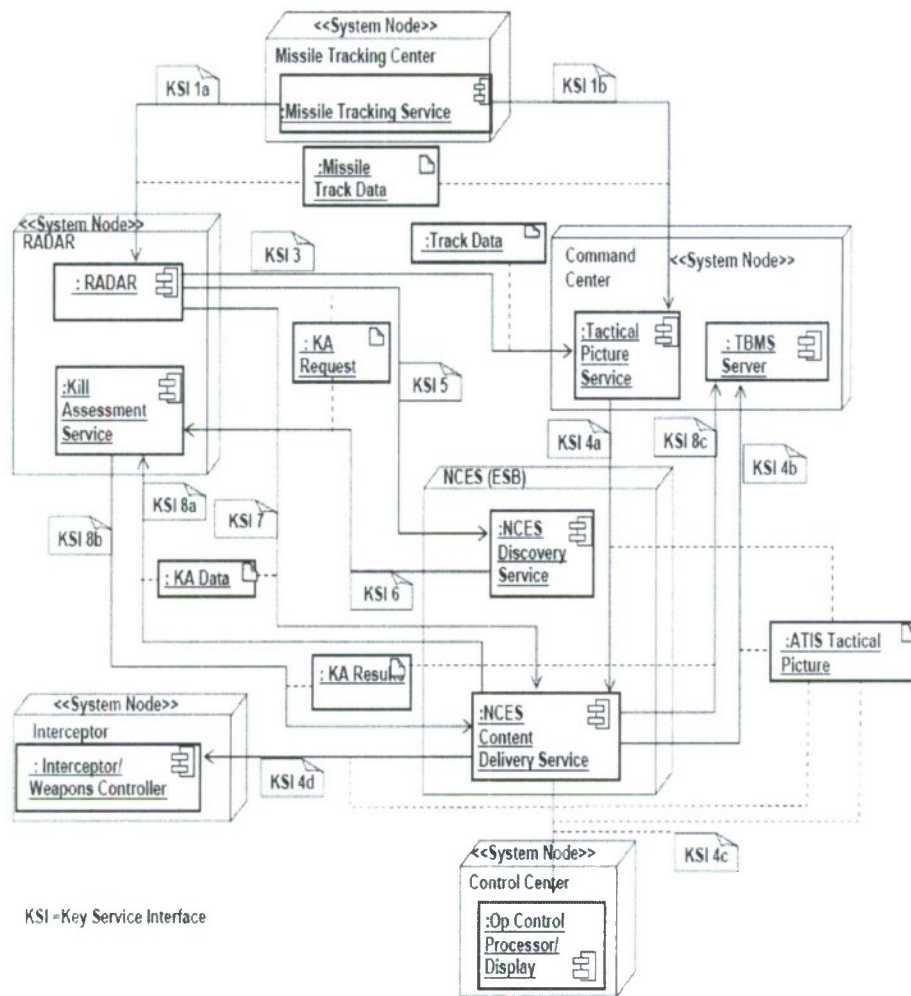


Fig. 5.29: SV-1, Services Interface Description.

The SV-6 (Systems and Services Data Exchange Matrices) are constructed in the same manner as the OV-3. Figure 5.30 shows part of this product that was produced for the Case Study. Other parts show Systems and Services Data Exchanges.

The focus now shifts to the Systems/Services Communications Description (SV-2). Given the interfaces that have been developed in the SV-4 and SV-1, the communications infrastructure description is developed. Generally this will require expertise in communications networks and systems. Figure 5.31 shows a high level description of this infrastructure that was created for the case study. Note that the connection to the NCES to reach the non-ATIS services is depicted as a satellite communications link. The time delays associated with these links will need to be considered. If delays are too long, the ATIS will not be fast enough to carry out the operational concept. For the initial case study, very short time delays were assumed.

The SV-3 (Systems-Systems Matrix, Systems-Services Matrix, and Services-Services Matrix) defines the interfaces, including communications that are depicted in the SV-1 and SV-2. A legend is created that provides codes for the different types of interfaces. Figure 5.32 shows a

portion of the case study Systems-Systems Matrix. The Systems-Services and Services-Services matrices of this product are not shown.

Interface ID	Data Exchange ID	Description	Producer		Consumer		Performance	Security
		Name	Sending System	System Function	Receiving System	System Function	Timeliness	Protection
KSI 1a	1a.1	Missile Track	Missile Warning Service	N/A	AN/MTS Radar	Search, Track, ID	2 seconds	SIPRNET
KSI 1b	1b.1	Missile Track	Missile Warning Service	N/A	TBMS	Tactical Picture Service	2 seconds	SIPRNET
KI 2	2.1	Authorization Message	JTF HQ	N/A	TBMS	Parse Messages	2 seconds	SIPRNET
KI 3	3.1	Surveillance Directive Message	TBMS	Generate Surveillance Directive	AN/MTS Radar	Search, Generate KA	2 seconds	Secure TDMA
KSI 3	3.2	Track Data	AN/TPS Radar	Generate Tactical Picture Data	TBMS	(Tactical Picture Service)	2 seconds	Secure TDMA
KSI 7	3.3	Kill Assessment Data	AN/TPS Radar	Generate Kill Assessment Data	NCES	Content Delivery	2 seconds	Secure TDMA
KSI 8a	3.3	Kill Assessment Data	NCES	Content Delivery	AN/MTS Radar	Kill Assessment Service	2 seconds	Secure TDMA
KSI 5	3.4	Kill Assessment Request	AN/TPS Radar	Generate Kill Assessment Data	NCES	Discovery Service	2 seconds	Secure TDMA
KSI 6	3.4	Kill Assessment Request	NCES	Discovery Service	AN/TPS Radar	Kill Assessment Service	2 seconds	Secure TDMA

Fig. 5.30: SV-6, Systems/Services Data Exchange Matrix (partial).

The remaining products, SV-7 (Systems/Services Performance Parameter Matrix), SV-8 (Systems/Services Evolution Description), and SV-9 (Systems/Services Technology Forecast) show aspects of the evolution of the architecture. Figures 5.33, 5.34, and 5.35 show these products.

As the architecture team completes the architecture development, it returns to the original questions that the architecture was designed to answer. For the case study there were three questions:

1. Can the operational concept be made to work? The answer seems to be yes, based on the architecture, but more detailed analysis of the engagement envelope capabilities of the proposed interceptors and missiles should be undertaken.
2. Can net centric concepts be leveraged, particularly the NCES? The answer seems to be a qualified yes, but the time critical aspects of the operational concept may require a

combination of loosely coupled service-based concepts backed up by a tightly coupled point to point solution.

3. Can we answer questions about the capability of the ATIS system to respond against two different types of adversaries whose exact launch capability is not known? This cannot be answered with the architecture description, but by converting that description to an executable model, some insights can be provided.

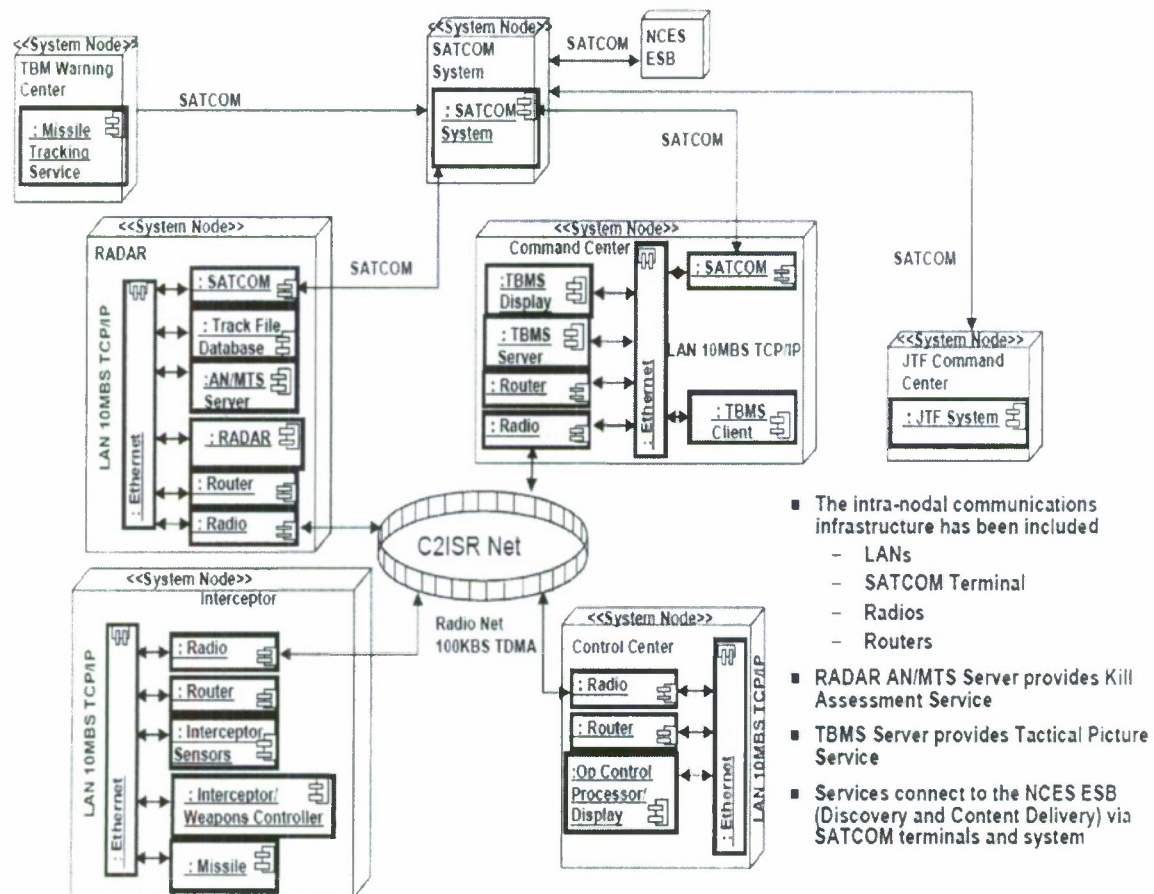


Fig. 5.31: SV-2, Systems/Services Communications Description.

To address question 3, the architecture description was converted to a CPN executable model. The structured analysis version of the architecture was used, and the executable model was created from the operational view. The technique described in Wagenhals et al. [2000] was used. CPNTools was the CPN software application. Figure 5.36 shows the first level of decomposition of the CPN model. Note the similarities between this CPN model page and the IDEF0 page shown in Figure 5.14.

From	To	Radar (AN/MPS)	TBMS	TBMS Client	TBMS Display	Operator Control/ Display	Interceptor Sensor	Weapon Controller	Missile	Missile Tracking	JTF Hq System
Radar (AN/MPS)			S2,P1, C2,M2, K1								
TBMS		S2,P1,C2, M2,K1		S1,P1, C2,M2, K1	S1,P1, C2,M2, K1	S2,P1,C2, M2,K1		S2,P1,C2, M2,K1			
TBMS Client			S1,P1, C2,M2, K1								
TBMS Display											
Operator Control/ Display			S1,P1, C2,M2, K1					S1,P1,C2, M2,K1			
Interceptor Sensor								S1,P1,C2, M2,K1	S1,P1,C2, M2,K1		
Weapon Controller		S1,P1,C2, M2,K1				S1,P1,C2, M2,K1			S1,P1,C2, M2,K1		
Missile						Status	Existing	S1	Means	JWICS	M1
							Planned	S2		SIPRNET	M2
							Potential	S3		Radio Net	P1
Missile Tracking		S2,P3,C2, M2,K1	S1,P3, C2,M2, K1			Purpose	C2	P1	Key interface	Radio Link	P2
							Intel	P2		SATCOM T1 link/WAN	P3
							Logistics	P3		10/100 MBS LAN	P4
JTF Hq System			S1,P3, C2,M2, K1			Classification Level	Unclassified	C1	Key interface	Yes	K1
							Secret	C2		No	K2
							Top Secret	C3			

Fig. 5.32: SV-3, Systems to Systems Matrix.

		Performance Range Measures					
		Time 2008		Time 2010		Time 2015	
System Name	Performance Parameters	Baseline	Objective	Baseline	Objective	Baseline	Objective
AN/MTS- Hardware element- Radar	MTTR MTTF Data Transfer Rate	24 Hr 3 yr 10Mbs	12hr 5yr 100Mbs	12 hr 5 yr 100 Mbs	6 hr 7yr 100MBS	6hr 5 yr 100Mbs	3hr 7 yr 100Mbs
AN/MTS- Software element- Radar	Throughput Response Time Operator Response time	100 KB/s 1ms 10 s	100 KB/s 1ms 10 s	100 KB/s 1ms 10s	1 MB/s 1ms 1s	1 MB/s 1ms 10s	10 MB/s 1ms 1s
TBMS- Hardware element	MTTR MTTF Data Transfer Rate	24 Hr 3 yr 10Mbs	12hr 5yr 100Mbs	12 hr 5 yr 100 Mbs	6 hr 7yr 100MBS	6hr 5 yr 100Mbs	3hr 7 yr 100Mbs
TBMS- Software element	Throughput Response Time Operator Response time	100 KB/s 1ms 5 s	100 KB/s 1ms 1 s	100 KB/s 1ms 3s	1 MB/s 1ms 1s	1 MB/s 1ms 1s	10MB/s 1ms 1s

Fig. 5.33: SV-7, Systems Performance Parameter Matrix.

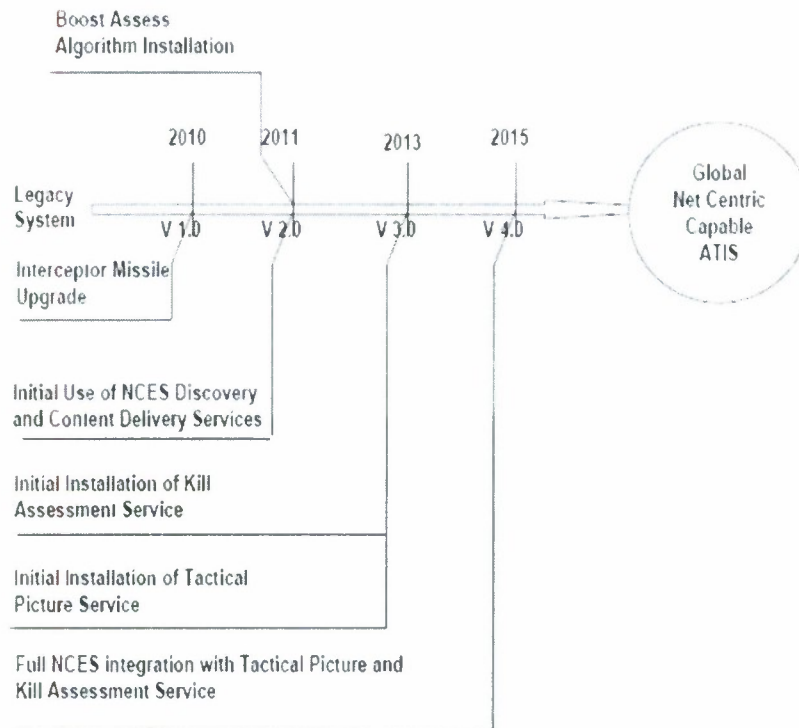


Fig. 5.34: SV-8, Systems/Service Evolution Description.

DISR Standard	0-12 Months	12-36 Months	38-60 Months	60+ Months
NCES Discovery Service	Initial Capability	Upgrade	Upgrade 2	Full Discovery Service
NCES Content Delivery Service	Initial Capability		Full Capability	
New Joint Tactical Radio	Initial Capability		Full Capability	
Hi Speed Sensor Technology for Missile	Initial Capability	Increased Sensitivity	Dual Phenomenology	Upgrade

Fig. 5.35: SV-9, Systems/Services Technology Forecast.

Once the CPN model was created, it was executed to check the logical correctness of the Operational View architecture description. As errors in it were detected, corrections were made in the CPN model and the architecture. Then the architecture was tested for various sets of inputs to be sure that the behavior of the architecture was satisfactory. This included checking the behavior of the system against both Adversary A and Adversary B. The sequencing of the

events, messages, and activities was checked and compared with the descriptions in the OV-5 and -6b and c. Again necessary changes were made to both the architecture description and the CPN model. The architecture description presented in this paper contains the architecture in its corrected form.

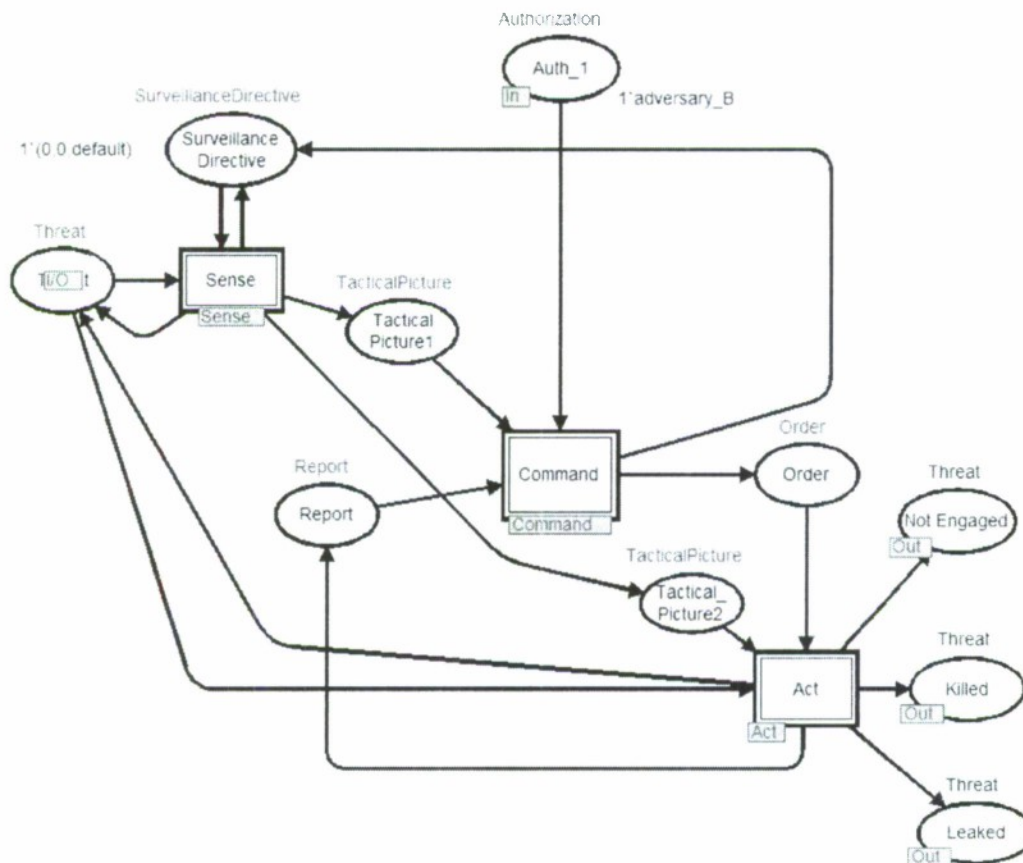


Fig. 5.36: CPN Model of the ATIS Operational View.

Once the initial logical and behavioral correctness was verified, a scenario was established to address the question about the capability of the ATIS against the adversaries. It was assumed that the adversaries would have a capability to launch multiple TBMs. It also was assumed that the ATIS system would be deployed and would have sufficient warning of potential adversary action to have all of its systems in place including the interceptor aircraft. Two questions to be addressed: (1) can the ATIS system issue the appropriate commands and data so that it can shoot down the TBMs that may be launched and (2) how many interceptors will be required to handle various threat capabilities?

The scenario was parameterized as follows. We assumed that that the ATIS system must be able to launch its interceptor missile at the TBM within 400 s of initially detecting the TBM with the ATIS Radar. If it takes more than 400 s (6 min 40 s) to launch the interceptor missile, the TBM will be out of range and will be declared as a "leaker." Indeed we would like the average response time to be less than 400 s.

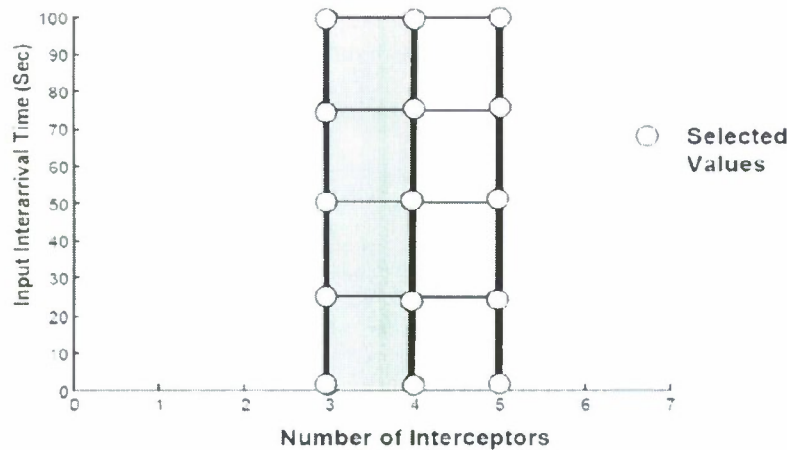


Fig. 5.37: Parameter Locus.

To address these questions it was necessary to set the CPN model up as a timed CPN. This is because the performance of the ATIS would be based on the number of successful intercepts it could make against multiple TBMs. Whether an intercept is successful or not is based on the amount of time it takes for the interceptor to get into position and for the ATIS system to issue the order to launch the interceptor missile. Time delays were estimated for each operational activity and applied as delays for each transition that represented an operational activity. These estimates were based on the notion that the systems, system functions, and services supporting the operational activities would be able to accomplish each task in a few seconds (see SV-7, Fig. 5.33). Additional time delay was added to each activity (transition) to account for operator reactions. For our initial analysis, we assumed that communications delays would be negligible compared to the processing and human decision making delays and therefore zero time delay for the communications network was assumed. Indeed the communications network was not modeled explicitly. The approach for modeling explicitly the communications network and linking it to the executable model of the architecture was described in Shin and Levis [2003].

Input, ATIS System, and Output variables were established for the scenario. The inputs to the ATIS model included the individual TBMs and the authorization message. For the TBMs the variables were the total number of TBMs and the time interval between TBM arrivals. For the Authorization variable the values were Adversary A and Adversary B. For the ATIS system the number of interceptors was the main variable. The output variables were four Measures of Performance (MOPs): Average Response Time (in seconds), Throughput Rate (kills per second), the number of kills (integer), and the number of leakers (integer).

To evaluate the potential performance the total number of TBMs fired was fixed at 10, and two parameters were varied. The TBM inter-arrival time was varied between 0 and 100 s, and the number of interceptors was varied between 3 and 5 (integer) interceptors. The range of these two parameters can be viewed as a parameter locus as shown in Figure 5.37. The 15 input pairs shown as the small circles indicate that values that were used for the analysis. Note that the TBM

inter arrival time is a continuous variable while the number of interceptors is a discrete variable. The area between the points has been shaded to aid in visualization.

The executable CPN was run in simulation mode at each of the 15 points in the parameter locus. The CPN has been set up so it records the time each TBM entered the ATIS system and the time that the intercept occurred. The CPN determines for each TBM if the intercept occurred within the 400 s requirement. If it did, it was considered a kill, and if not it is a leaker. Collecting these data enables the calculation of the values of the average time for intercept (average response time) and the throughput rate MOPs. Figure 5.38 shows an example of the results of a simulation run. The values on the tokens show the TBM number, the interceptor number, the time the TBM entered the ATIS system, and the time that the interceptor missile was fired. The figure shows that with four interceptors and 10 TBMs that arrive within the ATIS area every 20 s, 8 TBMs are successfully intercepted and 2 are not.

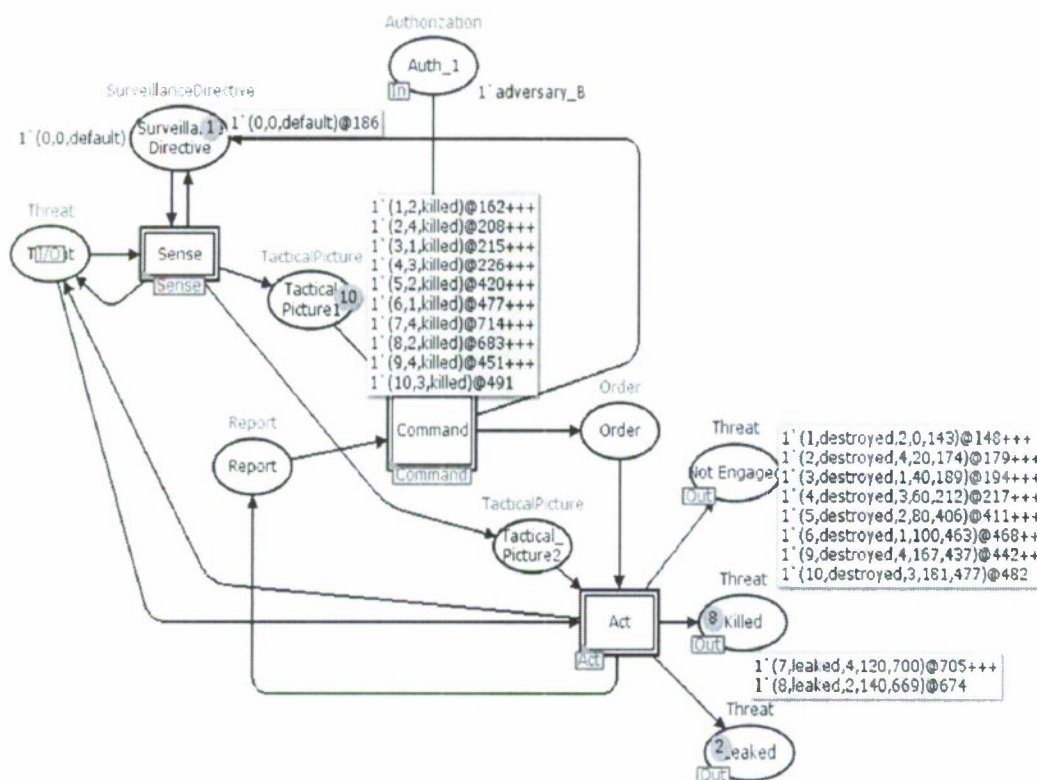


Fig. 5.38: Simulation Run (20 second interarrival time; 4 interceptors).

The results of the 15 simulation runs are plotted as a performance locus in a 3-dimension performance space as shown in Figure 5.39. The values of three key MOPs were calculated from the data in the simulation runs: average time per intercept, throughput rate, and number of leakers. Requirements were established for these MOPs (no more than two leakers and maximum allowed average intercept time of 400 s). The projection of the performance locus onto the Leaker—Average Response Time plane is shown in Figure 5.40. The combination of MOPs and requirements can be visualized in a plot showing a Requirements Locus overlaid on the projection of Figure 40, as shown in Figure 5.41. To help determine the potential effectiveness,

the mapping of the intersection of the requirements locus and the performance locus onto the parameter locus was accomplished as shown in Figure 5.42. A Measure of Effectiveness (MOE) value of 75% was calculated based on the percentage of the parameter locus that would result in meeting the performance requirements.

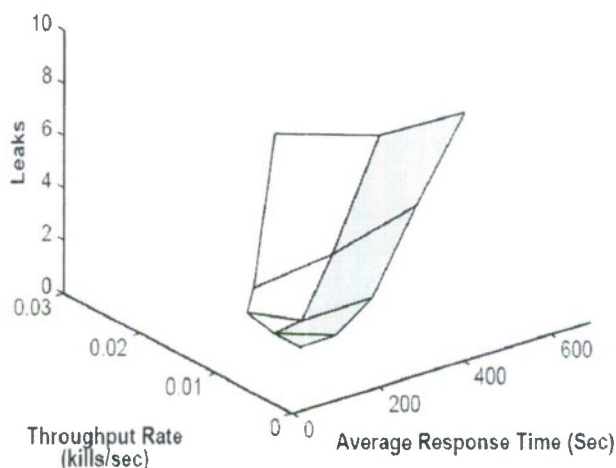


Fig. 5.39: Performance Locus for Simulation Run (20 second interarrival time; 4 interceptors).

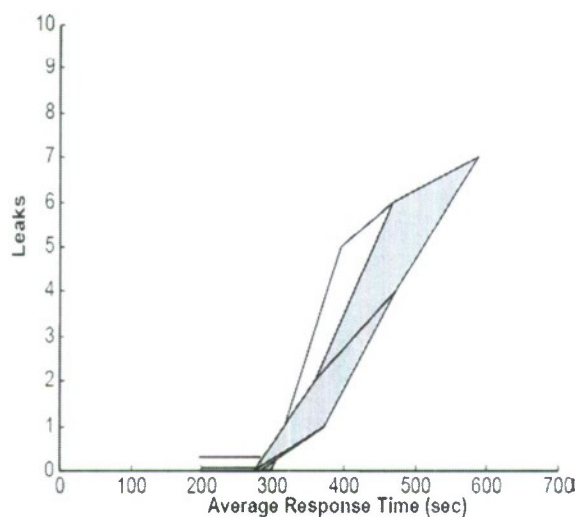


Fig. 5.40: Projection of Performance Locus onto the Leaks/Average Response Time Plane.

Additional results of the analysis are summarized as follows:

- Three interceptors can handle the 10 threats (with a max of 2 leakers) if they arrive at a rate slower than 1 in 45 s.
- Four interceptors can handle the 10 threats (with a max of 2 leakers) if they arrive at a rate slower than 1 in 20 s.
- Five interceptors can handle the 10 threats (with a max of 2 leakers) if they arrive at a rate slower than 1 in 10 s.

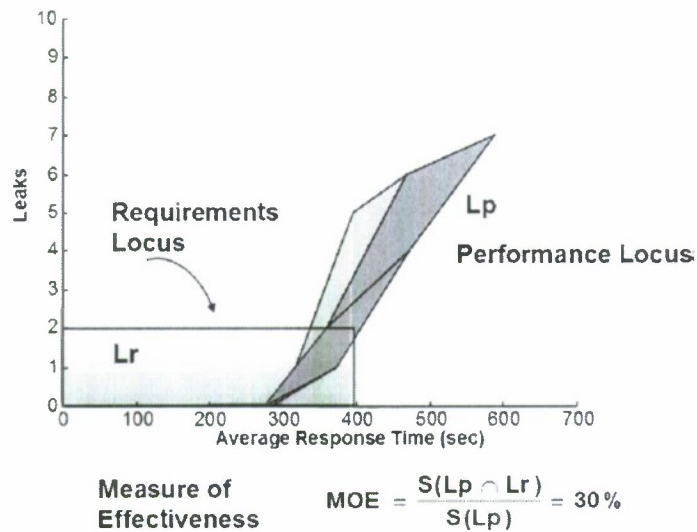


Fig. 5.41: Requirements Locus Superimposed on Performance Locus.

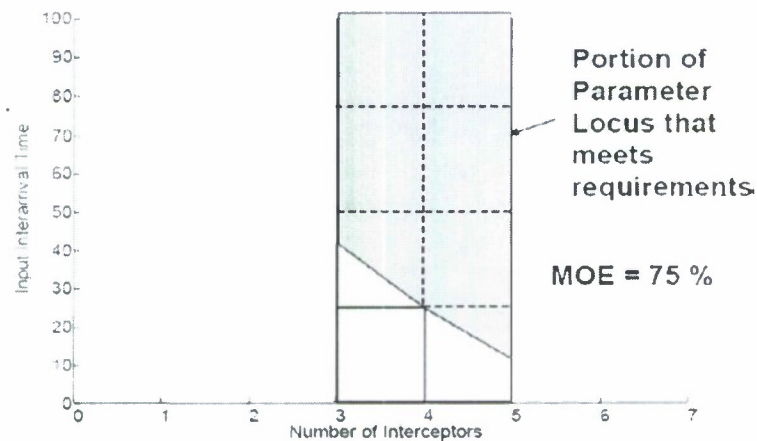


Fig. 5.42: Projection of Requirements Locus onto Parameter Locus.

A similar set of results was determined against Adversary A.

These results are based on the CPN executable model of the operational view. They are based on estimates of how long it will take to accomplish the operational activities. This time will be based on a combination of the time it takes for humans to interact with the systems that are supporting them and on the processing and communications delays of the systems, services, and communications networks that are described in the Systems and Services View. For our executable model we made a rough order of magnitude estimate of these time delays. An executable model of the Systems and Services view could be created using the same techniques that was used of the Operational View to better understand the system and communications time delays. Executable models of the alternative Systems and Services Views (such as the peer-to-peer architecture or the one based on Services) could be made to determine if these architectures can meet the requirements.

5.5. Comments and Conclusion

We have illustrated three interrelated processes for developing DoDAF 1.5 compliant architectures, deriving a CPN executable model from that architecture data, and using the CPN model to support analysis and evaluation. The techniques described allow for a much more thorough examination of the behavior and performance aspects of the architecture than what can be done using only the architecture description. Conducting this type of analysis and evaluation is becoming more critical as DoD continues its migration to net centricity with SOA as the postulated solution. However, the move to SOA increases the complexity (there are many moving parts). Research is underway to develop techniques for creating executable models that will examine behavior and performance with SOA. Various Communities of Interest (COIs) are considering SOAs. The desire to enable ubiquitous information sharing across domains and COIs implies the possibility of Federated SOAs which will require further investigation of behavior and performance. There are many ways to implement a SOA; there are many choices to be made. These choices can be informed by rigorous analysis and evaluation. A high level of abstraction has been used for our illustration. It is important not to go into too much detail early in the process as it must be shown that the abstract concepts work before investing a lot of time in more detailed descriptions. Evaluation becomes more complex as the communications infrastructure is considered in the systems view. To deal with this complexity, layered approaches that involve interconnecting the CPN executable with network simulators have been demonstrated [Shin and Levis, 2003]. These capabilities will enhance evaluation.

The process for conversion of the architecture description to the executable model is well understood. However, there is a need for improved tools that can generate the executable automatically from the architecture data which will make this step even easier. Some in the software community have been developing techniques (e.g., xUML) that will enable the automatic generation of executable models from UML representations [Mellor and Balcer, 2002]. The improved semantics and meta model of UML 2.1.1 are enablers of this automatic executable model generation. Recently, Liles [2008] developed and demonstrated a capability incorporated in Rational System Developer that enables the automatic generation of the CPN model (for CPNTools) from a UML architecture description using techniques similar to the ones described in this paper. If tool vendors will provide this mechanism in their tools, then the architect can have the executable model generated automatically as the architecture is developed, provided that the architect develops the complete description of the architecture in UML as required for the conversion. This will enable architects to “experiment” with the architecture description as it is developed. It is important to be sure that the execution model “engine” handles issues of concurrency and conflict properly; not all simulation engines do. This is why we have chosen CPN as the executable model.

Generation of the executable model is a necessary step in the detailed analysis and evaluation of the architecture description, but it is not sufficient. A basic executable model can help the architect check the logic and some aspect of behavior, but addressing more complex performance issues requires additional data to be incorporated in the executable as was illustrated in our case

study. It would be desirable to have evaluation techniques included in the tools. Education and training of practitioners in the concepts of the architecture descriptions and the use of the executable model along with training on the use of specific tools are also needed. Architectures, if done properly in a layered way, are a great tool for innovative design. They allow the exploration of radical alternatives in a short amount of time, thus expanding the number of alternatives to be considered. They provide a way—using executable models—for new ideas to be explored. However, we have not developed adequate algorithms, tools, and techniques to support the use of architectures in concept development and in considering many (including radical) alternatives. We understand the fundamentals. There is sufficient theory in mathematics and computer science as well as modeling & simulation technology that should be exploited for this class of problems. We need to refocus our efforts and develop applications that will take us to the desired end: an efficient, architecture based systems engineering process that enables us to integrate legacy systems with new systems and exploit technology advances to provide desired capabilities to the users.

5.6 References

- CPNTools, <http://wiki.daimi.au.dk/cpntools/cpntools.wiki>, 2008.
- J.R. Hurwitz, C. Bloor, C. Baroudi, and M. Kaufman, *Service oriented architectures for dummies*, Wiley, Hoboken, NJ, 2007, p. 27.
- L.M. Kristensen, S. Christensen, and K. Jensen, The practitioner's guide to coloured Petri nets, *Int J Software Tools Technology Transfer* 2 (1998), 98–132.
- S.W. Liles, *On the characterization and analysis of system of systems architectures*, Ph.D. Dissertation, George Mason University, Fairfax, VA, 2008.
- S.J. Mellor and M.J. Balcer, *Executable UML: A foundation for model-driven architecture*, Addison-Wesley, Reading, MA, 2002.
- I. Shin and A.H. Levis, Performance prediction of networked information systems via Petri nets and queuing nets, *SystEng* 6(1) (2003).
- Universal Joint Task List (UJTL), CJCSM 3500.04C, Joint Chiefs of Staff, Department of Defense, Washington, DC, 2002.
- F.R.H. Valraud and A.H. Levis, "On the quantitative evaluation of functionality in C3 systems," *Information technology for command and control*, S.J. Andriole and S.M. Halpin (Editors), IEEE, New York, 1991, pp. 558–563.
- L.W. Wagenhals, S Haider, and A.H. Levis, Synthesizing executable models of object oriented architectures, *SystEng* 6(4) (2003), 266–300.
- L.W. Wagenhals, I. Shin, D. Kim, and A.H. Levis, C4ISR architectures II: A structured analysis approach for architecture design, *Syst Eng* 3(4) (2000), 225–247.

SECTION 6

Analysis and Evaluation of System of Systems Architectures¹

Stewart W. Liles and Alexander H. Levis

6.1 Introduction

Agility is a necessary response to uncertainty. If planners do not know what to expect then the plan must address a much broader set of contingencies than when there is no uncertainty. [Alberts and Hayes, 2007] Inherent to Alberts' and Hayes' comments on agile planning, is the need for systems to possess the ability to adapt to an operating environment that may be significantly different from the one for which they were originally designed. One way to address the agility issue, is to build systems that are composed of different types of systems and components that operate together to accomplish the tasks required by the organization - a system of systems (SOS). An approach for studying the effects of design decisions and modeling the capabilities required by the organization is to produce an architecture that describes the interactions of constituent systems used to provide capabilities to the organization. The goal is to produce an architecture that will satisfy the needs of the customer by providing multiple capabilities concurrently and possess the ability to adapt its structure to unforeseen operating environments.

This research addresses the design and development of SOS solutions very early in the development process to assess the ability of the SOS to adapt to structural configurations for which it was not originally designed. The methodology uses model-driven development techniques to combine multiple operational and system architectures into a combined architecture that represents the attributes of the SOS implementation. That SOS implementation is then transformed into a dynamic model that enables an analysis of the interaction of the constituent systems of the SOS.

A challenge to system engineers when developing SOS solutions is analyzing characteristics that assess the aggregate performance of the SOS. Most SOS definitions focus on the managerial aspects rather than technical aspects of the SOS. System engineers need measures and characteristics that can be assessed early in the development process in order to contribute to analyses of alternative SOS architectures. When defined as described, SOS engineers tend to measure constituent system characteristics and aggregate those measures for the architecture as a whole. This leads to bounding the problem by defining particular operating scenarios and optimizing configurations for a particular scenario. This optimization can result in SOS configurations that are not able to adapt to unpredictable operating environments. "The wide

¹ This section consists of the slightly edited Ph.D. thesis of LCOL Stewart W. Liles, USA.

range of threats faced today, their dynamic nature, and the complexity of the environments in which they must be defeated make it imperative to avoid ‘optimizing’ (perhaps more clearly said, ‘fixating’) on an approach that handles only one type of threat or situation well” [Alberts and Hayes, 2007]. The methodology presented here assesses alternative SOS implementations for SOS characteristics that address the interaction of the constituent systems and the ability of the architecture to provide multiple capabilities for the organization.

System of Systems

Figure 6.1 illustrates the enormity of the Department of Defense (DOD) SOS as an example of the complexity of some organizations. This is a partial list of the resources available to the DOD enterprise. It is futile to model the interactions among the constituent systems of the SOS in their entirety. The environments each can be deployed in are diverse and virtually unpredictable. The various configurations cannot be accurately predicted and the potential adversaries have not been defined. Additionally, technological advances add further uncertainty to the deployed environment. Finally, the organizational structure is unpredictable given the uncertainty of the factors already mentioned.

An extended definition of SOS specifies the resources available to the enterprise and differentiates specific implementations used for particular purposes. The resources available to the enterprise compose the SOS. The resources are used by the enterprise to realize specific capabilities required by the organization. Identifying the specific implementation provides a structure on which to make measurements. It also provides a way to identify alternatives for comparison. A specific implementation requires a set of resources that are configured to provide a specific set of capabilities to the organization. The specific implementation is developed using the specification of the structural and behavioral relationships between resources defined in the architecture. The specific implementations of the architecture can be assessed for their ability to address the needs of the organization. The specific properties of the SOS will be detailed in Section 6.3.

Architecture Modeling

The goal of the SOS engineer is to demonstrate to the organization that a SOS architecture will meet the needs of the organization. In construction engineering, the vehicle for demonstration would be a paper model or a 3D computer generated representation. For the system engineer the vehicle for demonstration is an executable model that can represent the dynamic nature of the interacting systems modeled by the architecture. Operational architectures describe organizational roles that interact to provide a particular capability. System architectures describe a physical implementation that can be used to realize the capability. Current architecture modeling techniques tend to focus on the single system or single capability. However, a SOS architecture must describe multiple concurrently executing capabilities. While modeling a realization of a capability at the system level may require the use of multiple systems, engineers rarely model the multiple capabilities that a particular implementation must realize. The methodology uses operational and system architecture data to produce a combined SOS

architecture representation. The SOS architecture is used to create a specific implementation for analysis. The specific implementation is transformed into an executable form that enables the analysis of architecture alternatives in a static and dynamic environment. The measures developed for the methodology assess the ability of the architecture to adapt to configurations other than one for which it was designed.

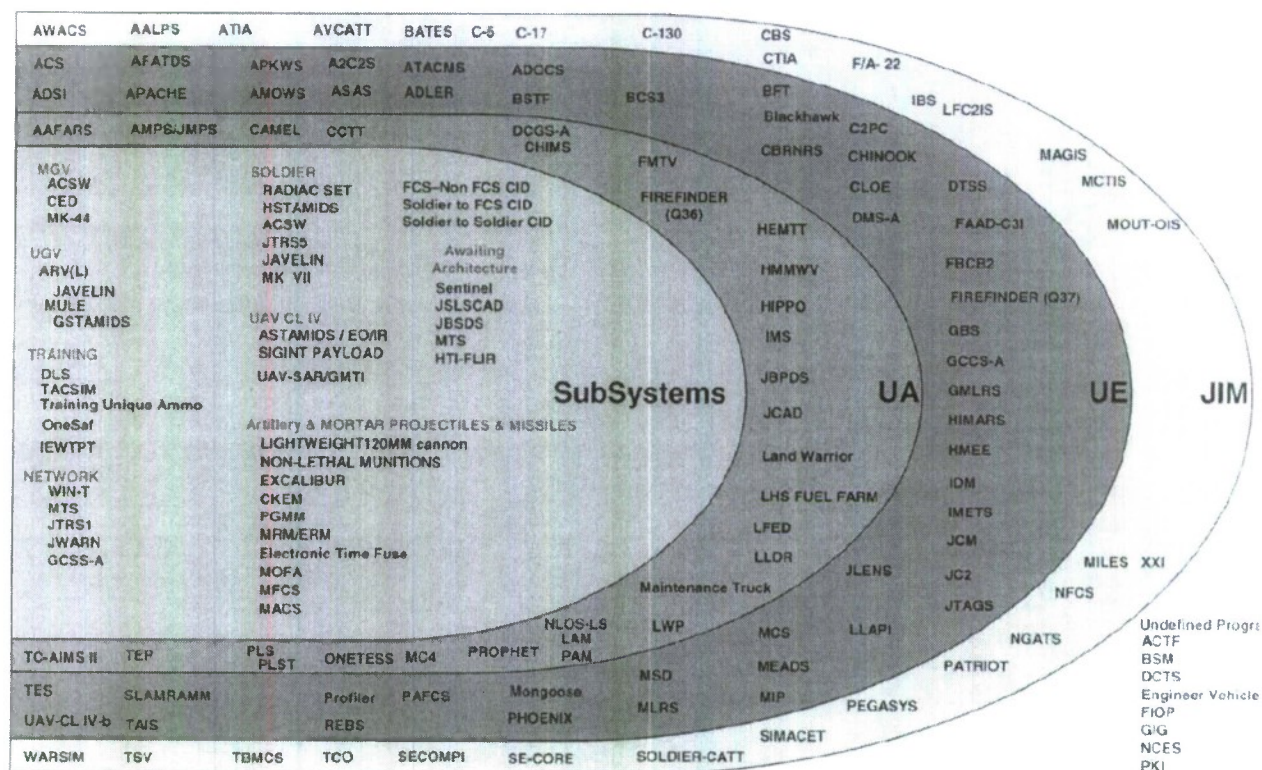


Fig. 6.1. System of Systems [Brown, 2005]

The methodology presented effectively models the interaction of constituent systems of the SOS and creates a boundary that allows the creation of multiple alternatives for comparison. This research focuses on the interaction between constituent systems and the nodes they occupy. Additionally, a method to capture the dynamic nature of the SOS at an architectural level is to produce an executable model from the architectural model. This research, then, uniquely characterizes the SOS and describes a methodology for assessing candidate architectures using the SOS measures Adaptability and Agility.

The problem statement for this research is: To develop a methodology for measuring and evaluating a set of characteristics that uniquely describe a system of systems.

Hypothesis: The performance measures Adaptability, Agility, and Degree of Reuse enable the comparison of alternative architectures for their ability to adapt to unforeseen configuration as the requirements of the organization change.

There are three primary contributions of this research: (1) Analytical measures that describe unique technical aspects of the system of systems; (2) Methodology for combining behavior models to create an executable model for analysis; and (3) SOS architecture assessment of SOS characteristics given the analytical measures and the methodology.

This section is organized around the technologies required to implement the concepts described above. Section 6.2 presents related research in the primary domains that were used to address the problem. Section 6.3 presents the SOS characteristics that were developed to assess the SOS architecture alternatives. Section 6.4 presents the methodology used to develop and assess the SOS architecture alternatives using the measures described in Section 6.3. Section 6.5 presents the process for transforming the static architecture representation into a dynamic representation for analysis. Section 6.6 details a case study that provides evidence concerning the validity of the assessment methodology, SOS characteristics, and the executable models used to evaluate them. Section 6.7 concludes with the contributions of the methodology and ideas for future research concerning SOS architecture development and analysis.

6.2 Related Work

The primary product of the methodology presented here is a formally defined executable model that enables a static analysis of the graph representing the executable model and a dynamic analysis that uses simulation results from the model. Each section of this chapter addresses a particular concept or technology that is used by the methodology to facilitate the creation of a dynamic representation of the SOS architecture for the purposes of assessing specific attributes that affect its ability to adapt to unpredicted structural configurations.

System of Systems

There are many perspectives on what constitutes a System of Systems (SOS). This section discusses some of the more common definitions. While they are workable definitions, they tend to address a SOS's managerial aspects rather than its technological aspects. While managerial aspects are important, this research is focused on the technological aspects.

Maier [1996] offers five SOS characteristics (listed in **Error! Reference source not found.**). They are: Operational Independence, Managerial Independence, Evolutionary Development, Emergent Behavior, and Geographic Distribution.

The Defense Acquisition Guidebook states that the objective of SOS engineering is to satisfy capabilities that can only be met with a mix of multiple, autonomous, and interacting systems. The mix of constituent systems may include existing, partially developed, and yet-to-be-designed independent systems. [DAU, 2006] Additionally, Sage and Cuppan [2001] offer a comprehensive paper on the subject of SOS management in which they address the characteristics of the SOS and differentiate between a SOS and a federation of systems (FOS). The discussion is mentioned here to highlight that what constitutes a SOS is much in the eye of the beholder. A formal definition of a SOS is offered in Section 6.3.

Table 6.1. System of Systems (SOS) Characteristics [Maier, 1996]

Operational Independence of the Elements: If the system-of-systems is disassembled into its component systems the component systems must be able to usefully operate independently. The system-of-systems is composed of systems which are independent and useful in their own right.
Managerial Independence of the Elements: The component systems not only can operate independently, they do operate independently. The component systems are separately acquired and integrated but maintain a continuing operational existence independent of the system-of-systems.
Evolutionary Development: The system-of-systems does not appear fully formed. Its development and existence is evolutionary with functions and purposes added, removed, and modified with experience.
Emergent Behavior: The system performs functions and carries out purposes that do not reside in any component system. These behaviors are emergent properties of the entire system-of-systems and cannot be localized to any component system. The principal purposes of the systems-of-systems are fulfilled by these behaviors.
Geographic Distribution: The geographic extent of the component systems is large. Large is a nebulous and relative concept as communication capabilities increase, but at a minimum it means that the components can readily exchange only information and not substantial quantities of mass or energy.

Maier's characteristics and those offered by Sage and Cuppan are appropriate to manage and acquire a SOS, but are not very informative in SOS testing and performance analysis. This research will attempt to answer part of the question of what, technically, must be modeled in order to accurately reflect the behavior and interaction among individual systems in the SOS. After all, it is the interaction among the constituent systems that provides synergistic or emergent behavior that is thought to be a SOS's primary characteristic.

A shortfall of the above definitions is that they fail to bound the SOS is a way that allows SOS engineers to measure aggregate characteristics. When defined as described, SOS engineers tend to measure constituent system characteristics and aggregate those measures for the architecture as a whole. This leads to bounding the problem by defining particular operating scenarios and optimizing configurations for a particular scenario. This optimization can result in SOS configurations that are not able to adapt to unpredictable operating environments. "The wide range of threats faced today, their dynamic nature, and the complexity of the environments in which they must be defeated make it imperative to avoid 'optimizing' (perhaps more clearly

said, ‘fixating’) on an approach that handles only one type of threat or situation well” (Alberts and Hayes, 2007).

SOS Taxonomy

SOS is an emerging research area. Because there is no generally accepted set of attributes that characterize a SOS, it is difficult to describe where research fits in the SOS domain. For example, when discussing a SOS acquisition is the research addressing managerial aspects of the SOS, as described by Maier, or structural aspects as described by DeLaurentis [2005] in his taxonomy. While not comprehensive, it does provide a start that will be built upon as the research domain matures. The DeLaurentis taxonomy is summarized in Table 6.2.

Table 6.2. Taxonomy for Describing a System of Systems [DeLaurentis, 2005]

Category	Description
Resources	The entities (systems) that give physical manifestation to the system-of-systems
Stakeholders	The non-physical entities that give intent to the SOS operation through values
Operations	The application of intent to direct the activity of physical and non-physical entities
Policies	The external forcing functions that impact the operation of physical and non-physical entities
Level	Description
Alpha (α)	Base level of entities in each category, further decomposition will not take place.
Beta (β)	Collections of α -level systems (across categories), organized in a network.
Gamma (γ)	Collections of β -level systems (across categories), organized in a network.
Delta (δ)	Collections of γ -level systems (across categories), organized in a network.

While Maier addresses managerial aspects and DeLaurentis addresses structural aspects, this research concentrates on the SOS’s technical characteristics. The SOS is composed of structural and behavioral characteristics that must be included in the model to accurately represent the SOS’s dynamic characteristics. A specific definition is used by the methodology to express the difference between a system and SOS. While DeLaurentis offers a hierarchical approach to

differentiate specific instances of the SOS (Fig. 6.2), the methodology presented defines the SOS in terms of a set of resources that provide a specific set of capabilities to the organization. An instance of a SOS architecture might display certain aspects of this taxonomy, but the SOS architecture is not a static hierarchical structure. The methodology presented provides an extended definition of a SOS that differentiates the SOS from a specific implementation of a SOS architecture.

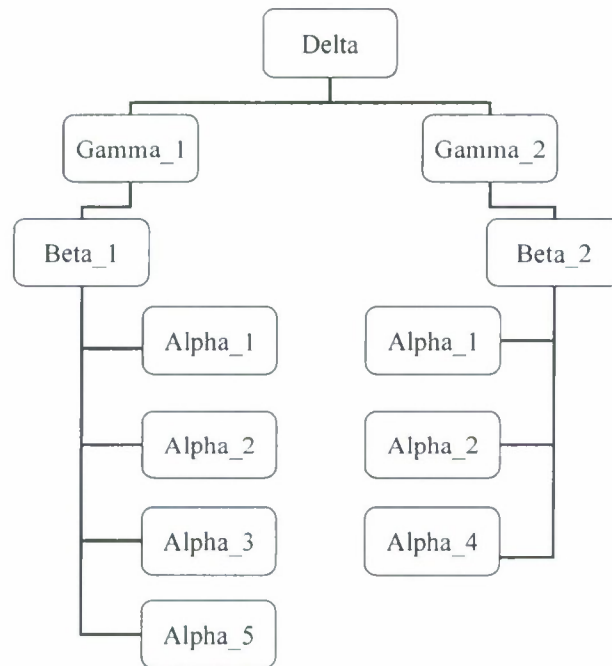


Figure 6.2. Graphical View of DeLaurentis Taxonomy

Architecture Modeling

The goal of the SOS engineer is to demonstrate to the organization that a SOS architecture will meet the needs of the organization. In construction engineering, the vehicle for demonstration would be a paper model or a 3D computer generated representation. For the system engineer the vehicle for demonstration is an executable model that can represent the dynamic nature of the interacting Elements modeled by the architecture. This section provides an overview of the technology used to create the executable models that will enable the assessment of the SOS architectures.

Levis and Wagenhals [2000] describe the information that must be available in the architecture to accurately create an executable model. “To obtain a specification of the architecture that allows the derivation of the executable model, an activity model, a data model, a rule model, and a dynamics model are required.” The executable model can also be a tool for

modeling concurrently executing behavior; therefore it is important that the architecture be complete enough to create an executable and that the behavior and data represented in the executable model be traceable to the architecture representation.

Operational architectures describe organizational roles that interact to provide a particular capability. System architectures describe a physical implementation that can be used to realize the capability. Current architecture modeling techniques tend to focus on the single system or single capability. However, a SOS architecture must describe multiple concurrently executing capabilities. While modeling a realization of a capability at the system level may require the use of multiple systems, engineers rarely model the multiple capabilities that a SOS implementation must realize. Rechtin and Maier [1996] and again Rechtin [1991 and 1992] offer detailed system engineering approaches that integrate multiple components, but they do not address a SOS development environment. The following sections discuss languages and frameworks that assist engineers in the development and assessment of SOS architectures.

DODAF

The Department of Defense Architecture Framework (DODAF) provides a framework for representing both operational and system architectures.

“The Framework provides the guidance, rules, and product descriptions for developing and representing architecture descriptions that ensure a common denominator for understanding, comparing, and integrating Families of Systems (FOs), Systems of Systems (SOSs), and interoperating and interacting architectures.” [DODAF, 2007a]

The DODAF uses a series of products to represent the architecture. The products are first divided into 4 categories: the Operational View, the System View, the Technical Standards View, and the All Views. The Operational Views primarily address the operational nodes and the data that must pass between them for operational success. The System Views address the specific physical systems that support the exchange of information between operational nodes. The Technical Standards Views describe the technological standards that will constrain the physical system design. The All Views describe those overarching aspects that apply to all three views. For example, they set the architecture’s scope and context. The DODAF documents provide a comprehensive explanation of each architecture product. As this research addresses aspects of the SOS, the appropriate DODAF product will be discussed in that context.

Tables 6.3, 6.4, 6.5, and 6.6 show the various DODAF Architecture View products: All View, Operational View, System View, and Technical Standards View.

Table 6.3. All View Products [DODAF, 2007b]

Product	Framework Product Name	General Description
AV-1	Overview and Summary Information	Scope, purpose, intended users, environment depicted analytical findings
AV-1	Integrated Dictionary	Architecture data repository with definitions of all terms used in all products

Table 6.4. Operational View Products [DODAF, 2007b]

Product	Framework Product Name	General Description
OV-1	High-Level Operational Concept Graphic	High-level graphical/textual description of operational concept
OV-2	Operational Node Connectivity Description	Operational nodes, connectivity, and information exchange need lines between nodes
OV-3	Operational Information Exchange Matrix	Information exchanged between nodes and the relevant attributes of that exchange
OV-4	Organizational Relationships Chart	Organizational, role, or other relationships among organizations
OV-5	Operational Activity Model	Capabilities, operational activities, relationships among activities, inputs, and outputs; overlays can show cost, performing nodes, or other pertinent information
OV-6a	Operational Rules Model	One of three products used to describe operational activity—identifies business rules that constrain operation
OV-6b	Operational State Transition Description	One of three products used to describe operational activity—identifies business process responses to events
OV-6c	Operational Event-Trace Description	One of three products used to describe operational activity—identifies business process responses to events
OV-7	Logical Data Model	Documentation of the system data requirements and structural business process rules of the Operational View

Table 6.5. System View Products [DODAF, 2007b]

Product	Framework Product Name	General Description
SV-1	Systems Interface Description	Identification of systems nodes, systems, system items, and their interconnections.
SV-2	Systems Communications Description	Systems nodes, systems, system items, and their related communications.
SV-3	Systems-Systems Matrix	Relationships among systems in a given architecture.
SV-4	Systems Functionality Description	Functions performed by systems and the data flows among system functions
SV-5	Operational Activity to Systems Function Traceability Matrix	Mapping of system functions to operational activities
SV-6	Systems Data Exchange Matrix	Details of system data elements being exchanged between systems and the attributes of that exchange
SV-7	Systems Performance Parameters Matrix	Performance characteristics of SV elements for the appropriate time frame
SV-8	Systems Evolution Description	Planned incremental steps toward migrating a suite of systems to a more efficient suite, or toward evolving a current system to a future version
SV-9	Systems Technology Forecast	Emerging technologies and software/hardware products will affect future development of the architecture
SV-10a	Systems Rules Model	Identifies constraints that are imposed on system functionality due to some aspect of system design.
SV-10b	Systems State Transition Description	Identifies responses of a system to events
SV-10c	Systems Event-Trace Description	Identifies system specific refinements of critical sequences of events described in the Operational View
SV-11	Physical Schema	Physical implementation of the Logical Data Model entities

Table 6.6. Technical Standards View Products [DODAF, 2007b]

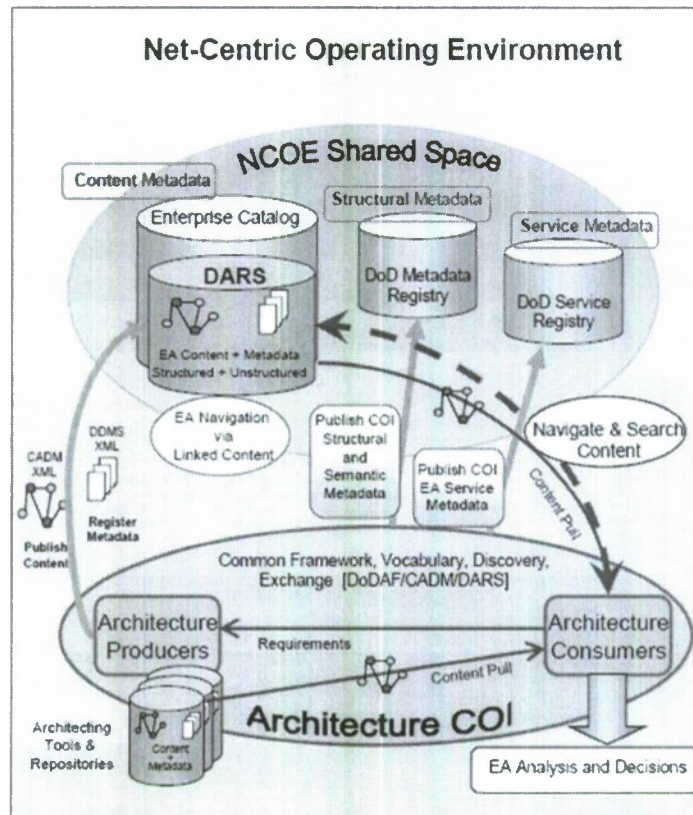
Framework Product	Framework Product Name	General Description
TV-1	Technical Standards Profile	Listing of standards that apply to Systems and Services View elements in a given architecture
TV-1	Technical Standards Forecast	Description of emerging standards and potential impact on current Systems and Services View elements, within a set of time frames

To support the representation of DODAF architectures in a methodology independent way, the DOD has developed the Core Architecture Data Model (CADM). CADM facilitates the data-centric environment by providing the data model for all data in the DODAF, including metadata about the architecture to facilitate interoperability and reuse of architecture data. The CADM enhances the DODAF through increased interoperability and reuse. The CADM is a primary enabler for the common framework, vocabulary, discovery, and exchange of architecture information.

The CADM provides a structure on which DODAF architectures can be stored and referenced by SOS engineers. The repository that catalogs architectures that are completed and in development is called the DOD Architecture Registry System.

“The DOD Architecture Registry System (DARS) provides for registration and linking of architecture metadata to enable the creation of a navigable and searchable enterprise architecture. It enforces the policies and governance that surround the usage of architecture, thus reinforcing robust interfaces and data relationships.” [DODAF, 2007]

The CADM is a primary enabler for the DARS by providing the data model for information stored in or referenced by the registry. The information exchange mechanism of architecture data is the CADM XML. The use of the CADM XML and the architecture metadata allow the registry of architecture data to be a significant asset in the Net-Centric Operating Environment (NCOE). The NCOE is the networked shared space that facilitates the interoperability of systems acquired by the DOD. Figure 6.3 illustrates how architecture producers register the developed architecture data in the DARS. The structure allows consumers to access the data to conduct enterprise architecture analysis and aid decision making. The DARS provides a mechanism that allows SOS developers to access the models that represent the constituent systems that they will use to build the SOS. It is these system representations that will be the basis for developing SOS architecture alternatives for comparison.



EA = Enterprise Architecture
 COI = Community of Interest
 CADM = Core Architecture Data Model
 DDMS = Defense Discovery Metadata Specification

Figure 6.3. Department of Defense (DOD) Architecture Registry System (DARS) Role in the Net-Centric Operating Environment (NCOE) [DODAF, 2007]

Unified Modeling Language

The DODAF does not prescribe a language for the representation of architecture data. Rather it defines the type of data that must be present for each architecture view product. The methodology presented here uses the Unified Modeling Language (UML) to represent the architecture and uses the DODAF products to provide a common view of that data.

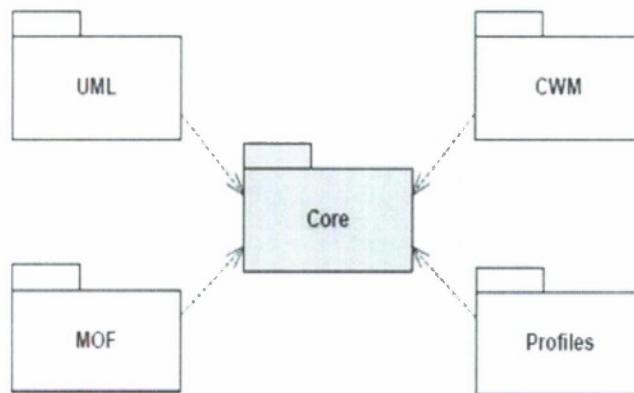
The Unified Modeling Language (UML) superstructure specification states that “the Unified Modeling Language is a visual language for specifying, constructing, and documenting the artifacts of systems” [Object Management Group (OMG), 2007b]. UML is primarily a graphical language that uses specific modeling artifacts to illustrate a system’s structural and behavioral aspects. The UML specification defines both structural and behavioral diagrams used to describe various aspects of a system. The Structural Diagrams include the Use Case Diagram, Class Diagram, Object Diagram, Component Diagram and Deployment Diagram. Behavior Diagrams include the Sequence and Communication Diagrams, the Activity Diagram and the State Machine Diagram. Fowler [2004] and Eriksson, et al. [2004] both offer descriptions of each

diagram and the purpose for each. The methodology presented makes extensive use of Activity Diagrams to model the behavior of constituent systems of the SOS and the interaction of those systems.

The UML is a formal language with a complete data model that facilitates an abstract syntax that describes all the components of the diagrams mentioned above. The component relationships that exist across diagrams are maintained by representing the common components in a high level data model. The high level model, or meta model, describes the UML components that are instantiated to create the relationships and behavior modeled in the diagrams. The meta model provides a standard representation that is used to facilitate transformations to other model representations and for exchanging model data between tools using XML files built to the UML specification. Additionally, the UML semantics describe the meaning conveyed by the interaction of model elements in both the structural and behavior diagrams. The UML uses a detailed architecture to describe the language and its interactions. The next section offers an overview of the UML language architecture. The UML Infrastructure and Superstructure Specification [OMG, 2007a, 2007b] has a more detailed explanation of the UML specification. This research makes extensive use of the UML meta model to facilitate a transformation from UML to an executable representation.

The UML designers use modularity, layering, and partitioning to facilitate the UML's extensibility and reuse. The UML has two primary layers: the infrastructure layer and the superstructure layer. The Core packages of the infrastructure describe highly re-useable constructs that are used throughout the superstructure layer. The superstructure reuses the constructs of the infrastructure to create the top-level constructs that are used every day by modelers. Figure 6.4 shows an example of this reuse by illustrating the use of the Core packages in other modeling languages sponsored by the OMG, and shows how the Core packages underlie the Meta Object Facility (MOF), the Common Warehouse Model (CWM), the UML, and Profiles. The items relevant to this research are the MOF and UML. The Core underlies the Meta-Object Facility (MOF). The UML and other languages are described using the common meta-model MOF [OMG, 2007a]. There is a recursive relationship between the MOF and UML because the Core packages from UML are used to describe the MOF.

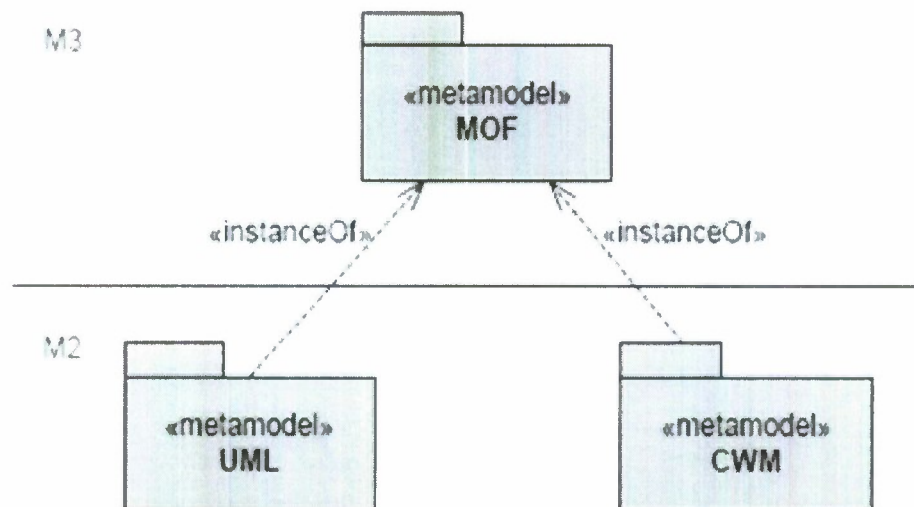
Figure 6.5 illustrates how the MOF is the meta-model used to describe artifacts in the UML. Figure 6.6 shows the OMG meta-model hierarchy, depicting the relationship of model artifacts from the runtime instance to the MOF model at level M3. The runtime instance is denoted at the M0 level and represents a runtime instance of the Class Video described by the user model at the M1 level. At the M1 level, modeled Class Video and modeled instance “:Video” represent instances of the UML artifacts at the M2 level. Notice that the artifact Class is used at both the M2 and M3 levels. Recall that UML and MOF share the core packages—i.e. Class is a part of the Core package. Finally, UML Class and Instance are instances of the MOF artifact Class. It is important to note that each sublevel is an instance of artifacts at the level above.



UML = Unified Modeling Language
MOF = Meta object Facility

CWM = Common Warehouse Model

Figure 6.4. Role of UML Common Core [OMG, 2007a]



UML = Unified Modeling Language
MOF = Meta object Facility

CWM = Common Warehouse Model

Figure 6.5. UML-MOF Meta-Levels [OMG, 2007b]

This relationship facilitates data exchange among modeling tools and transforming models between domains. It especially facilitates transformations between languages that share the same meta-model. The methodology concentrates on modeling system behavior for alternative instances of a specific SOS Architecture and transforming that model into an executable form at the level M1. Then the executable model is created thus instantiating model components at the level M0. The SOS implementations are M1 level instances of the SOSI Architecture described at M2 using UML.

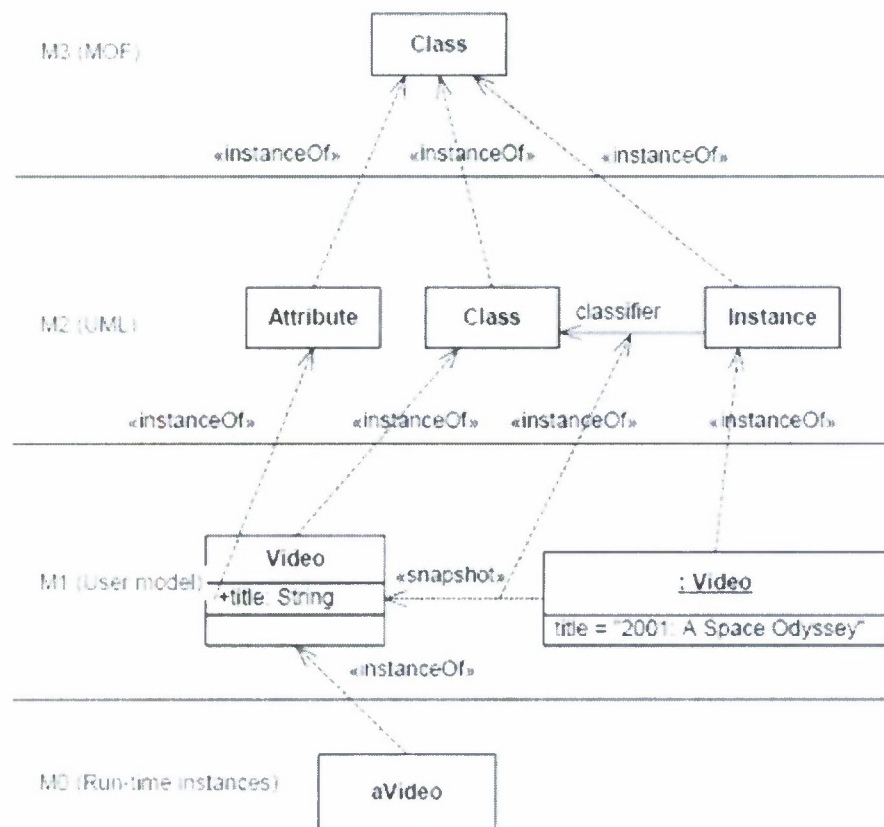


Figure 6.6. Four Layer Meta-Model Hierarchy [OMG, 2007b]

The meta model hierarchy is a key component that enables the model transformation developed for the methodology. The transformation makes extensive use of the M2 layer of the UML meta model hierarchy to accomplish the transformation.

Model Driven Development

Model Driven Development (MDD) is a general term used in the system and software engineering domains to describe a development process that makes extensive use of an abstract representation of the system to make analysis and design decisions. The basic concept is to use graphical models to provide a higher level of abstraction of the system rather than using code or written documentation.

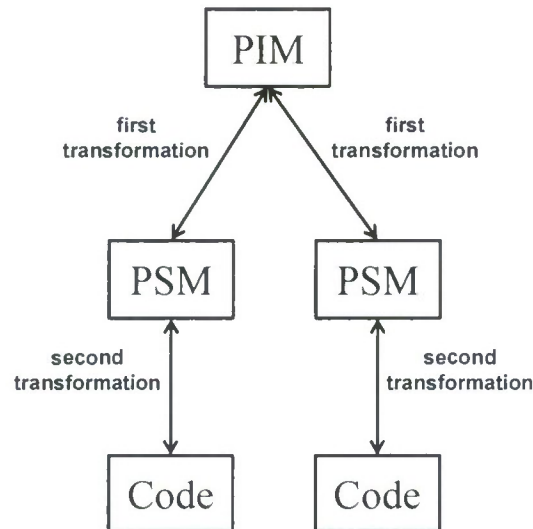
“Models are used to reason about the problem domain and the solution domain. Relationships between these models provide a web of dependencies that record the process by which a solution was created and help us to understand the implications of the changes at any point in the process.” [Beydeda et al., 2005]

The Model Driven Architecture (MDA) is an approach to MDD championed by the Object Management Group. MDA's underlying principles are described in differing levels of detail in Beydeda et al. [2005] and Kleppe et al. [2003]. The MDA principles follow:

1. Models expressed in a well-defined notation are the cornerstone to system understanding for enterprise-scale solutions.
2. Building systems can be organized around a set of models by imposing a series of transformations between models, organized into an architectural framework of layers and transformations.
3. A formal underpinning for describing models in a set of meta-models facilitates meaningful integration and transformation among models and is the basis for automation through tools.
4. Acceptance and broad adoption of this model-based approach requires industry standards to provide openness to consumers, and foster competition among vendors.

Figure 6.7 represents the fundamentals of MDA. One of the primary concepts of MDA is platform independence of the initial model, also called the Platform Independence Model (PIM). The PIM is transformed into a Platform Specific Model (PSM) that adds the specific requirements of the target language or platform. For example, a PIM represented as a class diagram in UML might be transformed into a model form that adds the details for a Java-specific implementation PSM. The final step in the MDA process would be to generate the Java code from the PSM. In MDA, the code generation is viewed as another automated transformation. Note that this process is Model driven, any changes required in the PSM are first implemented in the PIM and then the PSM is regenerated. The idea is that the PIM is the reference, not the code.

This research used a modified Model Driven Architecture approach to provide a Model Driven Development environment for modeling the dynamic behavior of a SOS architecture. This transformation includes creating a PIM using UML and performing two transformations to create an executable representation of the UML. The primary advantage of an MDD environment for the methodology presented is the ability to trace model artifacts that appear in the transformed representation (PSM) to the original representation (PIM). Then changes required in the PSM are made in the PIM and the PSM regenerated. This ensures the model remains consistent with the implementation.



PIM = Platform Independent Model, PSM = Platform Specific Model

Figure 6.7. Model Driven Architecture Fundamental Concept [OMG, 2007b]

Executable Modeling Languages

Carl Petri [1966] first described the nets that bear his name in 1962. While we will not use this initial representation for this research, Petri's paper is the seminal work for this concept and provides the foundation for all Petri Net (PN) work to date.

The "low level" nets described by Petri require the developer to model at a very low level of abstraction. Research was done to raise the level of abstraction of Petri Nets. Genrich and Lautenbach [1979] developed Predicate Transition Nets which extended the range of application of the Petri Net construct. Predicate Transition Nets also maintain a close relationship to lower level nets. Jensen (1991) described High-level Petri Nets which add the ability to describe conditions and actions that cause a transition to fire. This raises the net's level of abstraction by allowing the transition to consider the characteristics of input tokens and model the action represented by the transition.

Jensen [1991] once again extended the state-of-the-art of Petri Nets by adding the concept of color to the tokens, analogous to data types in functional programming. This addition allows the tokens to hold specific characteristics that are passed from place to place through the logic contained in the transitions that represent the actions performed. This allows the modeler to use complex logic at the transition to more completely model the system's behavior. Jensen also offers 12 advantages of CPNs:

1. CPNs have a graphical representation.
2. CPNs integrate the description of control and synchronization with the description of data manipulation.
3. CPNs offer hierarchical descriptions.

4. CPNs offer interactive simulations.
5. CPNs have a number of formal analysis methods by which properties of CP-nets can be proved.
6. CPNs have computer tools supporting their drawing, simulation and formal analysis.
7. CPNs have a well-defined semantics which unambiguously defines the behavior of each CP-net.
8. CPNs are very general and can be used to describe a large variety of different system.
9. CPNs have very few, but powerful, primitives.
10. CPNs have an explicit description of both states and actions.
11. CPNs have a semantics which build upon true concurrency, instead of interleaving.
12. CPNs are stable towards minor changes of the modeled system.

In addition to color, Jensen [1991] also introduced hierarchies in Colored Petri Nets. The hierarchy allows analysts to structure a complex model into a series of related “subpages” that relate to one another using specific constructs that define the subpage’s inputs and outputs and the relationship of the subpage to a transition on a higher-level page. The techniques that provide the ability to decompose the CPN into a set of hierarchical set of subpages do not extend the theoretical underpinnings of the CPN. A hierarchical CPN can be “flattened” to represent a non-hierarchical CPN. In fact, Jensen [1991] showed that the hierarchical nets are equivalent to the flattened CPNs. This research takes advantage of these hierarchies and transforms hierarchical UML Activity Diagrams into hierarchical CPNs.

The grounding of PN and CPN in graph theory allows the use of formal algorithms for the analysis of such graphs. One such analysis is the identification of invariants of the graph. Farkas [1902], Genrich and Lautenbach [1979] and Hillion [1986] offer algorithms for identifying the invariants of a PN graph. This methodology uses the Farkas algorithm and concepts developed by Hillion to analyze the graphs that represent the executable models created by this methodology.

Levis and Wagenhals [2000] offer the basic ingredients of an executable model when using the results of a system architecture development as the input into the executable representation. The architecture must provide the activity model, data model, and rule model to completely address the requirements of the executable. Levis and Wagenhals do not address the semantics of the architecture model, nor that there be a requirement for a model driven development environment. The semantics define the behavior of each model artifact. A model driven environment demands clear description of the effects each model artifact has on the behavior of the system. Semantics are particularly important for constructs that represent dynamic behavior, like State Machine Diagrams, Activity Diagrams and, Interaction Diagrams.

Jensen [1992] described how to translate an activity model based on functional decomposition to a CPN, when the activity model is expressed in IDEF0 (Integration Definition for Function Modeling (IDEF0) [National Institute for Standards and Technology, 1993]). Levis and Wagenhals [2001] and Wagenhals et al. [2000] built on the concept in their process for both

developing a DODAF architecture and analyzing the architecture using a CPN. The CPN provides formal semantics to the IDEF0 so a dynamic model can be created [Jensen, 1992, 1997].

Breton and Bézivin [2001] and Hansen [2001] address executable models from the perspective of the meta-model for PN and CPN respectively. Breton and Bézivin address the PN meta-model in an attempt to provide the tools to do a Meta Object Facility (MOF) style transformation from the UML. They concentrate on the advantages of meta modeling for the purposes of transformation, and use the dynamic aspects, as opposed to the static aspects, to address shortcomings in the semantics for UML. They use PNs as an example to support their argument that the UML needs a clear set of semantics to deal with the models' executability. Hansen [2001] proposes to create a profile for UML so it can represent the semantics of CPN. His approach proposes to extend UML to represent a CPN as another diagram available to the UML modeler to represent behavior: "CPN have, in contrast to UML state machines, a precise semantics and powerful analysis methods. Thus, CP-nets should be separate to state machines and could, possibly, replace state machines in the UML." Both approaches address what must be included in the meta-model to accurately describe the semantics for UML models especially the dynamic portions of those models. The methodology presented uses the attributes that must be included in the UML model to properly represent the SOS characteristics and the associated executable model attributes.

Much of this research makes use of methods from the software engineering community. Selic [1994] describes a modeling technique for a specific subset of the software domain concerning real-time systems. The Real-Time Object Oriented Modeling (ROOM) technique uses an executable model to analyze the behavior of the software design. Together with the ObjecTime™ environment, ROOM provides a construct that allows architects to create executable models at all phases of development: from the analysis phase through design and implementation [Selic, 1994]. The ObjecTime environment was based on the state machine model. While not as general as a CPN, it was very effective for developing real-time systems.

UML to Colored Petri Net (CPN) Transformation

Most of the literature concerning transformation of UML to Petri Net (PN) addresses the state machine and collaboration diagrams. Merseguer [2002], Bernardi et al. [2002], Pooley and King [1999], and Saldhanna and Shatz [2000] all address the transformation of state machines and collaboration/communication diagrams for the purposes of system performance analysis. They address developing a PN but do not fully address the types of analysis that can be conducted with the PN. Eshuis and Wieringa [2001a], López-Grao et al. [2004], Petriu and Shen [2002], and Störrle [2005] address the transformation of activity diagrams to Petri Nets. Once again, the transformation is the primary goal. The analysis and what can be represented in the UML model are left to the reader. This research describes a transformation from UML Activity Diagrams to create an executable model of the SOS.

Software Metrics

Coupling and cohesion are important concepts used in this research. In this section their traditional definitions will be discussed followed by a short overview of each metric and its usefulness in the software engineering community. Both the static and dynamic measurement of cohesion and coupling are addressed. There is also research that compares the validity of dynamic measures to static measures [Briand et al., 1999; Hassoun et al., 2005].

Traditionally, software coupling is defined as “the degree of interdependency between modules” [Yourdon and Constantine, 1979]. The software engineering community agrees that it is good practice to minimize coupling, as lower coupling promotes reusability and maintainability of the class or module [Chidamber and Kemerer, 1994]. Most of the work concerns static measures of coupling in which the code is developed then analyzed for inter-object coupling [Chidamber and Kemerer, 1994; Yourdon and Constantine, 1979]. More recent work has been done addressing dynamic metrics for coupling. These measures involve analyzing the application as it is running to understand the dynamic interaction of the interoperating objects. The direction of coupling between objects is described as Import Coupling and Export Coupling. The concept depends on the object’s perspective: When an object calls a method in another object, this is import coupling. When an object’s method is invoked by another object is referred to as export coupling [Arisholm et al., 2004]. The concept is that the result of the method call either exports data out of the object or imports data from another object. Much of the research is differentiated by when the analysis is accomplished in the development process. There is research where coupling is analyzed late in the design process, Arisholm et al. [2002] analyzes running code. While other research addresses the dynamic coupling metric earlier in the development phase [Yacoub, 1999]. Yacoub’s research contained two measures Import Object Coupling (IOC) and Export Object Coupling (EOC) and was completed in the context of developing a real-time system using the Real-Time Object Oriented Modeling (ROOM) paradigm. The ROOM charts that are analogous to a state transition diagram are used to create a simulation of the application before final application code development. An analysis of the ROOM-derived simulation provided the measures for import and export coupling [Yacoub, 1999].

Cohesion is the practice of keeping related things together. In large part, this is fundamental to object oriented design. Cohesion relates to the idea of “similarity” of methods and attributes in a class. In other words, the cohesiveness of a class is the degree to which a given class encapsulates a set of consistent, semantically related attributes and methods [Chidamber and Kemerer, 1994]. Chidamber and Kemerer [1994] approach the relatedness of attributes and methods of a class obliquely with their lack of cohesion measure (LCOM). A class with a high LCOM may not have a focused objective and may be trying to achieve unrelated objectives. The behavior of such a class may be harder to predict than a class with a lower LCOM. Additionally, lack of cohesion also increases the complexity of a class, thus increasing the likelihood of errors during the development process. The lack of cohesion could also reveal positive attributes for

the purposes of SOS development. A component that lacks cohesion because it lacks a particular focus may be more flexible because it can be used in more situations without reconfiguring the system. This aspect of cohesion will be explored in Section 6.3.

Summary

The methodology describes a method for assessing the ability of a SOS architecture to adapt to unpredicted operating environments. To that end, this research uses methods from both the systems engineering and software engineering communities to create a dynamic model that represents the interacting systems of the SOS. This chapter provided an overview of the technology that provided the foundation for the methodology presented.

The concept of the SOS is described and the weaknesses of the current definitions are discussed. The literature addresses the managerial aspects of the SOS but fails to address the technical aspects especially as they apply to the requirements of the organization. The discussion also point out that without clear boundary it is very difficult to create alternative architectures for comparison.

In order to create the dynamic representation of the architecture, the rule model, data model and behavior model must be represented in the architecture. The DODAF was presented as a common framework for identifying and presenting the data required for the executable model. Additionally, the UML is used to represent the required architecture information.

The methodology requires the executable model so the UML must be transformed in to the executable form. A CPN is the chosen executable form of the methodology. The transformation of UML into CPN is well understood; however the analysis of multiple SOS processes is not addressed fully. The methodology creates an executable model for analysis of the simultaneously executing processes that a SOS must support in order to provide the set of capabilities desired by the organization. Finally, neither discipline addresses architecture-wide measures that enable the comparison of SOS performance in the initial analysis and design phases of development. The following chapters address the transformation of multiple process descriptions into an executable form and describe assessment measures that identify characteristics important to an agile organization.

6.3 Assessment Measures

This section provides a technical definition of a SOS and specific characteristics that relate to the SOS's ability to change its structural configuration to adapt to a new operational environment. The first section establishes the conceptual relationship between the components of the SOS definition. The second section defines a SOS and several properties. The third section describes assessment measures and the formulas used to calculate them. The last section is an example illustrating how the assessment measures are calculated.

System of Systems Definition

The relationship between SOS components is significant and warrants discussion and clarification in the context of this methodology. Thus, in order to properly define the SOS, the relationships between its various components must be established. An extended definition of SOS specifies the resources available to the enterprise and differentiates the specific implementation used for a particular purpose. The resources available to the enterprise compose the SOS. The resources are used by the enterprise to realize specific capabilities required by the organization. Identifying the specific implementation provides a structure on which to make measurements. It also provides a way to identify alternatives for comparison. A specific implementation requires a set of resources that are configured to provide a specific set of capabilities to the organization. The specific implementation is developed using the specification of the structural and behavioral relationships between resources defined in an architecture. The specific instances of the architecture can be assessed for their ability to address the needs of the organization.

Figure 6.8 is a concept map of the component relationships of the SOS taxonomy. Concept maps “are graphical tools for organizing and representing knowledge” [Novak and Cañas, 2006]. Concept maps are usually organized with the most abstract component at the top of the diagram and the more specific concepts arranged below. This hierarchical structure of the concept map is effective for illustrating the taxonomic relationships of the components of the SOS. The concept map provides a succinct graphical method for communicating the relationship of SOS concepts.

We define a particular resource available to the enterprise as an Element. An Element can represent any level of abstraction. Examples used here represent the Element as an information system. The specific implementation of an architecture for a particular purpose is the SOS Instance (SOSI). The SOSI consists of a subset of the Elements available to the enterprise. A Node provides structure to the SOSI since each Element is assigned to one and only one Node. Nodes possess a communication structure that ensures the internal communication between Elements assigned to the Node and an external Link that represents the presence of a communication facility between Nodes. The behavior of the SOSI is described by a SOSI Capability (SOSIC). A SOSIC represents a process that uses a subset of SOSI Elements to realize a capability required by the organization. Elements of a SOSI may be members of more than one SOSIC. Messages represent the data passed between Elements in the execution of the SOSIC. Finally, the SOSI Architecture is a description of the relationship between the Elements, Nodes, SOSICs, Links and Messages; therefore the SOSI is a particular instance of a SOSI Architecture that uses a subset of the Elements, allocated to Nodes, to model specific behavior represented by a SOSIC.

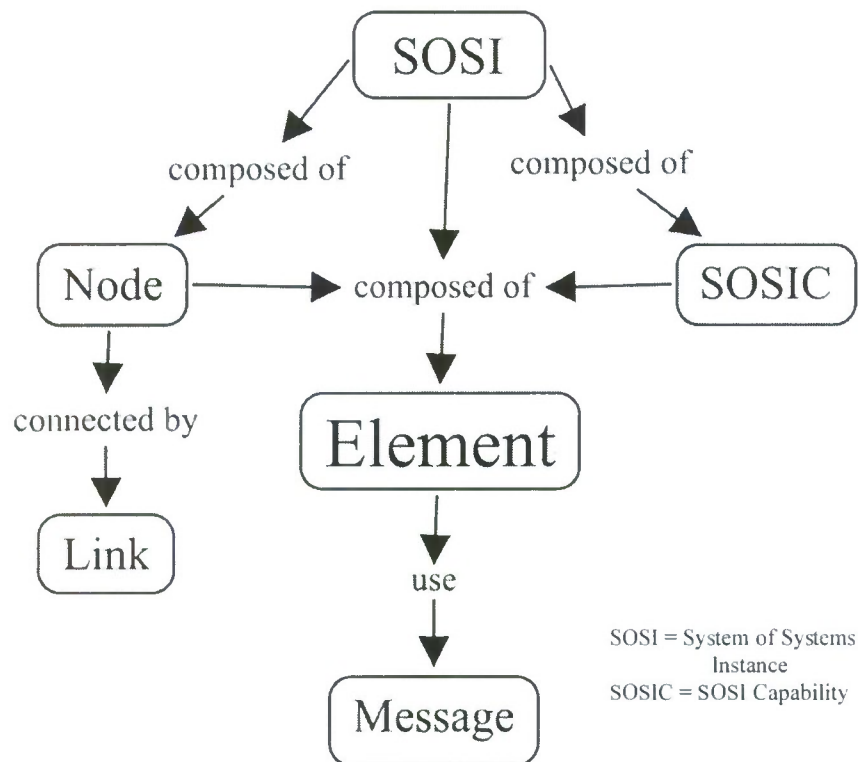


Figure 6.8. SOSI Taxonomy

The next section discusses the properties that differentiate the SOSI from the SOSI.

SOS Properties

1. The SOS defines a predetermined set of capabilities.
2. The set of Elements that compose the SOS changes over time.
3. The SOS Elements are heterogeneous.
4. The SOS Elements are at different program maturity levels.

SOS Property 1

Each SOS defines a set of predetermined capabilities—the planned capabilities. The predetermined capabilities are the ones that are specifically provided by a set of Elements that are members of the SOS. The planned capabilities are described in operational and systems architectures available to the SOS engineer.

SOS Property 2

The SOS Elements change over time. The individual Elements enter and leave the SOS as required. As the organization's focus changes, Elements will be retired and new Elements will be introduced to enhance existing capability or realize a newly defined capability. This property addresses the nature of organizations as they evolve. The available Elements must change to meet the needs of the organization.

SOS Property 3

The various Elements are heterogeneous. From an information technology perspective, they execute on different platforms and are developed using various technologies and languages. Also, some systems are primarily hardware and others are primarily software. Elements can also represent the behavior of humans in the SOS.

SOS Property 4

The SOS Elements are at different program maturity levels. The SOS can contain both experimental Elements and Elements that have existed for some time. The abstraction level is high, so systems and capabilities can be tested at the analysis and early design phases of development rather than waiting for lab tests of development software and hardware.

Definition 6.1 is the formal description of the SOS. The SOS Ω is a triple $\{E, F, M\}$ where E is the set of Elements that compose the SOS, F is the set of SOSI defined from elements of E in the SOS, and M is the set of messages used by the Elements and SOSI of the SOS. Each member of the set F is a disjoint subset of E .

Definition 6.1: $\Omega = \{E, F, M\}$

where,

$E = \{e_1, e_2, e_3, \dots, e_v\}$; set of Elements that compose SOS

$F = \{f_1, f_2, f_3, \dots, f_k\}$; set of SOSI developed from E

$M = \{m_1, m_2, m_3, \dots, m_h\}$; set of messages used by SOS

SOSI Properties

1. A SOSI is created/instantiated from available Elements of the SOS based on the relationships described in the SOSI Architecture.
2. A SOSI provides a particular subset of the planned capabilities.
3. Each SOSI is unique.

SOSI Property 1

A SOSI is instantiated from existing Elements in the SOS following the relationships described in the SOSI Architecture. It is very important to understand the relationship of Elements, Nodes, and SOSICs, so careful decisions must be made as to how the SOSI will be configured. Additionally, as new Elements are developed to add capability to the SOS, we must effectively evaluate the effects of the new Elements on the SOSI Architecture and the SOSI. New Elements can be added to the set of SOS Elements; they are not necessarily added to the SOSI. Finally, Elements leave the SOS as they are retired, damaged, or the mission changes.

When an Element leaves, the SOSI Architecture and any SOSI using that Element must be reevaluated. Figure 6.9 illustrates the changing nature of the SOS. The cloud represents the SOS's dynamic nature while the SOSI objects show that, at least for specific periods of time, the SOSI is constant and can be observed. A challenge in previous SOS analysis was the attempt to analyze the SOS as whole. Thus, the cloud also represents the difficulty in defining a boundary around the SOS. Without a clear boundary, how do you analyze the SOS's characteristics? The methodology presented addresses the individual SOSI created from Elements available in the SOS. The individual SOSI have boundaries which makes assessment and comparison of SOSI alternatives possible.

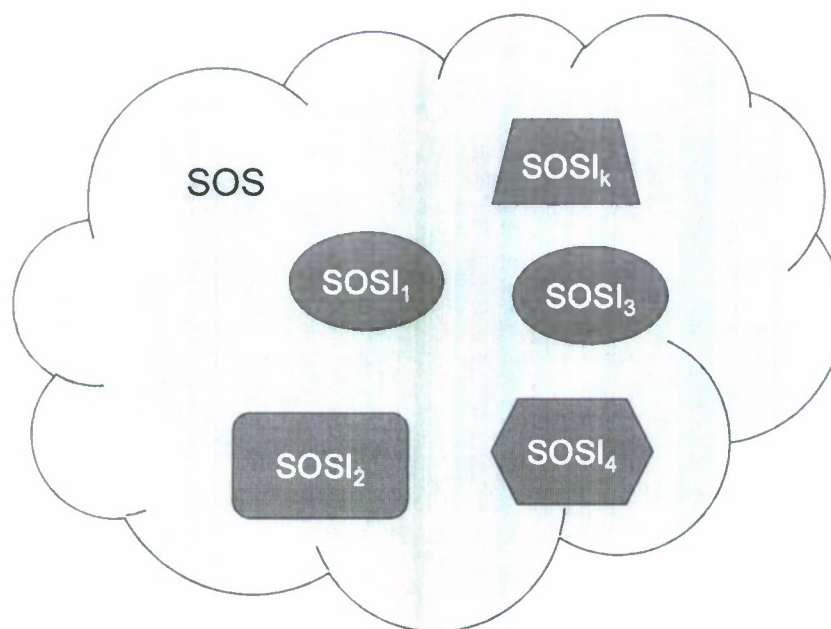


Figure 6.9. System of Systems (SOS) Boundary

SOSI Property 2

The SOSI is instantiated to provide a specific set of capabilities that have proven challenging to acquire with a single system. It is difficult to decide what Elements will comprise the SOSI and test the attributes of their interaction in the SOSI. The decisions on what Elements and how they are configured require rigorous analysis that is not provided before the Elements are brought together for integration. This can produce tightly coupled configurations that are difficult to modify as the situation for their use changes.

SOSI Property 3

Each SOSI is unique. The SOSI Architecture describes the relationships between Nodes, Elements and SOSICs. A particular operating environment and mission governs the way the

SOSI will be instantiated. SOSIs can differ in the number of Elements, how the Elements are distributed on the Nodes, and how the Nodes are linked together. They can also differ in the way Elements are used to provide the specific SOSIC. This methodology specifically addresses structural changes.

SOSI f_k is a 5-tuple $\{E_k, N_k, P_k, M_k, C_k\}$ formally described by definition 6.2. The components of the SOSI f_k are the sets E_k, N_k, P_k, M_k , and C_k that represent the Elements, Nodes, SOSICs, Messages and Links of f_k , respectively. The set of Elements E_k represents the subset of E from the SOS that composes the SOSI f_k . The set of the Nodes N_k represents the Nodes that provide the structure for SOSI f_k . The set of SOSIC P_k represents the SOSICs that describe the capabilities present in the SOSI f_k . The set of Messages M_k represents the Messages used by SOSI f_k . The set of Links C_k represents the communication facility between Nodes.

Definition 6.2: $f_k = \{E_k, N_k, P_k, M_k, C_k\}$

where,

$E_k = \{e_{k1}, e_{k2}, e_{k3}, \dots, e_{ks}\}$; set of Elements in f_k

$N_k = \{n_{k1}, n_{k2}, n_{k3}, \dots, n_{kw}\}$; set of Nodes in f_k

$P_k = \{p_{k1}, p_{k2}, p_{k3}, \dots, p_{kr}\}$; set of SOSIC in f_k

$M_k = \{m_{k1}, m_{k2}, m_{k3}, \dots, m_{kq}\}$; set of Messages in f_k

$C_k = \{c_{k1}, c_{k2}, c_{k3}, \dots, c_{kb}\}$; set of Links in f_k

Equations 6.1 and 6.2 describe the relationship of the Node to the SOSI. Equation 6.1 shows that an Element of SOSI f_k shall be a member of one and only one Node. This is expressed formally stating that the Nodes in f_k are disjoint. Equation 6.2 shows that the union of all the Nodes in the SOSI f_k accounts for all the Elements in SOSI f_k . The elements of set N_k are a partition on the set of Elements E_k . Equation 6.3 shows that Elements may be members of more than SOSIC. Equation 6.4 shows that all SOSIC P_k that compose the SOSI f_k shall account for only elements of E_k . The set P_k is a cover of the Elements of E_k . Figure 6.10 includes a diagram showing the relationships described in Equations 6.1, 6.2, 6.3, and 6.4. The box on the left shows that the Nodes $n_{1,1}, n_{1,2}$ and $n_{1,3}$ contain all the elements of E_k and that each Element is a member of one and only one Node. The box on the right shows that the Elements of the SOSI f_k can be members of more than one SOSIC and that it is possible for an Element to stand alone.

$$\bigcap_{j=1}^m n_{kj} = \emptyset \quad (6.1)$$

$$\bigcup_{j=1}^m n_{kj} = E_k \quad (6.2)$$

where,

m = the number of Nodes in SOSI f_k

$$\bigcap_{j=1}^m p_{kj} \neq \emptyset \quad (6.3)$$

$$\bigcup_{j=1}^n p_{kj} \leq E_k \quad (6.4)$$

where

n = the number of SOSICs in SOSI f_k

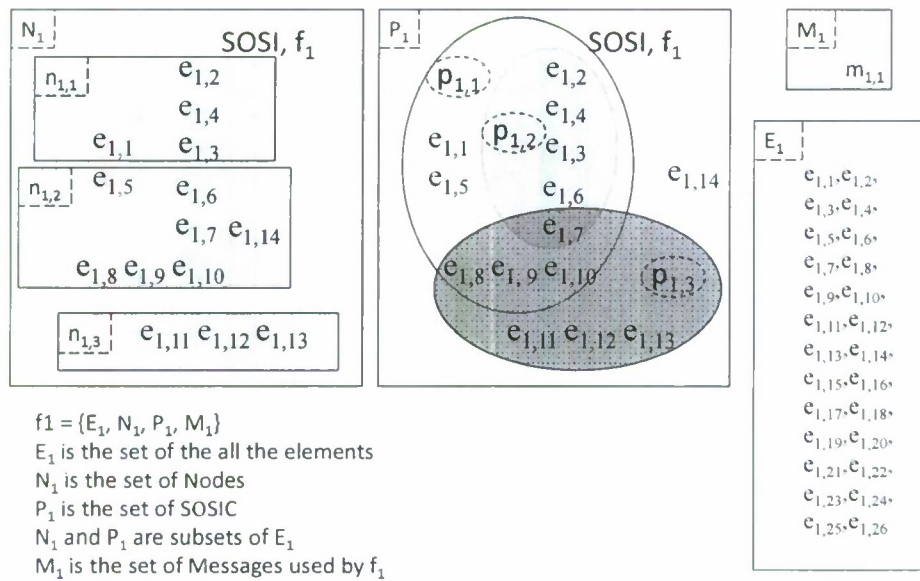


Figure 6.10. SOS Taxonomy Relationships Venn Diagram

SOSI Measures

The proposed measures that will enable the assessment of SOSI alternatives describe the degree of interaction of the Elements within and among the Nodes of the SOSI and the degree of

reuse of the Elements among the SOSICs. Five measures are defined: Cohesion, Coupling, Degree of Reuse, Adaptability and Agility.

Cohesion

Cohesion is a measure of “how tightly bound or related internal elements [of a software engineering module] are to one another.” [Yourdan and Constantine, 1979] Bieman [1998] defines cohesion as a measure of the relatedness of inputs to outputs of a Module. In the SOSI, a Node with low cohesion is easier to reconfigure than a Node with high cohesion. In the software engineering domain, a class with low cohesion can and should be split into more cohesive classes. This aids in the maintenance of the software because cohesive classes do specific tasks effectively encapsulating functionality and aiding maintenance. This methodology uses the idea to show that adaptable Nodes should show low Cohesion. That means they can be reconfigured without significantly effecting the execution of the SOSI. The level of cohesion can be measured in various ways to provide insight into how strongly Node inputs are related to Node outputs.

The computation for cohesion is adapted from Bieman’s work [1998] on Design Level Cohesion measures. Fundamentally, Bieman’s metric is a measure of the relationship between inputs and outputs of a module. Most software measures of cohesion analyze the application code to compute the cohesion measure. A contribution of this research is the idea that the cohesion measure can be computed by analyzing the paths connecting inputs and outputs modeled by a graph the represents the interaction of Elements on a Node. The cohesion measure is a significant departure from Bieman, but capitalizes on the relationship between inputs and outputs. A Node with high cohesion will have a high proportion of inputs related to the outputs. A Node with low cohesion will have a low proportion of inputs that are related to the outputs.

Cohesion is measured by calculating the number of paths that can be traced through the Elements from the Node inputs to Node outputs. The more paths that connect inputs to outputs the higher the cohesion of the node. Equation 6.5 shows the calculation of Node Cohesion. SOSI Cohesion, Equation 6.6, is the average Node Cohesion. The following section is a discussion on computing the number of paths.

$$\text{Coh}(n_{ki}) = \frac{Z_{ki}}{X_{ki}} \quad (6.5)$$

where,

Z_{ki} = number of paths in Node n_{ki}

$X_{ki} = I_{ki} * Q_{ki}$

where,

I_{ki} = number of inputs for the Node

Q_{ki} = number of outputs for the Node

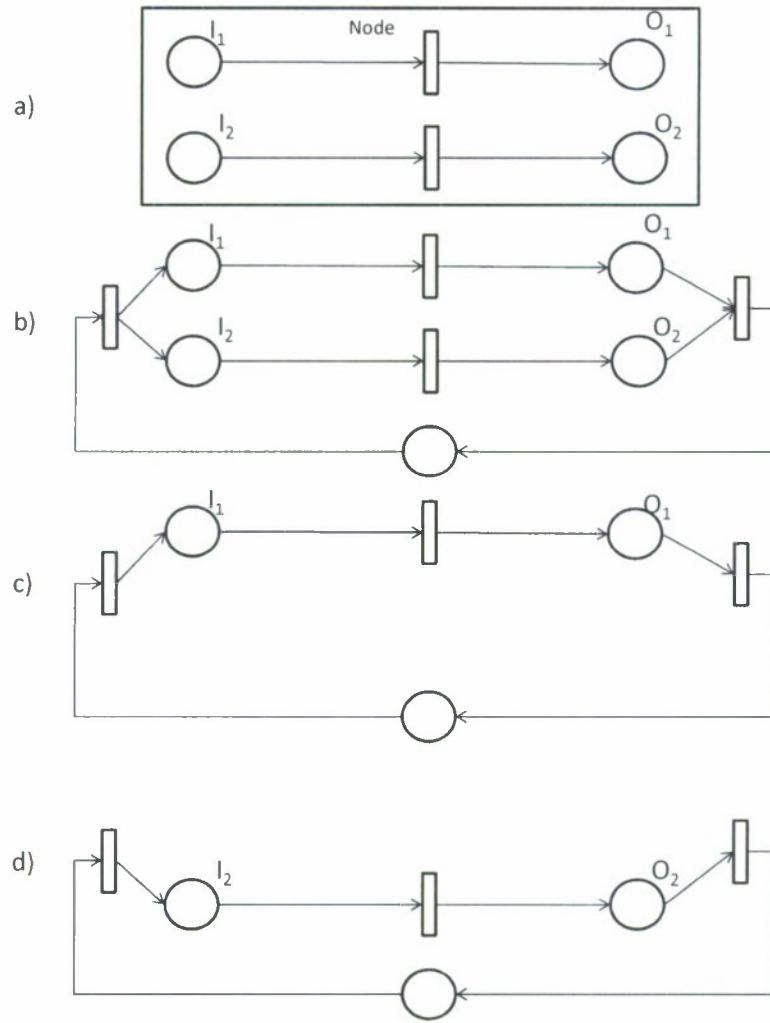
$$\text{Coh}(f_k) = \frac{\sum_{i=1}^m \text{Coh}(n_{ki})}{m} \quad (6.6)$$

where,

m = number of Nodes in SOSI

The paths that connect the inputs to the outputs can be computed formally using algorithms that rely on formally defined graphs that represent the Nodes and the Elements that are assigned to them. There are many different graphs styles that are formally defined. For the purposes of this methodology, a Colored Petri Net (CPN) is used to represent the Node and the Elements. The transitions of the CPN represent the Elements of the SOSI assigned to the Node. The paths that connect inputs to outputs are computed by calculating the S-invariants of the Petri-net (PN) that represents a Node. For the purposes of this methodology, the S-invariants of the graph identify the paths that connect the Node inputs to the Node outputs. The “Farkas Algorithm” [Farkas, 1902] is used to compute the S-invariants of the Node. The methodology described here creates a CPN for each Node. Figure 6.11 illustrates the modification of the CPN that represents the Node and the calculation of the paths. Figure 6.11a is converted to a PN by removing the hierarchical constructs and colors from the net. The remainder is a PN that represents the structure of the inter-connections between the inputs and outputs of the Node. Next, the PN must be modified to include a common input transition that connects to all input places and an output transition that all output places are connected to. This technique was developed by Hillion [1986]. The output and input transition are connected with a connecting place. Figure 6.11b shows the addition of the connecting place and transitions. Then, the resulting S-invariants are computed on the modified CPN. All S-invariants that include the common connecting place are paths that connect Node inputs to Node outputs. Figures 6.11 c and d show the paths for this net.

Figure 6.12 shows two CPNs that represent SOSI Nodes. Node1 is an example of low cohesion. Node2 is an example of higher cohesion. Both Nodes have three inputs and three outputs. In Node1 the inputs are associated with only one output, so the cohesion is lower than that of Node2 where all the inputs affect all the outputs.



9

Figure 6.11: Simple Information Flow Path

Figure 6.12 shows a more complicated node with loops in the CPN representation. In the case of Figure 6.13, Node3 has nine direct paths that touch the four loops created by places P1, P2, P3, and P4. This makes the total number of paths 36 because each of the nine direct paths now has four more paths that it could follow through the loops.

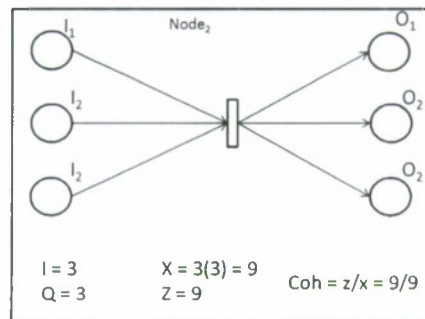
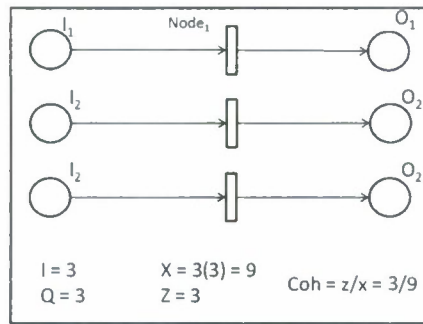


Figure 6.12. Cohesion Example

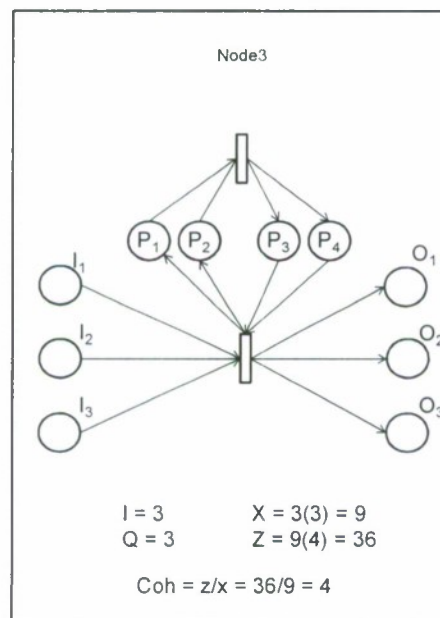


Figure 6.13. Node with Loops

Coupling

Coupling is a measure of the interdependence between Nodes of the SOSI. Node coupling is the degree that one Node is dependent on another Node in the SOSI. Node coupling will be measured dynamically in this methodology, which helps the developer understand the extent of the relationship between Nodes within the SOSI. As mentioned in Chapter 2, coupling can be measured as import coupling or export coupling. This methodology measures export coupling among the Nodes of the SOSI.

Because the coupling measure is dynamic, there is a requirement for a Scenario that sets the parameters of the executable model and provides the initial conditions. Definition 6.3 is the formal definition of the Scenario for the calculation of the Coupling measure. In order to get consistent execution results from multiple architectures, a common scenario must be defined. The scenario X is defined by setting the initial conditions I_X and the parameters R_X for the SOSI under consideration.

Definition 6.3 $\text{Scenario}_x = \{I_x, R_x, f_k\}$
 I_x = the initial condition
 R_x = parameters
 f_k = the particular SOSI

Equation 6.7 shows the formula for Node Coupling. This indicates the degree of dependence of a Node to all the other Nodes.

$$\text{Coup}_x(n_{ki}) = \frac{\sum_{i \neq j} |M_x(n_{ki}, n_{kj})|}{C} \quad (6.7)$$

where,

x = scenario generated for the analysis

M_x = messages generated by Scenario x

$C = \left(\frac{m(m-1)}{2} \right)$ = number of possible connections

between Nodes in SOSI f_k

m = number of Nodes in SOSI

Nodes that are highly coupled reduce the ability of the SOSI to change because changes in the highly coupled Nodes propagate changes to other Nodes. The propagation of change reduces the ability of the organization to adapt because highly coupled systems are difficult change due

to the complexity of the interface between Nodes. This is similar to the notion of coupling in software engineering. The more interconnected two classes are the more difficult it is to make changes in one without reflective changes in the other. The coupling measure reveals the level of dependence among the Nodes of the SOSI.

The SOSI Coupling measure is the average of the Node Coupling measures. This measure is shown in equation 6.8.

$$\text{Coup}_x(f_k) = \frac{\sum_{i=1}^w \text{Coup}_x(n_{ki})}{m} \quad (6.8)$$

Adaptability Computation

Adaptability is a function of the product of Cohesion and Coupling. Equation 6.9 is the formula for Adaptability. D is the Cobb-Douglas [1928] production function. Cobb-Douglas provides an effective method for relating Coupling and Cohesion. Cobb-Douglas is explored later in the section. Equation 6.9 is the formula for calculating Node Adaptability. Equation 6.9 uses Node Coupling and Node Cohesion and the inputs for the calculation. Equation 6.10 is the SOSI level measure of Adaptability. It uses the SOSI level measure of Cohesion and Coupling from Equations 6.6 and 6.8, respectively.

$$\text{Adapt}_x(n_{ki}) = \frac{1}{D} \quad (6.9)$$

where,

$$D = \text{Coh}(n_{ki})^\alpha \text{Coup}(n_{ki})^\beta$$

α = elasticity constant for Cohesion

β = elasticity constant for Coupling

$$\alpha + \beta = 1$$

$$\text{Adapt}_x = \frac{1}{\text{Coh}(f_k)^\alpha \text{Coup}_x(f_k)^\beta} \quad (6.10)$$

Two assumptions must be made in relation to Equation 6.9 and 6.10. These are modified from Cobb and Douglas [1928] to reflect the context of coupling and cohesion.

1. The Adaptability score is proportional to the Adaptability of an actual system when measuring only Coupling and Cohesion.
2. Other factors are accounted for with the scaling factor B. We will make use of the scaling factor to calculate Agility.

The Cobb-Douglas form is inverted to make lower Cobb-Douglas products result in higher Adaptability. Figure 6.14 shows Cobb-Douglas in its traditional form with different values for α and β . Notice how the value of production changes with the value of the elasticity constants. The elasticity constant can be used to adjust the function for the particular characteristics valued by the organization.

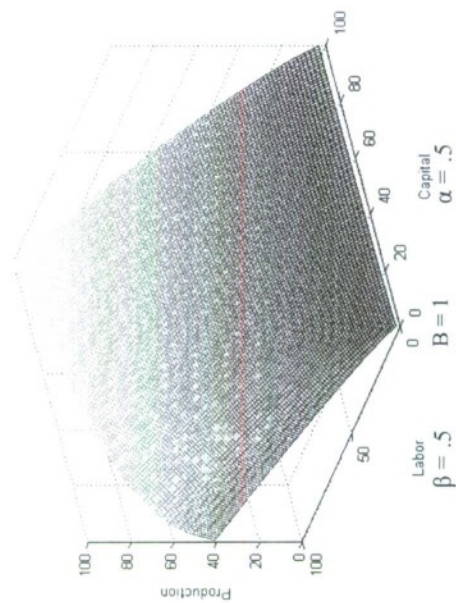
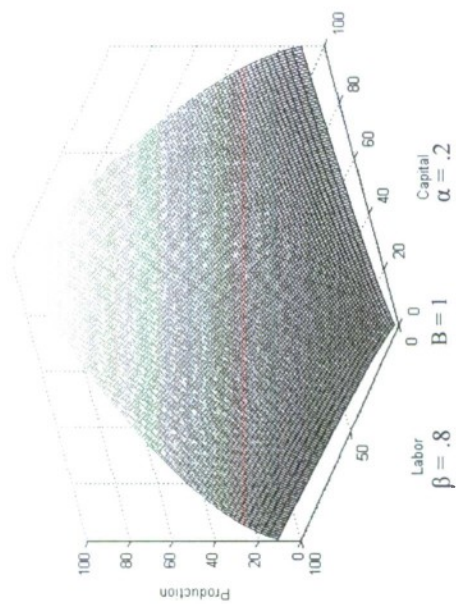
Cobb and Douglas made similar assumptions to account for factors not measured by labor and capital. The scale factor is used to account for other factors that are not measured by Coupling and Cohesion.

Figure 6.15 illustrates the relationship of Cohesion to Coupling and how that relationship can be used to identify a SOSI's level of Adaptability. The figure shows that a SOSI that has low Cohesion and low Coupling will have a higher level of adaptability than a SOSI that has high Cohesion and low Coupling. Cohesion is the primary driver for the Adaptability measure indicated by an associated elasticity value of 0.8. If Coupling remains constant, Adaptability will increase as Cohesion increases. However, if Cohesion is constant and Coupling increases, the Adaptability score increases but at a much lower rate than when Cohesion increases. The elasticity constants, α and β define the relationship between changes in Cohesion and Coupling.

Figure 6.16 is a contour plot of Fig. 6.15. These diagrams show the relationship of Coupling and Cohesion in the computation of the Adaptability. Adaptability will be low when both Coupling and Cohesion are low. Figure 6.16 shows an example contour diagram of the Adaptability plot in Figure 6.15. Notice how the score decreases with the increase in cohesion as opposed to an increase in coupling.

Figure 6.17 illustrates how the contour plot is calculated. The contour diagram (bottom) has a contour line indicating the Adaptability value (z-axis of Figure 6.15) for the values 1 through 10. The top diagram shows planes intersecting the surface thus illustrating how the contour lines are created. The planes are depicted at the odd numbered values of the Adaptability function for illustration purposes.

Figure 6.18 shows three Adaptability plots with different α and β values to show how the shape changes with changes in the elasticity constants. The middle plot with α and β equal to 0.8 and 0.2 respectively, is the shape used to make the calculations for all the examples and case study in this research. The top left plot shows the shape when α and β are reversed. The top right plot shows α and β equal to 0.5.



$$P = BC^{\alpha}L^{\beta}$$

where;

P = Production

C = Capital

L = Labor

B = Scale factor

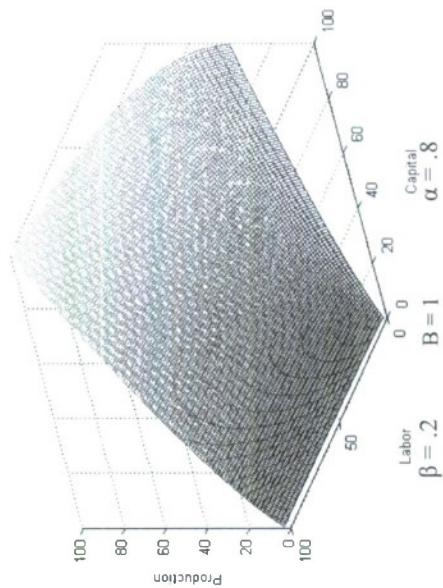


Figure 6.14. Cobb-Douglas Production Function

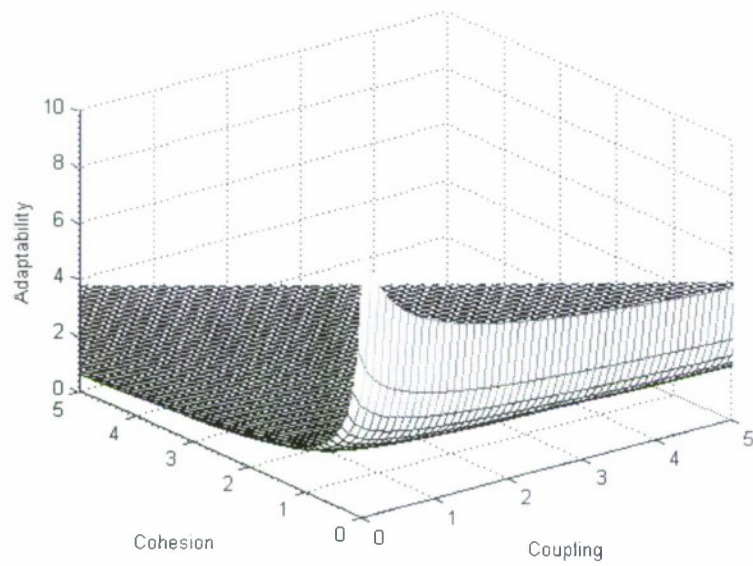


Figure 6.15. Adaptability Plot

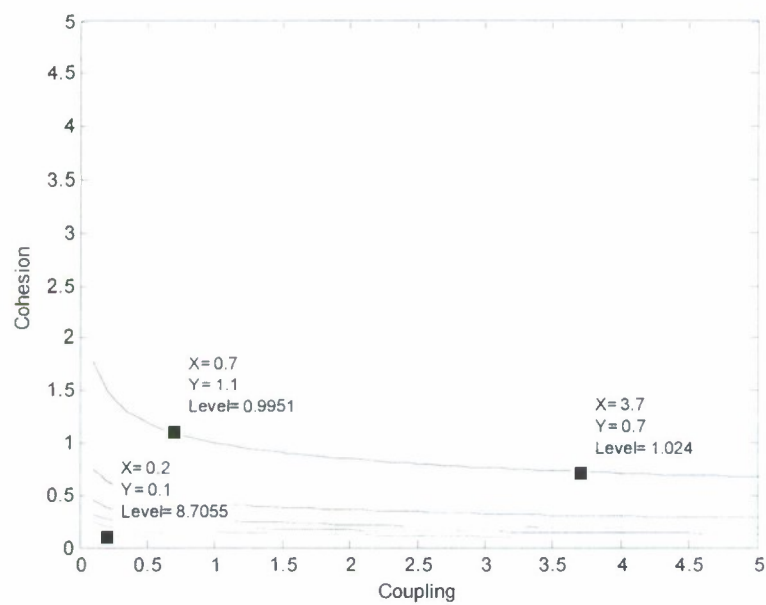


Figure 6.16. Adaptability Contour

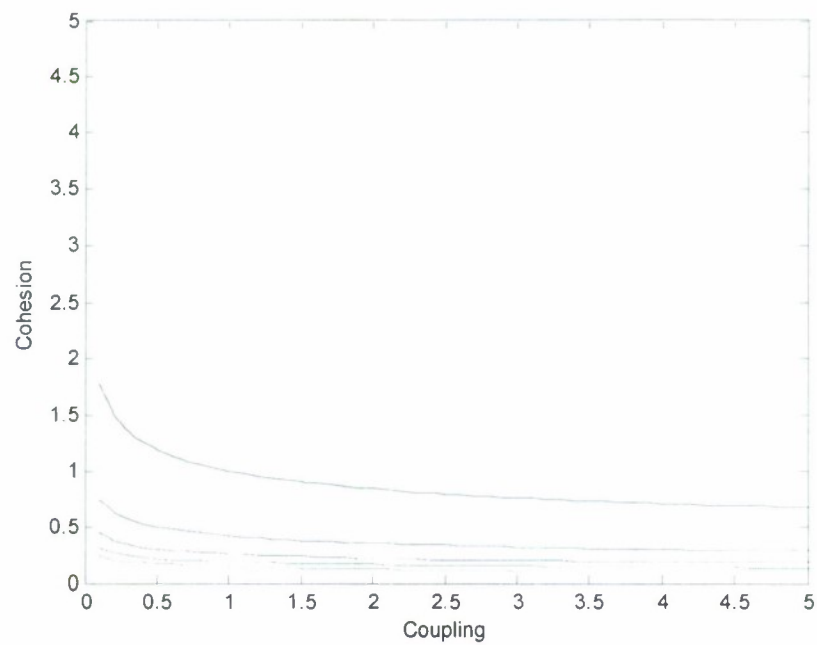
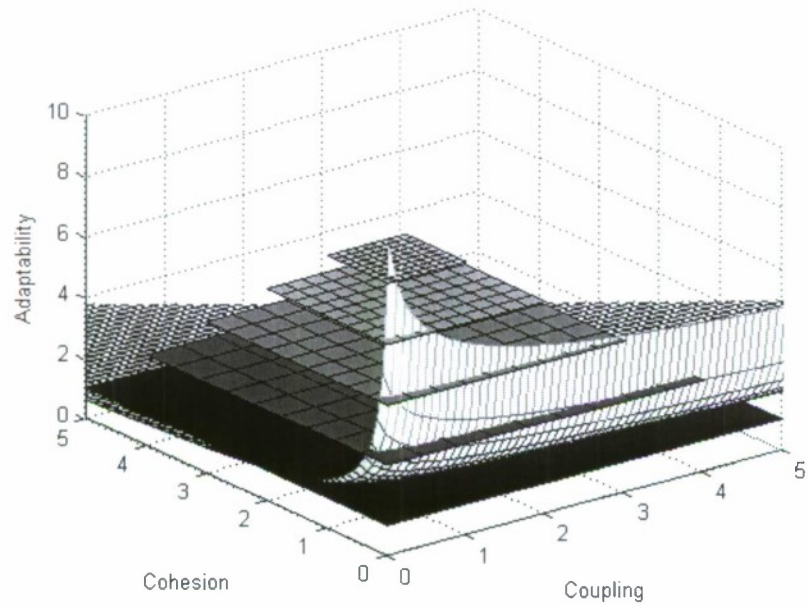


Figure 6.17. Contour Calculation

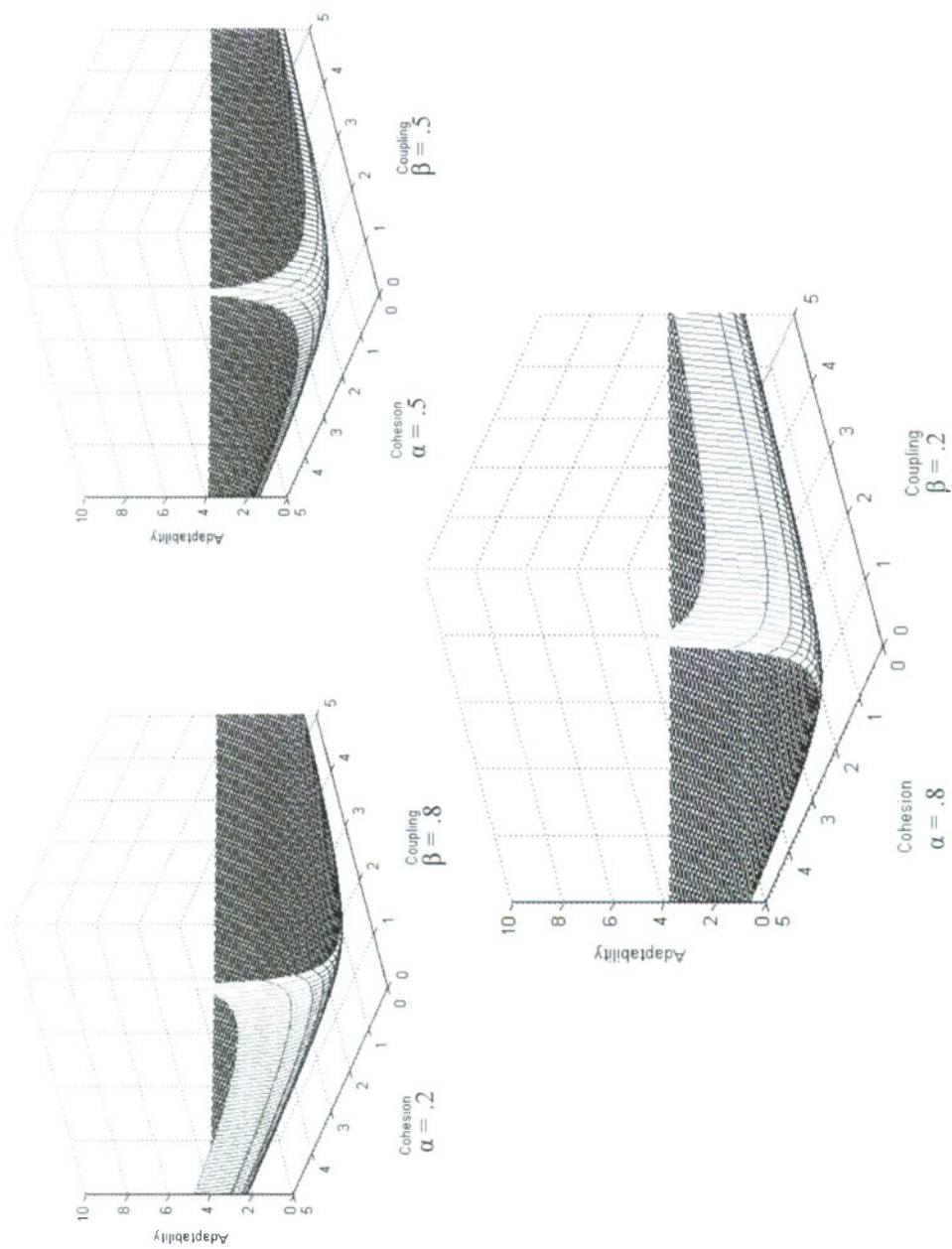


Figure 6.18. Effects of Elasticity Constant on Adaptability

Therefore, given a set of Elements assigned to Nodes, an Architecture that describes the relationships among Nodes and their Elements, and a set of capabilities enabled by the architecture, we distinguish four cases of SOSI Adaptability:

Low Cohesion and Low Coupling = High Adaptability

Low Cohesion and High Coupling = Medium Adaptability

High Cohesion and Low Coupling = Medium Adaptability

High Cohesion and High Coupling = Low Adaptability

Figure 6.19 shows the four cases graphically. A SOSI with high adaptability has relatively low Cohesion and Coupling. This also applies in the limiting cases - zero Coupling and zero Cohesion results in infinite Adaptability while Coupling and Cohesion at extremely high levels cause Adaptability to approach zero (very few if any changes are possible). The first extreme describes a SOSI that has zero interaction between Elements within a Node and zero interaction among Nodes so it can be deployed in any configuration. This is a set of totally uncoupled Nodes with each Node containing a single element – this is not a system of systems. The second extreme of a highly cohesive and coupled SOSI might only be capable of being deployed in one configuration.

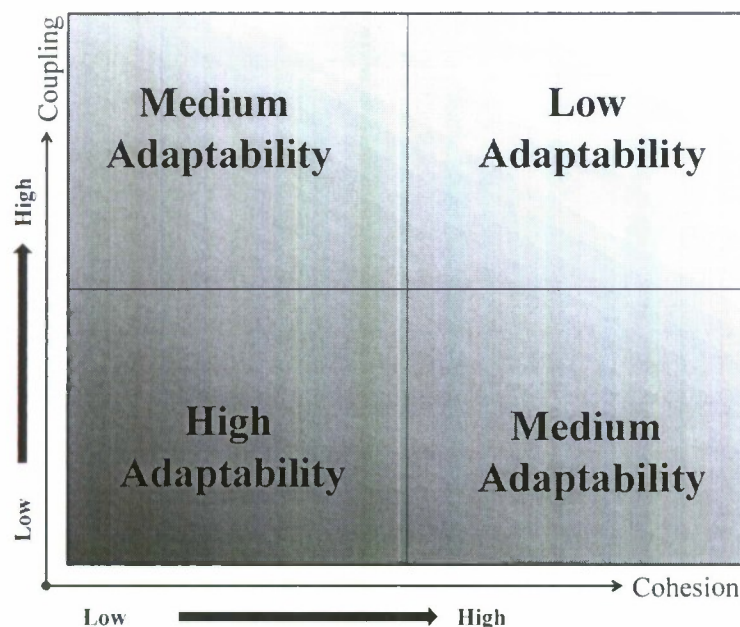


Figure 6.19. Relationship of Cohesion and Coupling to Adaptability

Degree of Reuse and Exclusiveness

The Degree of Reuse measures the extent of reuse of the Elements among the SOSICs of the SOSI. It reflects the ability of the SOSI to execute multiple SOSICs concurrently. It is calculated by counting the number of SOSICs that an Element supports. The Degree of Reuse

for the SOSI is the average Degree of Reuse computed for each Element. The higher the Degree of Reuse the lower the ability of the organization to execute the SOSICs concurrently. In order to create an index with values between 0 and 1 where 1 indicates that each Element in the SOSI supports one and only one SOSIC, Exclusiveness is calculated as the inverse of the Degree of Reuse. The Elements with highest reuse are identified as “highly reused Elements.” High reuse affects the ability of the SOSI to execute SOSICs concurrently because of the potential contentions for resources (use of Element).

Equation 6.11 is a generic function that returns 1 if x is an element of A . The member function is used to calculate the number of SOSICs that use a particular Element. Equation 6.12 computes the Degree of Reuse (DoR) for a particular SOSIC p_{ki} . It is the average Degree of Reuse of the Elements in the SOSIC.

$$\text{member}_A(x) = \begin{cases} 1, & x \in A \\ 0, & x \notin A \end{cases} \quad (6.11)$$

where,

A = a set

x = a possible element of A

$$\text{DoR}(p_{ki}) = \frac{\sum_{e \in p_{ki}} \sum_{i=1}^r \text{member}_{p_{ki}}(e)}{r} \quad (6.12)$$

where,

r = number of Elements in SOSIC p_{ki}

Equation 6.13 shows the SOSI computation for Degree of Reuse. It measures the overall Degree of Reuse for the SOSI. Equation 6.14 identifies the Elements that are members of the most SOSICs. This helps identify “highly reused” Elements of the SOSI for the assessment methodology.

$$\text{DoR}(f_k) = \frac{\sum_{e \in E_k} \sum_{p_{ki} \in P_k} \text{member}_{p_{ki}}(e)}{s} \quad (6.13)$$

where,

s = the number of Elements in SOSI f_k

$$E \max(f_k) = \sum_{c \in E_k} \max \left(\sum_{p_{ki} \in P_k} \text{member}_{p_{ki}}(c) \right) \quad (6.14)$$

Finally, Exclusiveness is computed as the inverse of degree of reuse. It is calculated for the SOSI as a whole. Equation 6.15 shows the formula to Exclusiveness.

$$\text{Exclusiveness}(f_k) = \frac{1}{\text{DoR}(f_k)} \quad (6.15)$$

Agility

Agility is Adaptability times Exclusiveness. Exclusiveness represents the scale factor defined by Cobb-Douglas. Agility is the degree that a SOSI can adapt to different configurations and execute SOSIC simultaneously, reflecting the notion that the Agility of the SOSI will be reduced if the Exclusiveness measure is low. Equation 6.16 shows the agility computation. Agility is a SOSI level computation because the measure is revealing the aggregate ability of the SOSI to provide capability concurrently and adapt its configuration to different operating environments. Figure 6.20 shows an Adaptability plot from Figure 6.15 that has been scaled by Exclusiveness = 0.5.

$$\text{Agility}_x(f_k) = \text{Exclusiveness}(f_k) \text{Adapt}(f_k) \quad (6.16)$$

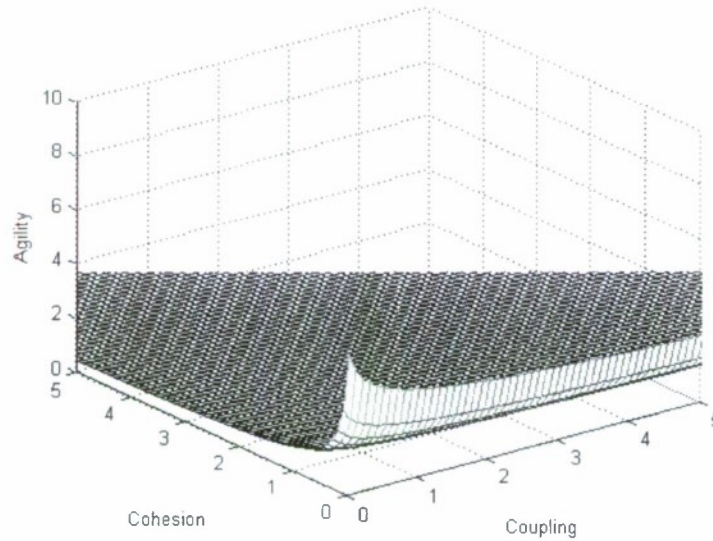


Figure 6.20. Agility Plot

Example Calculations

The working example for the adaptability calculations extends the concurrence example from earlier in the chapter. Adaptability addresses the effects of the SOSI's structure by measuring the level of coupling between each node and the SOSI. Adaptability also addresses the cohesion of each Node. Figure 6.21 shows the concurrently available SOSICs and the Nodes to which the Elements have been allocated. The set of Nodes, $N1 = \{n1,1, n1,2, n1,3\}$ in Figure 6.21 graphically illustrates the allocation of elements. Figures 6.22, 6.23, and 6.24 show the Nodes represented as CPNs, in which a graph analysis of the CPN will identify the number of paths that connect inputs to the outputs.

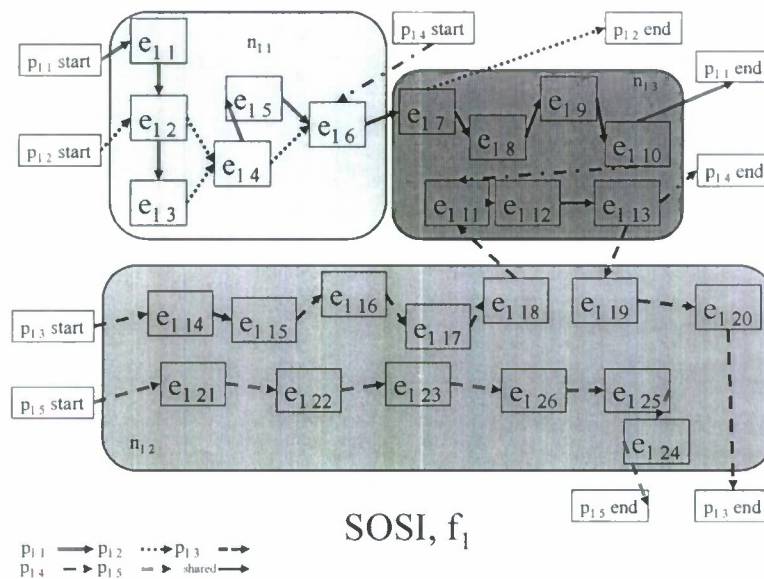
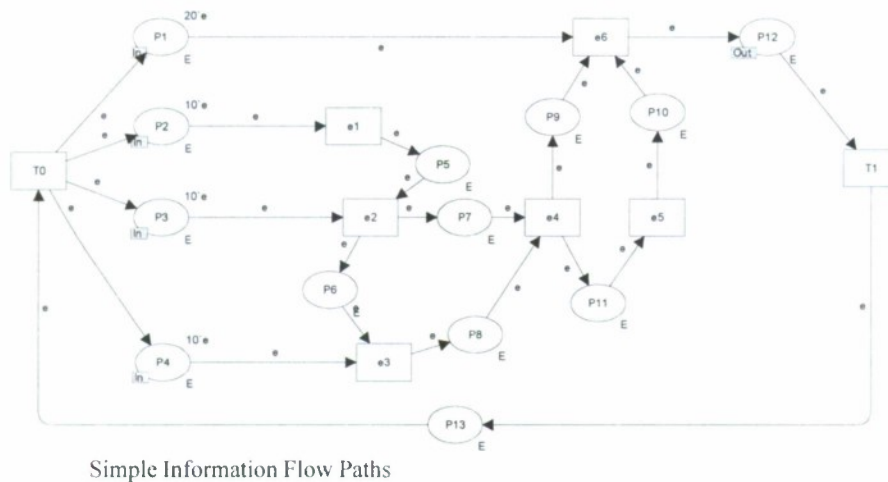


Figure 6.21. Working Example for Adaptability Calculations with Nodes

Cohesion

The cohesion computation begins with computing the number of paths that connect Node inputs with Node outputs represented in the CPN. Node $n1,1$ from Figure 6.21 will be the initial example. Figure 6.22 includes the common input and output transitions and the common place that connects input and output transitions. The ovals marked with an "in" box are the input places and the ovals marked with an "out" box are the output places. The common place is named P13. Next, the Farkas algorithm is applied to the modified graph to identify the invariants that include the common place P13. The invariants that include P13 identify the paths that connect the node inputs to the node outputs. The number of paths in this case is 11. There are 4 inputs 1 output therefore cohesion for Node $n1,1$ is $11/4$. Figures 6.23 and 6.24 show the CPN for Node $n1,2$ and Node $n1,3$ from the example. The results of the Cohesion computation appear at the bottom of each diagram. Table 6.7 summarizes the Cohesion results of the example.



P1, P12	P3, P7, P9, P12	$I = 4, Q = 1$
P2, P5, P7, P9, P12	P3, P7, P10, P11, P12	$x = 4(I) = 4$
P2, P5, P7, P10, P11, P12	P3, P6, P8, P9, P12	$z = 11$
P2, P5, P6, P8, P9, P12	P3, P6, P8, P10, P11, P12	
P2, P5, P6, P8, P10, P11, P12	P4, P8, P9, P12	$\text{Coh}(n_{1,1}) = 11/4$
	P4, P8, P10, P11, P12	

Figure 6.22. CPN for $n_{1,1}$

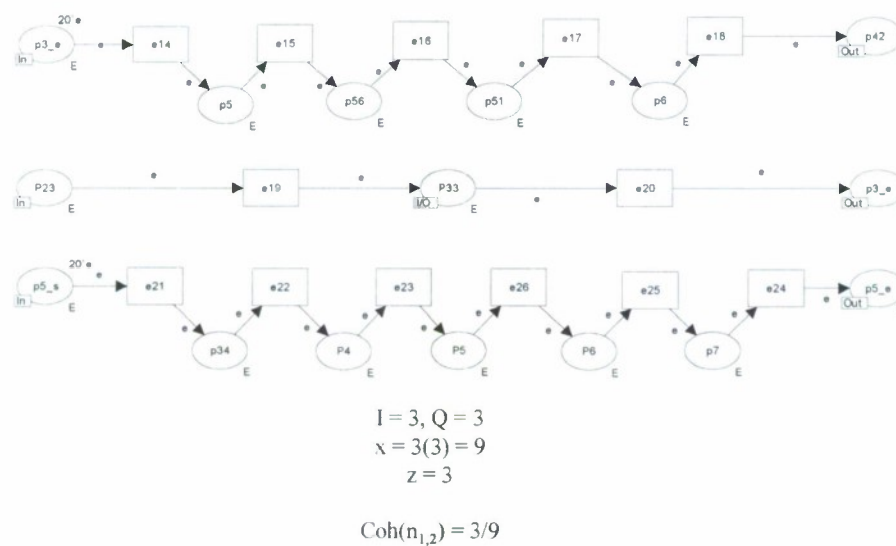


Figure 6.23. Colored Petri Net (CPN) for $n_{1,2}$

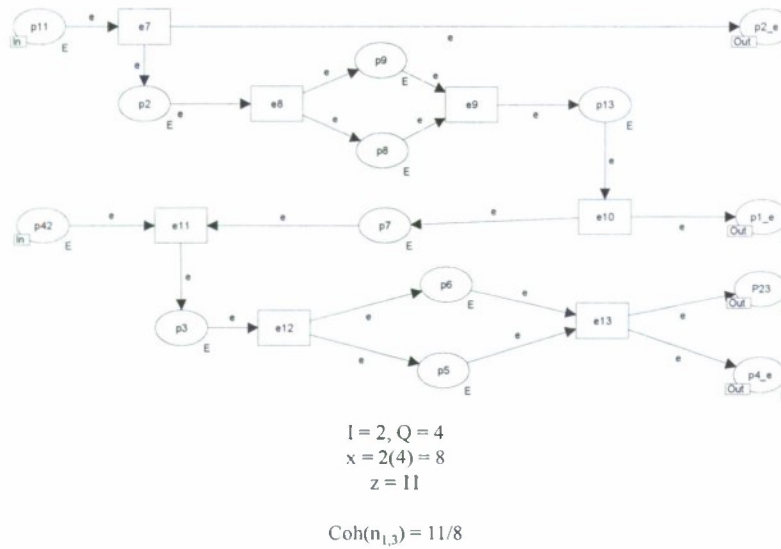


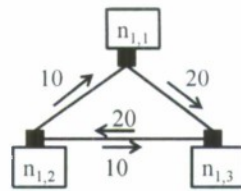
Figure 6.24. Colored Petri Net (CPN) for n1,3

Table 6.7. Cohesion Example

Node	I (Inputs)	Q (Outputs)	z (poss. paths)	x (total paths)	Coh
n1,1	4	1	4	11	11/4
n1,1	3	3	9	3	3/9
n1,1	2	4	8	11	11/8
SOSI fl					3/2

Coupling

The second component of Adaptability is Coupling. The coupling measure is a ratio of the number of messages sent by a Node to other Nodes in the SOSI and the total number of connections possible in the SOSI. In this example, 60 messages were generated by the scenario and there are three Nodes so there are three possible connections between Nodes in the SOSI. Figure 6.25 shows the results for the coupling calculations for SOSI fl. Node n1,1 exported 20 messages to Node n1,3 and zero to Node n1,2. That resulted in a coupling measure of $20/3 = 6.67$. The coupling measure does not include messages generated internally in the node - the only messages that are counted are those that cross from one Node to another.



		Importer			
		n _{1,1}	n _{1,2}	n _{1,3}	Total
Exporter	n _{1,1}	X	0	20	20
	n _{1,2}	10	X	10	20
	n _{1,3}	0	20	X	20

$$\text{Coup}_x(n_{ki}) = \frac{\sum_{i \neq j} |M_x(n_{ki}, n_{kj})|}{C}$$

$$\text{Coup}_x(n_{1,1}) = 20/3 = 6.67$$

$$\text{Coup}_x(n_{1,2}) = 20/3 = 6.67$$

$$\text{Coup}_x(n_{1,3}) = 20/3 = 6.67$$

$$C = 3(2) / 2 = 3$$

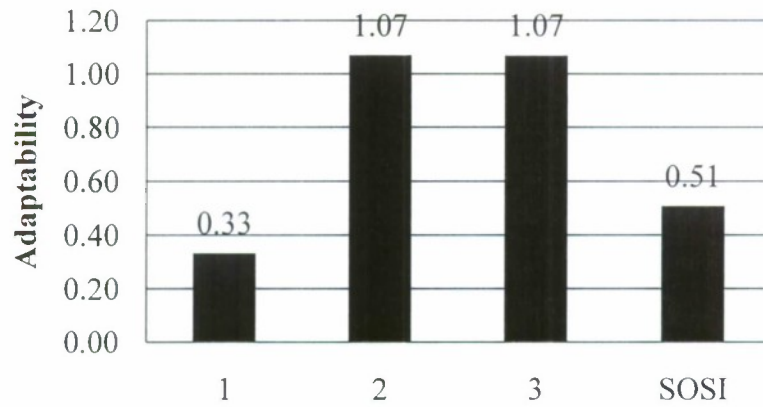
$$\text{Coup}_x(f_1) = 6.67$$

Figure 6.25. Coupling Results for SOSI fl

With the calculation for Coupling and Cohesion completed, adaptability can be calculated. The Adaptability results for the example are shown in Figure 6.26. This example shows how the measures are calculated. The numbers are meant for comparison among architecture alternatives. The results here show that the Nodes have different values for Adaptability.

Degree of Reuse

The set of interest for these calculations is the set of SOSIC P1 associated with SOSI fl. There are three SOSIC represented. Figure 6.27 shows a Venn diagram that shows the SOSICs supported by the Elements. The overlap of the Venn diagram illustrates the Degree of Reuse of the Elements. For example, Element e1,8 supports all three SOSICs.



Node	Coup	Coh	Adapt
1	1.00	3.96	0.33
2	4.67	0.63	1.07
3	4.00	0.65	1.07
SOSI	3.22	1.74	0.51

Figure 6.26. Adaptability Results for SOSI fl

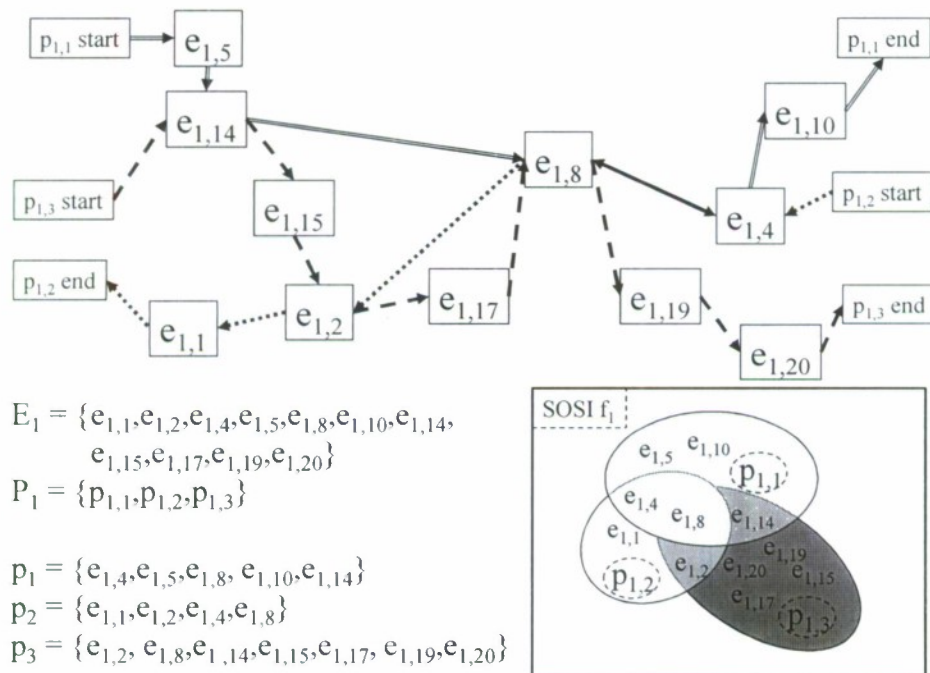


Figure 6.27. SOSI fl with Three SOSIC

The Degree of Reuse of each Element is calculated in the numerator of Equation 3.6. The degree of reuse for each Element is 9, 8, and 11 for SOSIC p1,1, p1,2, and p1,3, respectively. This results in degree of reuse values for the SOSIC p1,1, p1,2, and p1,3 of 9/5, 8/4 and 11/7, respectively. The SOSI Degree of reuse is 16/11. The Overlap calculations are summarized in Tables 6.8 and 6.9. Each x in Table 6.8 indicates the SOSICs that include a particular Element. The total (tot) shows the total number of SOSICs for that Element. Table 6.9 shows the Degree of reuse for the SOSIC and SOSI. It also shows the Exclusiveness measures and high reuse elements results for the SOSI. Higher Exclusiveness means that the SOSI has more ability to execute SOSIC concurrently because there are few Elements that are reused by the SOSICs. An Exclusiveness score of 1 means there is no reuse of Elements in the SOSI. Low overlap scores mean that reuse of Elements among the SOSIC is high increasing the potential for resource conflict.

Table 6.8. Degree of Reuse Example Data

	e1,1	e1,2	e1,4	e1,5	e1,8	e1,10	e1,14	e1,15	e1,17	e1,19	e1,20
p1,1			x	x	x	x	x				
p1,2	x	x	x		x						
p1,3		x			x		x	x	x	x	x
tot	1	2	2	1	3	1	2	1	1	1	1

Table 6.9. Degree of Reuse Calculation for SOSI f1

	Num Elements	Total Reuse	Degree of Reuse		Summary	
p1,1	5	9	9/5	1.8	High Reuse	3
p1,2	4	8	8/4	2.0	e1,8	
p1,3	7	11	11/7	1.57	Exclusiveness	0.69
f1	11	16	16/11	1.45		

The working example shows that SOSIC p1,3 has the highest degree of reuse and SOSIC p1,2 has the lowest; therefore, we can conclude, in relative terms, that p1,3 has less potential for resource conflict than p1,2. Degree of Reuse also indicates Elements that may be used beyond their capacity. In this simple example, Element e1,8 is a highly reused Element. It is used by all SOSICs indicating that the Element might be an integral component of the SOSI and its utilization deserves further analysis.

Agility

Agility is Adaptability scaled by Exclusiveness. Figure 6.28 shows the effect of Exclusiveness on the overall Agility of the SOSI. In this example, Agility is less than Adaptability because the Degree of Reuse is high among the Elements of the SOSI. The reuse reduces the ability of the SOSI to execute SOSICs concurrently and reduces the overall Agility of the SOSI.

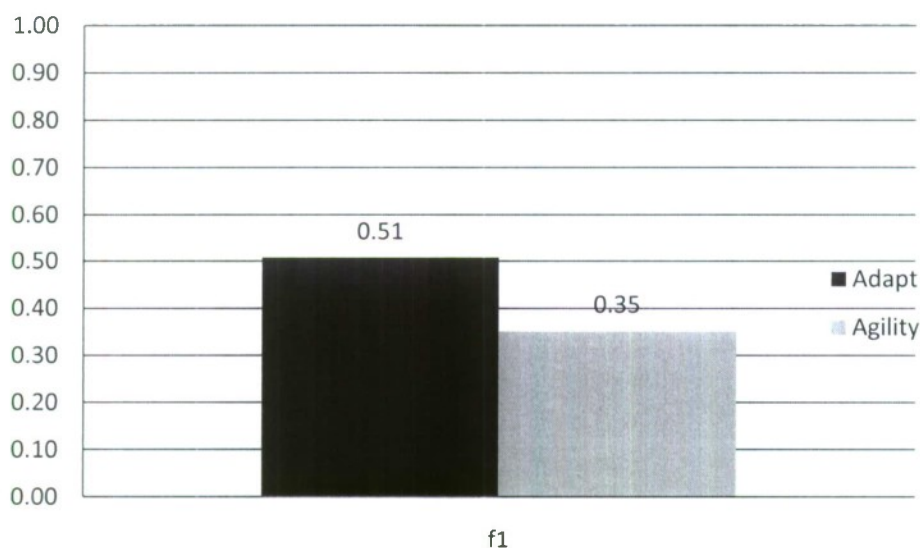


Figure 6.28. Example Results for Adaptability and Agility

Summary

This section defined specific SOS properties that are different from the traditional definitions. The definitions and the associated properties help structure the SOS so that a boundary can be defined and assessments conducted. The measures provide a way to make comparisons of alternative architectures very early in the development process. Adaptability measures the ability of the SOSI to adapt different structural configurations. Degree of Reuse measures the ability of the SOSI to execute multiple capabilities concurrently. The product of Exclusiveness and Adaptability provides a measure of the SOSI's Agility. Section 6.4 presents the methodology used to gather the information required to generate the executable model and conduct the analysis.

6.4 Methodology

This section describes the methodology used to create the SOSI Architectures and SOSI alternatives for comparison. The operational architecture views that describe the capability required by the organization and their associated system architecture views are the inputs. The system views are combined in a SOSI Architecture that merges the rule and data model and represents the processes described in the system architectures as SOSICs. The SOSI alternatives are created from the SOSI Architecture and transformed into an executable form for a static and dynamic analysis of the interaction between Elements and Nodes of the SOSI.

Figure 6.29 shows the methodology graphically. The operational architectures and system architectures that realize the capability are taken as input. The dotted line connecting the system view box and the operational view box shows that they are not independent. System architecture views are developed to meet the needs of the organization described in the operational architecture views. The combined SOSI Architecture model (represented in the center box) illustrates multiple system architecture views realizing multiple capabilities. The ovals in the center box represent the SOSICs. SOSIC_{jk} represents the *k*th capability realized by the *j*th system architecture. The overlapping SOSICs illustrate the reuse of Elements as multiple system architecture views are merged to create a combined model of the SOSI Architecture. The process of merging of the system architecture views reveals Elements that are used by more than one SOSIC. Then the combined model of a SOSI is transformed into an executable form. If the behavior of the executable is acceptable, then Adaptability and Agility are calculated using the SOSI executable model and architecture view. If its behavior is not acceptable, then the SOSI Architecture must be modified to correct inaccurate behavior. The feedback into the SOSI Architecture representation ensures the behavior of the executable can be traced to the model representation.

One of the challenges in SOSI analysis is the dynamic nature of the SOS. Recall SOS property 2, the set of elements that compose the SOS changes over time, meaning that potentially the Elements chosen to compose a SOSI will also change. The process described as part of this methodology analyzes the SOSI for those periods where the Elements of the SOSI are constant. Figure 6.30 shows the SOSI's piecewise constant nature. The SOSI existed in an initial configuration prior to time *t*₁. The changes at times *t*₁, *t*₂, and *t*₃ are meant to illustrate significant changes to the SOSI environment—and so the SOSI must change in order to address the new operating environment. Many things can happen to change the composition of the SOSI. When a change occurs, if the SOSI is not adaptable enough to accommodate the new environment and new Elements or SOSIC are required, then the SOSI must change and a new SOSI is instantiated. The analysis of each SOSI, therefore, is done piecewise for the time period that the SOSI is not expected to change Elements. The next section describes the specific comparisons accomplished by the methodology.

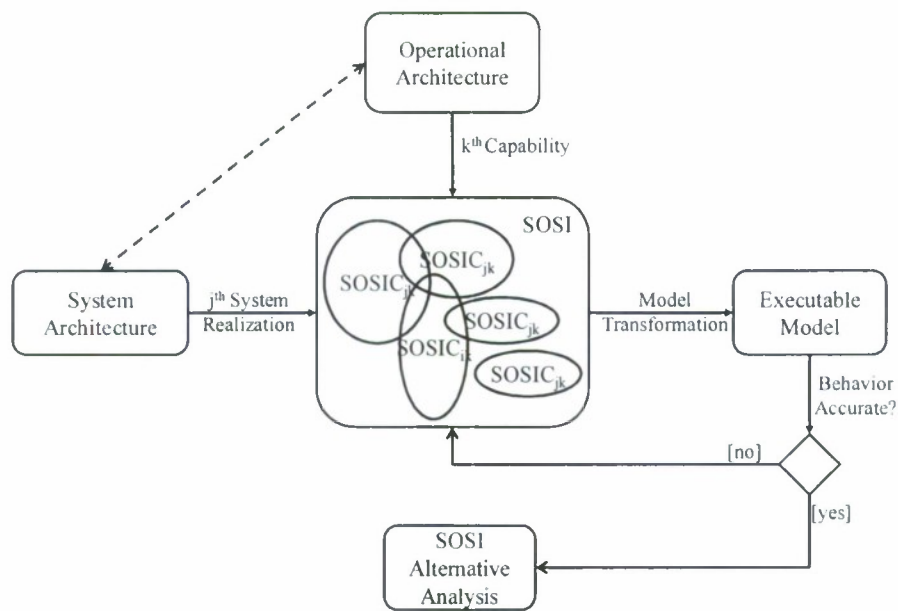


Figure 6.29. Methodology

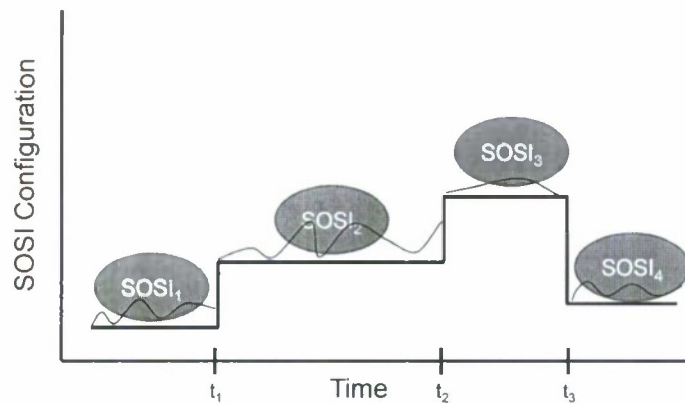


Figure 6.30. Piecewise Constant SOSI

Comparisons

There are many ways that a SOSI can change. Some require a change to the SOSI Architecture and others only change a particular instance of a SOSI Architecture. Given that the methodology describes the SOSI in terms of Nodes, Elements, and SOSICs. If the type of Nodes, Elements or SOSIC change then a change in the SOSI Architecture is required, because the relationships between the Elements, Nodes and SOSIC must be modified to reflect the change. If the number of instances of Elements, Nodes or SOSICs change then the SOSI

representation must be modified, but the SOSI Architecture remains the same as long the new SOSI does not attempt to associate Elements or Nodes that are unrelated in the SOSI Architecture representation.

Given the types of changes that can occur, the problem is scoped in the following way. SOSI groups are SOSI alternatives composed of the same set of Elements and based on the same SOSI Architecture. There are two comparisons, Comparison A and Comparison B. Comparison A compares SOSI within a SOSI group. That means that the instances of Elements and SOSIC sets are held constant and the alternatives are differentiated by the allocation Elements to Nodes. Comparison B compares the assessment results between SOSI Architecture alternatives using the results from Comparison A. This way the effects of adding and deleting Elements to the SOSI can be assessed by comparing SOSI groups. Comparing SOSI groups created from different SOSI Architecture assesses the relative Adaptability and Agility of the SOSI Architecture.

Assumptions

Because the thrust of the analysis is based on the SOSI characteristics, assumptions concerning the performance of individual systems, the Operational Capabilities, and the execution scenario need to be made.

The Elements that interact within the SOSI are assumed to be interoperable. If there is a noted data dependency, then it is assumed that the protocol and associated communications details are sufficient.

The behavior model of individual Elements is assumed accurate. The Elements of the SOSI are assumed to perform in the modeled manner. Because concurrently executing capabilities are modeled, we assume that the Elements used to accomplish the capability do indeed enable the capability and that they have been modeled accurately.

The modeled capability that is represented by the Operational View is accurate. The operational activities described in the OV products are accurate and provide the capability modeled. The performance of the capability is assumed to meet the stakeholder's requirements.

The mission of the implementing organization is known. In order to identify required capabilities, the organization's mission must be known before analysis can begin. The mission provides the purpose for the SOSI Architecture.

The required set of capabilities does not change. The number of each type of capability may change, but the organization does not modify the list of required capabilities. The measures are relative, so the set of capabilities cannot change in order to ensure that the performance characteristics of the candidate architectures can be compared.

Behavior models exist for each Element modeled. Because the Elements represent the constituent systems and their functions, these models must be available so they can be executed to ensure that the capabilities required are provided by the mix of Nodes and Elements in the SOSI.

Given the methodology and the assumptions described above, the following are the steps required develop the structural, operational, and behavior information required to construct the combined SOSI architecture.

SOS Instance Assessment Process

The process developed realizes the requirements of the methodology and uses concrete representations for the types of architecture products required to combine the system architecture views and create the executable model. The Department of Defense Architecture Framework (DODAF) (DISA, 2007) provides a framework for the representation of the various architecture views required by the analysis process. The DODAF prescribes multiple views represent pertinent aspects of the architecture from different perspectives. Operational views describe aspects of the operational architecture and system views describe aspects of the system architecture. While, the DODAF does not stipulate a modeling language, the process described here uses the Unified Modeling Language (UML) (OMG, 2007) to represent the relationship between components of the models because it uses a high level data model that can be used to facilitate the transformation to the executable form. The executable form used for the analysis is Colored Petri Nets (CPN) [Jensen, 1991]. Colored Petri Nets possess the formal execution semantics and a formal graph theoretic representation. The formal execution semantics of the CPN enable an accurate transformation of the UML execution semantics for an accurate representation of the modeled behavior of the SOSI which enables the model to simulate the interaction of the Elements and Nodes of the SOSI in order to compute Coupling. The formal graph theory that underlies the CPN enables the use of tan invariant analysis of the graph generated by the transformation for the computation of Cohesion. The process, Figure 6.31, is described in seven steps.

1. Identify/Develop Operational Views of the Architecture for the desired capabilities.
2. Identify/Develop System Views of the Architecture that realize the capabilities describe in Step 1.
3. Produce SOSI Architectures. Combine the system views of the architectures from Step 3 to produce a SOSI Architecture that represents an architecture alternative. (Multiple alternatives can be developed from the information gathered in steps one and two.) The SOSI Architecture uses DODAF system architecture view products to model the architecture information.
4. Transform the SOSI alternatives created from the SOSI Architectures into CPN for each alternative.
5. Develop an execution scenario that complements the organization's operational environment.
6. Conduct the analysis for each alternative using the assessment measures, Cohesion, Coupling, Degree of Reuse, Adaptability and Agility.

7. Draw conclusions and complete the comparison analysis.

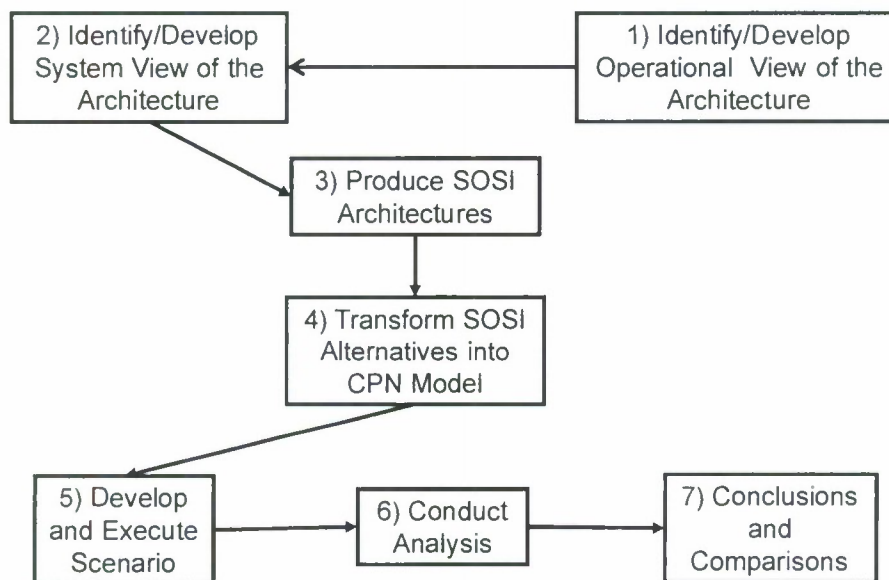


Figure 6.31. SOSI Analysis Process

Step 1. Identify/Develop Operational View of the Architecture

This step identifies the DODAF operational architecture view products (OV) that describe the organization's required capabilities. It also identifies the organizational roles that are expected to interact to provide the specific capability. Furthermore, it identifies the required information exchanges of each role, and identifies the operational activities that must be executed by each to role accomplish the capability. Then the capability must have a rule and data model defining the data exchanged that rules governing the exchange. Finally, an operational activity model shows how the operational activities interact and the data passed between them.

Each product defines a particular aspect of the operational architecture. In order to meet the methodology requirements, the following operational view (OV) products are required:

OV-2 Operational Node Connectivity Description describes the roles that organization requires to accomplish the capability, and each role's information needs. Information needs are described the form of inputs and outputs for each role that are required to accomplish the capability.

OV-3 Operational Information Exchange Matrix identifies the data exchanged between roles and the attributes that compose that data. The OV-2 and OV-3 are related and pivot on the information represented by the data transferred between roles.

OV-5 Operational Activity Model describes the operational activities that must be accomplished to provide the capability. The OV-5 can also show the roles that should contain those activities. The relationships between activities and the data produced and consumed by each activity are also important in this product.

OV-6a Operational Rules Model describes the rules that govern the behavior of the operational activities.

OV-7 Logical Data Model describes abstract relationships between Elements and Messages and the attributes of the Messages.

This is a simple example used to show the primary data elements provided by the products. There are two operational architectures modeled that represent two different capabilities, Capability 1 and Capability 2. The products for each operational architecture view will be shown together. Capability 1 is represented in OA1. It has two roles Sender and Receiver. Capability 2 has two roles, Receiver and Executor. There are messages and simple operational activities that implement a rudimentary rule model. The Operational Node Connectivity Diagrams, OV2s, shown in Figure 6.32, describe the roles required, their interaction, and the data exchanged for the two capabilities. The lines connecting the roles identify the information exchanged between roles. In this case, the Receiver receives Msg1 and sends Msg2 and Msg3.

The Operational Information Exchange Matrices, OV-3s add more detail about the data exchanged. The OV-3s are shown in Table 6.10. In this simple case, the OV-3 shows the size and type of the data exchanged between roles. For example, Msg1 is sent from role Sender to role receiver. Msg1 is of type Text and expected to be 10 characters long.

The Operational Rules Model, OV-6a shows the rules that govern the behavior of the activities that are used by the roles described in the OV-2. Table 6.11 represents the rule models for the example. The simple rules show that Msg1 is received by the Receiver who then decides whether to send a Msg1 or a Msg2 depending on who sent the message.

The Operational Data Models, OV-7s in Figure 6.33 show the messages and roles that appear in the Operational Architecture Views for each capability.

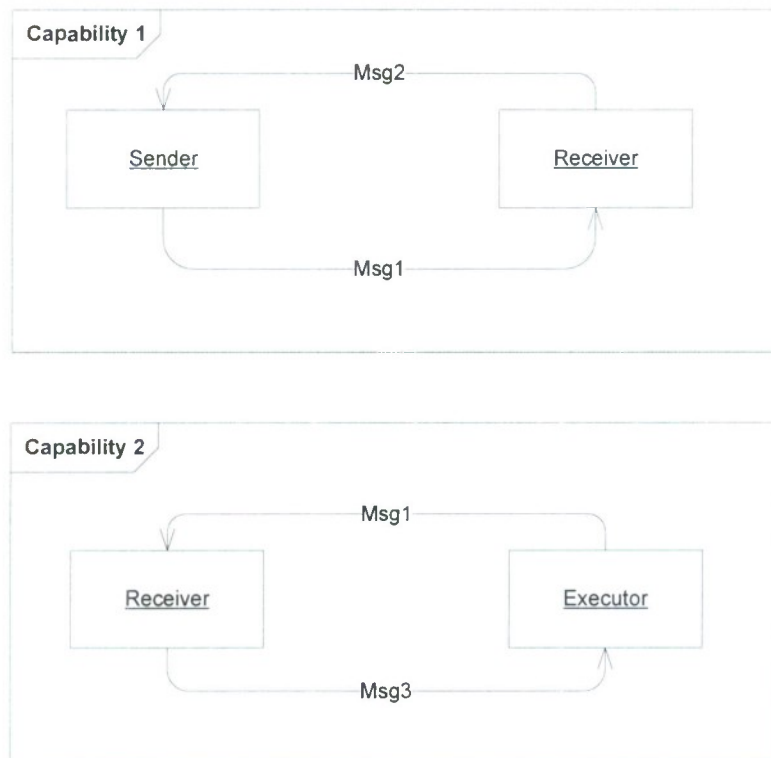


Figure 6.32. Operational Node Connectivity Diagrams, OV-2s

Table 6.10. Operational Information Exchange Matrix, OV-3

Capability 1				
	Sender	Receiver	Type	Length
Msg1	Sender	Receiver	Text	10
Msg2	Receiver	Sender	Text	20
Capability 2				
	Sender	Receiver	Type	Length
Msg1	Executor	Receiver	Text	10
Msg3	Receiver	Executor	Text	32

Table 6.11. Operational Rules Model, OV-6a

Capability 1	
Sender	if msg2 send msg1
Receiver	if msg1 = yes send msg2
Capability 2	
Receiver	if msg1= no send msg3
Executor	if msg3 then send msg1

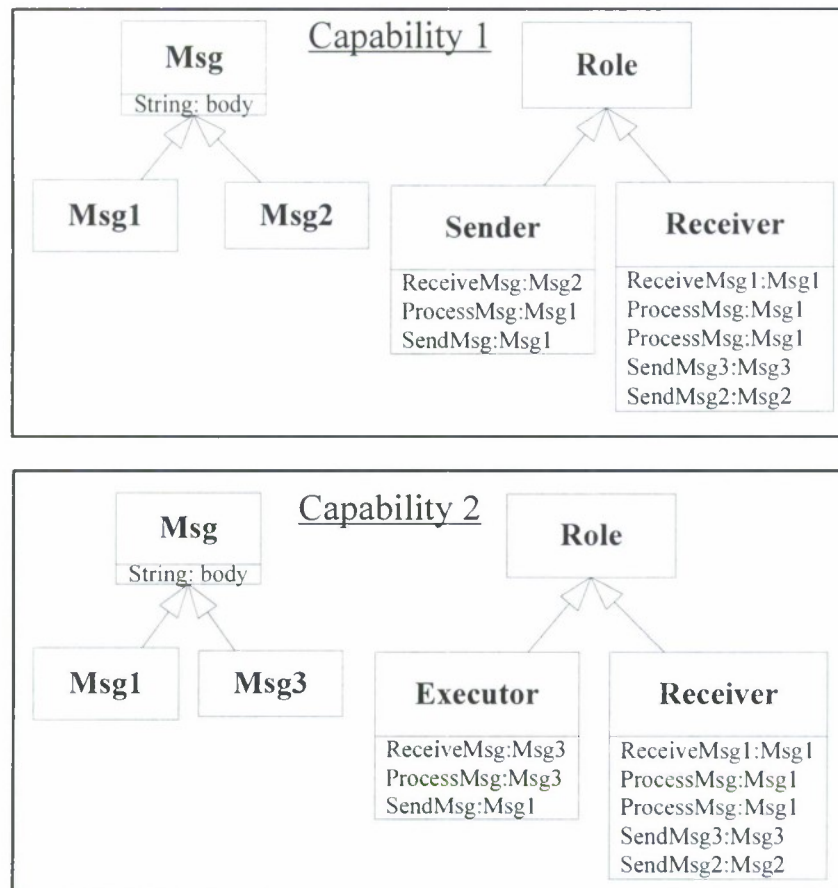


Figure 6.33. Operational Data Model, OV-7

The final diagram, Figure 6.34, is the Operational Activity Model, OV-5. This diagram uses the information from the other operational views. The activities implement the rules shown in the OV-6a and the data passed from activity to activity is represented by the OV-7. The interacting roles and the information exchanged are shown in the OV-2 and OV-3.

The Operational Architecture products presented above represent the minimum information required by the methodology. The system view products for step 2 realize the operational capabilities illustrated by the operational views developed for step 1.

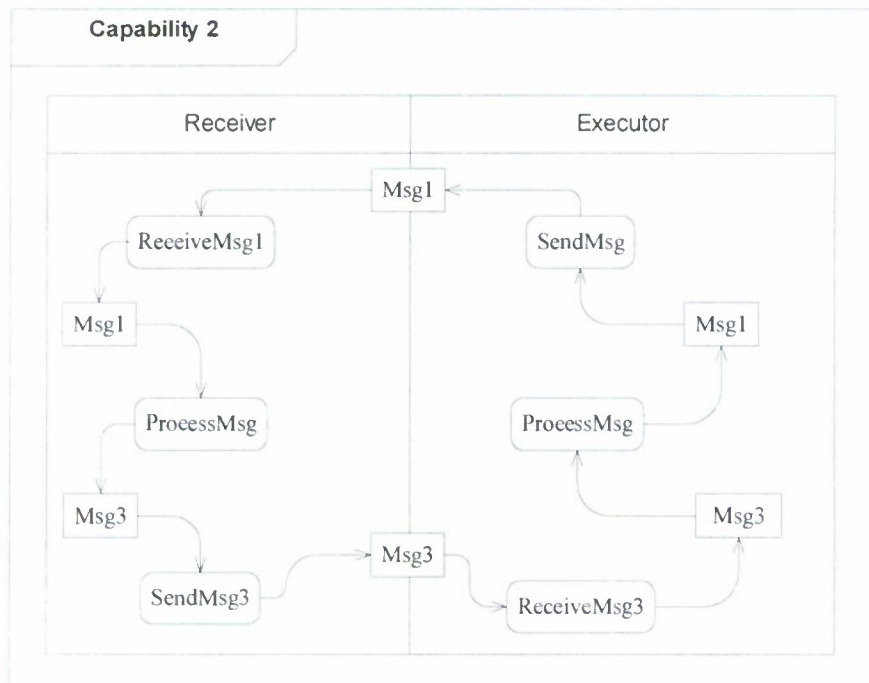
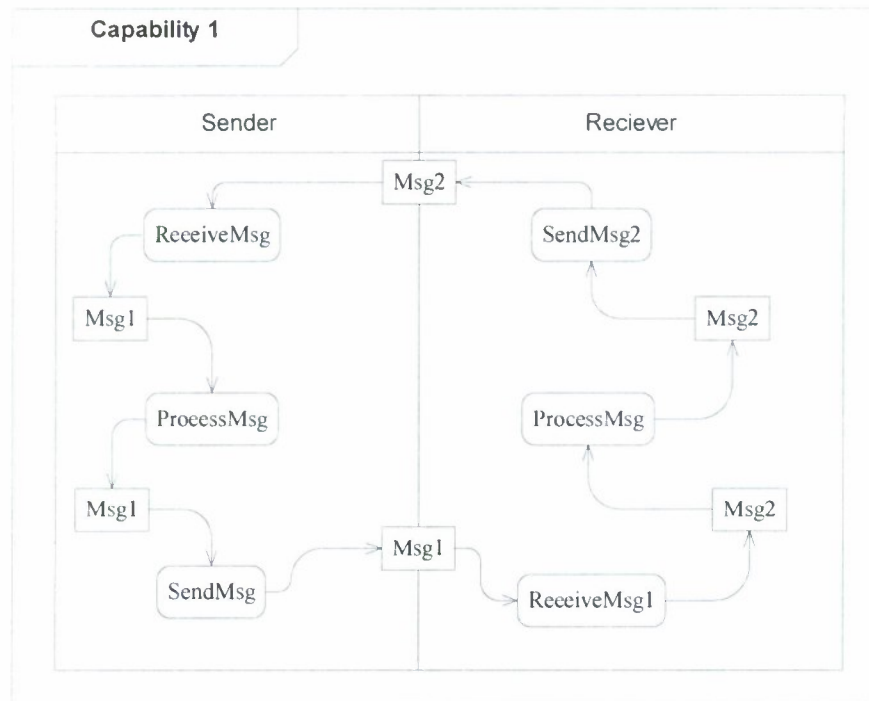


Figure 6.34. Operational Activity Diagram, OV-5

Step 2. Identify/Develop System View of the Architecture

This step involves identifying the Elements that will be grouped in Nodes to accomplish the capability. Step 2 describes how the capabilities will be provided and with what Elements. The system architecture must include the following system view products:

SV-1 Systems Interface Description depicts the Nodes and the Elements resident on those Nodes. It also describes the interfaces between Elements and Nodes. The SV-1 is related to the OV-2 in that the SV-1 shows which Elements are fulfilling the roles described in the OV-2. The interfaces of the SV-1 map to data exchanges between roles in the OV-2.

SV-6 Systems Data Exchange Matrix describes the data exchanges between systems and the attributes that compose those exchanges. The information described in the OV-3 must be reflected in the SV-6.

SV-10a Systems Rules Model is the rule model that governs the behavior of the system functions that realize the operational activities.

SV-11 Physical Schema is the physical data model used to show the relationships between Elements, Nodes and Messages. The schema must reflect data embodied in the OV-7.

SV-4 Systems Functionality Description depicts system functions and the data flows between functions. This product can also provide the individual Element behavior model, which is a union of the behaviors of that particular Element type described in all the capabilities where the Element type is employed. This view is analogous to the OV-5.

SV-5 Operational Activity to Systems Function Traceability Matrix provides the mapping between the implemented system functions and the required operational activity. This is not always a one-to-one mapping. There are many instances where multiple system functions are required to accomplish a single operational activity and vice versa.

The example develops two system architecture views that realize the capabilities described by the operational architecture views identified in step 1. SA-1 realizes Capability 1 and SA-2 realizes Capability 2.

The System Interface Descriptions, SV-1, identify the Elements will realize the roles defined in the OV-2. Figure 6.35 shows the SV-1s for both capabilities. It shows the Elements, Nodes they are member of, and the Messages exchanged between the systems. System1 represents the Sender. System2 and System 3 represent the Receiver and Executor, respectively.

Similar to the OV-3, the Systems Data Exchange Matrices, SV-6s, shown in Table 6.12, identify the details of the data exchanged between the systems. In this case the SV-6s describe the size and type of the Message classes exchanged between systems. The table reflects the details of the interface between systems shown in the SV-1s.

The Physical Schemas, SV-11s, in Figure 6.36 show the physical format of the data that is exchanged between the systems. The SV-11 is the physical representation of the logical data model represented by the OV-7.

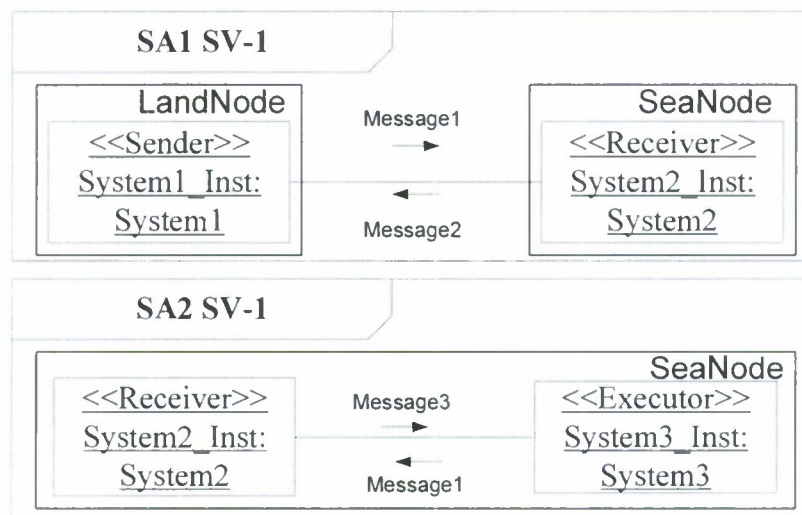


Figure 6.35. System Interface Descriptions, SV-1s

Table 6.12. Systems Data Exchange Matrix, SV-6

SA1				
	Sender	Receiver	Type	Length
Msg1	System1	System2	Text	10
Msg2	System2	System1	Text	20
SA2				
	Sender	Receiver	Type	Length
Msg1	System3	System2	Text	10
Msg3	System2	System3	Text	32

The SV-5s, shown in Table 6.13, show the mapping between operational activities the Element functions that realize the activity. This methodology defines a type of Element as a system. For example, the role Sender has three operational activities, ReceiveMsg, ProcessMsg and SendMsg that are all mapped to one Element function Sys1Action that is part of the System1 Element.

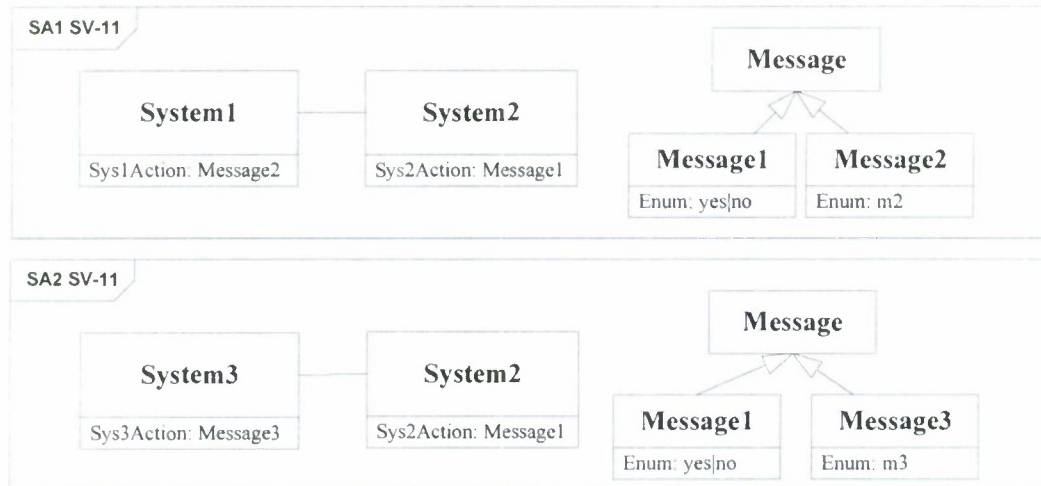


Figure 6.36. Physical Schema, SV-11

Table 6.13. Operational Activity to Systems Function Traceability Matrices, SV-5s

SA1		System 1	System 2
Role	Op Activity	Sys1Action	Sys2Action
Sender	ReceiveMsg	X	
	ProcessMsg	X	
	SendMsg	X	
Receiver	ReceiveMsg		X
	ProcessMsg		X
	SendMsg2		X

SA2		System 2	System 3
Role	Op Activity	Sys2Action2	Sys3Action
Receiver	ReceiveMsg	X	
	ProcessMsg	X	
	SendMsg3	X	
Executor	ReceiveMsg3		X
	ProcessMsg		X
	SendMsg		X

The final representations describe the behavior of the system architecture. The Systems Rules Model, SV-10a, Table 6.14 , defines the rules that govern the behavior of the Elements of the system architecture views. The SV-10a is implemented by the Systems Functionality Description, SV-4. For example, if System1 receives a msg2 then it should send a msg1.

Table 6.14. Systems Rules Models, SV-10a's

SA1
System1
if msg2 send msg1
System2
if msg1= no send msg3
if msg1 = yes send msg2

SA2
System2
if msg1= no send msg3
if msg1 = yes send msg2
System3
if msg3 send msg1

The SV-4 is represented as a UML Activity Diagram. The SV-4 provides an integrated model of the previously described system architecture view products. The SV-4s, Figure 6.37, implement the rule models and use the data described in the respective SV-11s to model the behavior of the interaction between the Elements of the architecture. This SV-4 shows the interaction of System1 and System2 in SA1 and the interaction of System3 and System2 in SA2. The behavior of System1 and System2 implement the rules modeled in the SV-10a. Message2 and Message1 are represented on the ports on the actions.

Step 3. Produce SOSI Architecture

Step 3 combines the system architecture views to produce a architecture model that represents the combination of the system view products described in step 2. Combining the system views ensures a concordant architecture that includes all the system architecture behavior models in combined system architecture view. The behaviors for each Element type must be combined, and the data models must be correlated to provide the data described in the SV-1 and SV-6 of each capability. This single UML model represents each capability in at least one SOSIC that is modeled in an Activity Diagram and viewed as a SV-4. This provides an opportunity to ensure that the Physical Schema and the Rule Model are concordant across the Activity Diagrams.

The process of merging the systems architecture begins with the SV-1s. The SOSI Architecture must represent the types of Nodes that the Elements will occupy. Each SOSI alternative generated from the SOSI Architecture will need a unique SV-1 showing the Nodes that the Elements occupy for that alternative. Figure 6.38 is an SV-1 for a SOSI

alternative. It shows that an instance of System1 is assigned to a LandNode and the instances of System2 and System3 are assigned to a ShipNode.

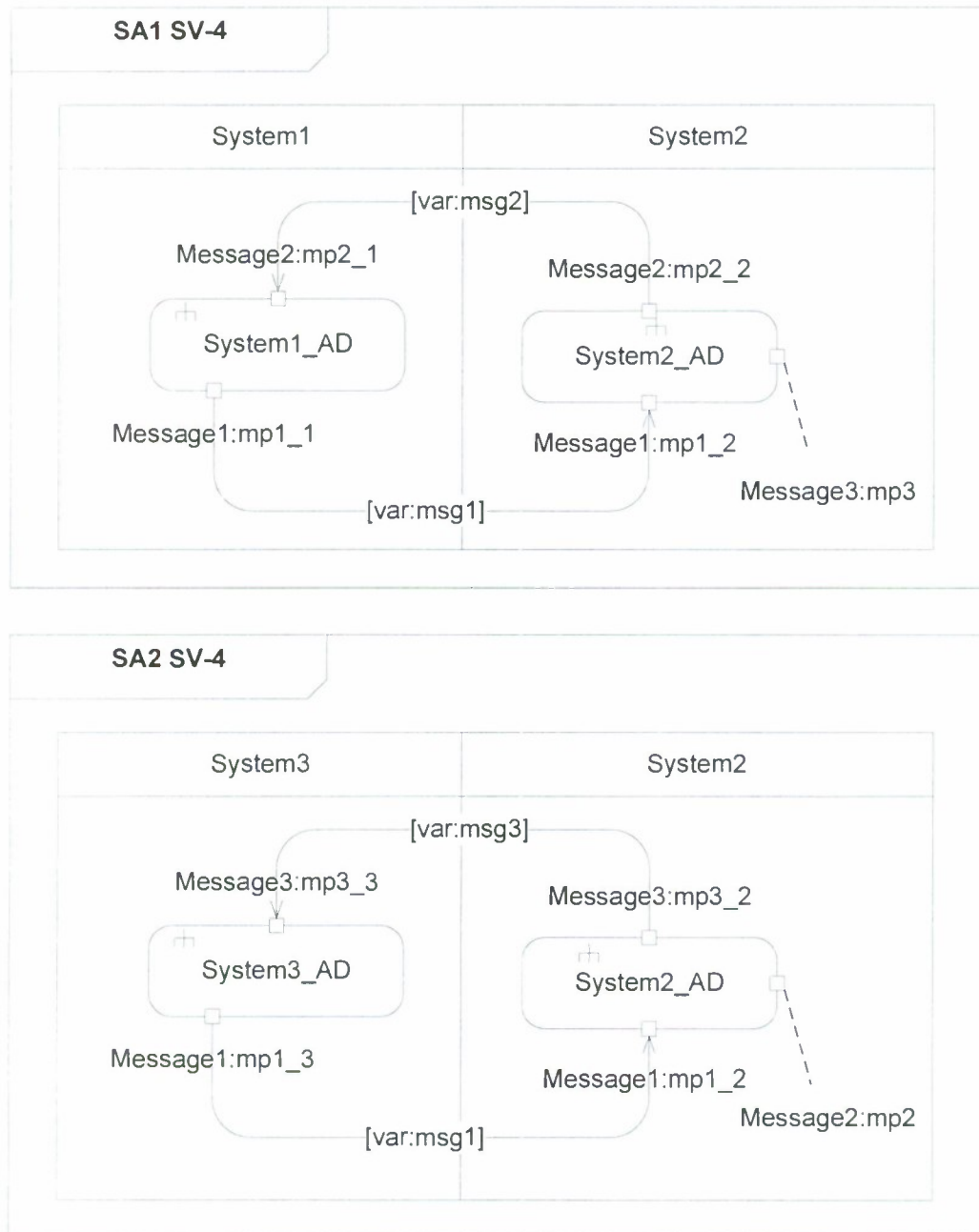


Figure 6.37. Systems Functionality Description, SV-4

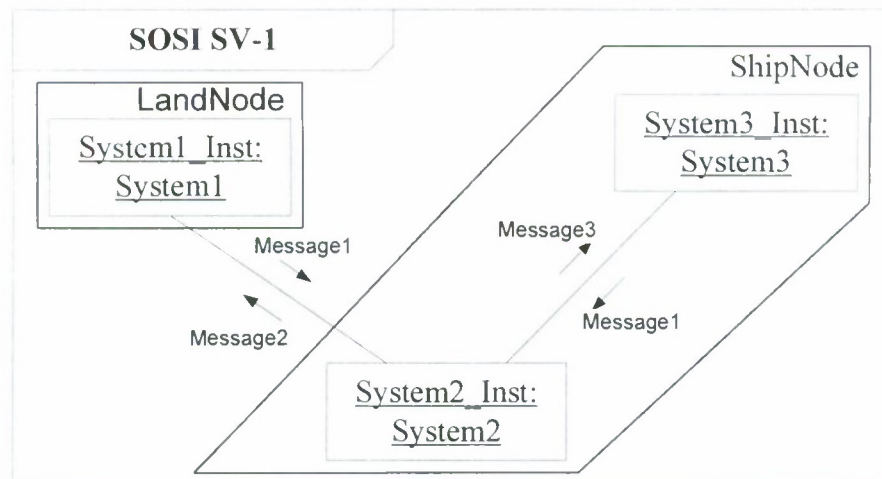


Figure 6.38. SOSI Architecture SV-1

The SV-6, shown in Table 6.15, merges the SV-6s from the two system architecture views presented in step 2. System2 appeared in both architecture views. It appears only once in this view and will be used in two SOSICs. System2 processes two types of messages. The combined behavior model of System2 should contain system functions that address both messages. This is an example of the importance of merging the system views.

Table 6.15. SOSI Architecture SV-6

SA1				
	Sender	Receiver	Type	Length
Msg1	System1	System2	Text	10
Msg1	System3	System2	Text	10
Msg2	System2	System1	Text	20
Msg3	System2	System3	Text	32

The SV-11s are merged to ensure that all the data in both system architecture views is represented in the SOSI architecture. Figure 6.39 is the merged SV-11. The merged SV-11 shows that System2 has an association with both System1 and System3.

Table 6.16 is the merged rule model, SV-10a. This table reveals that System2 must respond with two types of messages msg2 or msg3 depending on the content of the input message msg1.

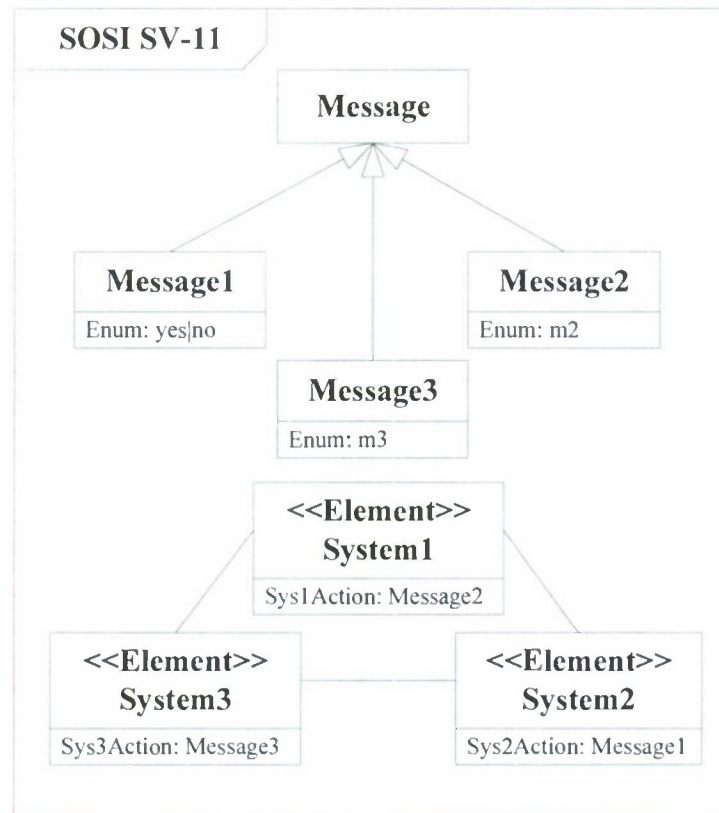


Figure 6.39. SOSI Architecture SV-11

Table 6.16. SOSI Architecture Rules Models, SV-10a

SOSI Architecture
System 1
if msg2 send msg1
System 2
if msg1= no send msg3
if msg1 = yes send msg2
System 3
if msg3 send msg1

With the rule and data represented, the SOSICs are modified versions of the SV-4s taken from the identified system architecture views. Figure 6.40 shows the SV-4s that represent this SOSI architecture. Notice that the Elements are identified with Element numbers. Element2 appears in both SOSICs and represents System2_Inst. The behavior of Element2 is represented by the Activity Diagram, System2_AD. The resulting Element behavior models reflect a union of the Element functionality represented in the system architecture views that

contain the same Element. This product also models the interaction of the Elements to realize the operational capabilities. This example has a one to one relationship between SOSIC and capabilities modeled. This is not the case in general. Multiple SOSICs can be developed that accomplish particular tasks using the same capability.

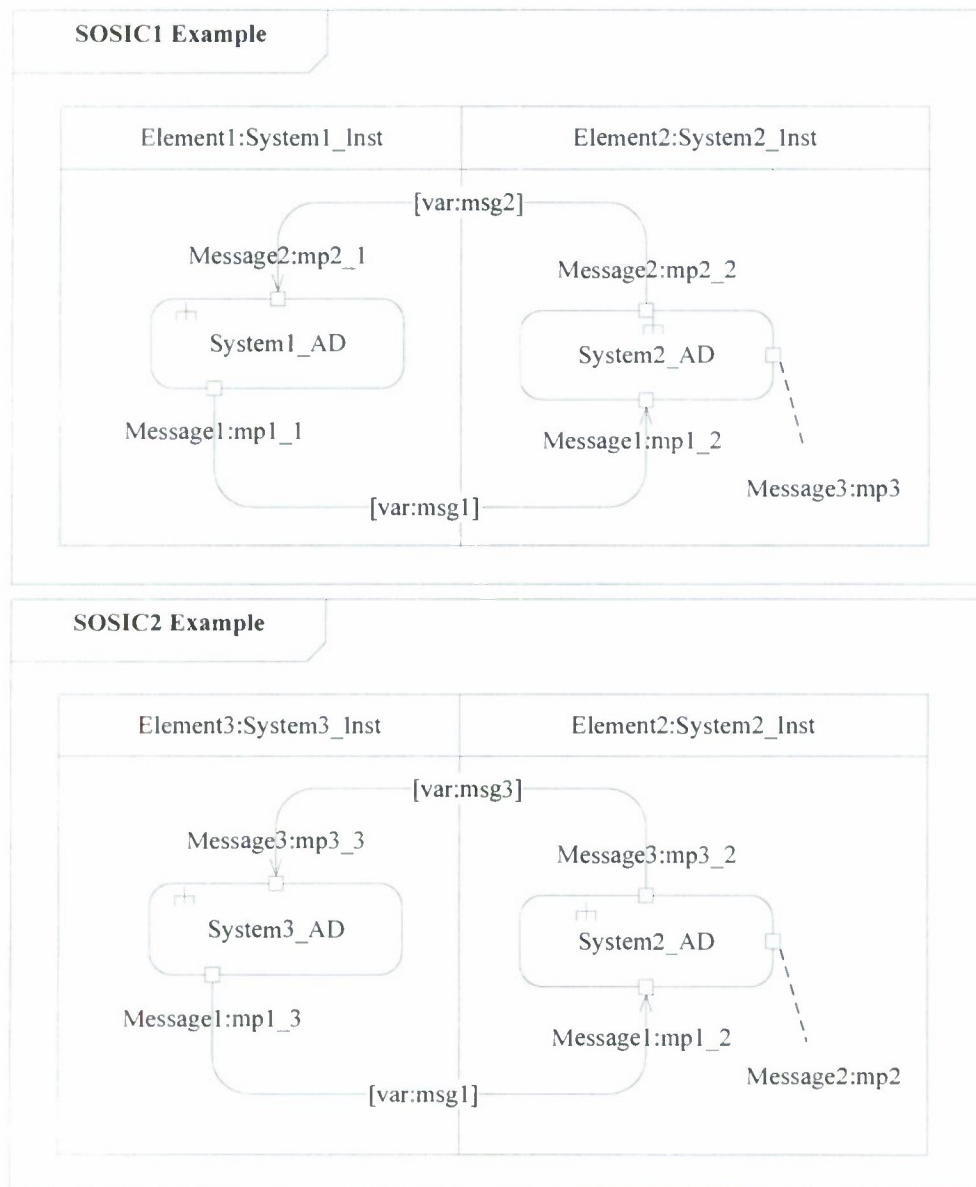


Figure 6.40. SOSI Architecture SV-4

Step 4. Transform the Combined Model into an Executable Form

Step 4 transforms the SOSI representation developed from the SOSI Architecture into a CPN model of the combined behaviors of the SOSI. This ensures that the concurrent use of the Element in multiple SOSICs is modeled in the CPN. The transformation of the UML Activity Diagrams into a CPN is detailed in Section 6.5.

Step 5. Develop an Execution Scenario

Step 5 develops an execution scenario that complements the organization's operational environment. This step also sets the parameters of the executable model and the initial conditions for the scenario that is to be executed. Additionally, the executable model is instrumented to facilitate the collection of appropriate data for the coupling measurements. This step is detailed in section 6.5.

Step 6. Conduct the Analysis for Each Alternative

Step 6 conducts the analysis for each alternative using the measures described in section 6.3. This step involves computing the measures for each architecture alternative and conducting an analysis of the results. This step includes computing Cohesion, Coupling and Degree of Reuse for each SOSI alternative. Then, the level of Adaptability and Agility must be computed to provide points of comparison among the SOSI alternatives. The Adaptability of each Node in the SOSI is computed so that the Nodes that are driving the SOSI assessment can be identified. This is a tool SOS architects can use to make decisions about where certain Elements should be placed in the architecture.

The last part of step 6 involves computing the Adaptability and Agility for the SOSI alternatives. This data provides the information required to make comparisons of the SOSI Architecture based on the results of the various SOSI alternatives.

Step 7. Draw Conclusions and Complete the Comparison Analysis

The final step completes the comparison analysis and draws conclusions as to why certain SOSIs assess better than others in terms of Coupling, Cohesion and Degree of Reuse.

Summary

This chapter described the methods used to evaluate SOSI Architecture alternatives. The methodology has seven steps. It begins with identifying the operational capabilities required by the organization and ends with an analysis of the performance of the SOSI with regard to its Adaptability and Agility. There are many ways to structure the comparison of SOSI alternatives. This methodology described the two comparisons that were used in the research. Comparison A compares SOSI alternatives that are the same except for the way Elements are distributed on Nodes. Comparison B uses the results of Comparison A to facilitate comparisons between SOSI Architecture alternatives. An example was presented that provides simple examples of the DODAF products used by the methodology. The entire methodology is outlined in this chapter with particular attention on steps one, two, three and six. These steps identify the capabilities required by the organization; develop the system architecture views that meet the operational architecture view requirements and produce the combined SOSI Architecture view that represents all the capabilities required. Finally step six discusses the way the comparisons are carried out. The details of the transformation, step four and five, from the UML Activity Diagram to the CPN are detailed in section 6.5.

6.5 Transformation

This chapter describes the transformation from the UML Activity Diagrams that describe the various SOSICs that the SOSI will execute into an executable model expressed as a Colored Petri Net (CPN). Each SOSI defines multiple SOSICs. This chapter describes the details of step 4 and step 5 of the assessment methodology described in section 6.4. Levis and Wagenhals [2000], Calderon [2005], and Pettit [2003] all offer processes to produce a CPN from architecture information. Levis and Wagenhals' transformation is not automatic, but it does offer a process to transform an architecture into an executable form in a traceable manner. Pettit and Calderon create automatic transformations to a Petri Net. The methodology provides an automated transformation from the UML Activity Diagrams to a CPN. The first part provides an overview of the methodology for transforming the Activity Diagram into a CPN. The second part describes the transformation of an Activity Diagram. This transformation into the CPN represents Step 4 of the primary methodology. The last part is Step 5 of the primary methodology and describes the unique modeling artifacts that are required to complete the executable model.

Transformation Process

The steps in the transformation process are similar to the MDA process depicted in Figure 6.41. An idealized MDA process begins with a Platform Independent Model (PIM). The first transformation creates a PSM from the PIM. The second transformation creates an executable form from the PSM. The process developed for this methodology represents the PIM as the static representation of the SOSI in UML. The first transformation creates a PSM that represents an instance of a CPN data model. The second transformation creates the xml file that is executed by the CPNTools.

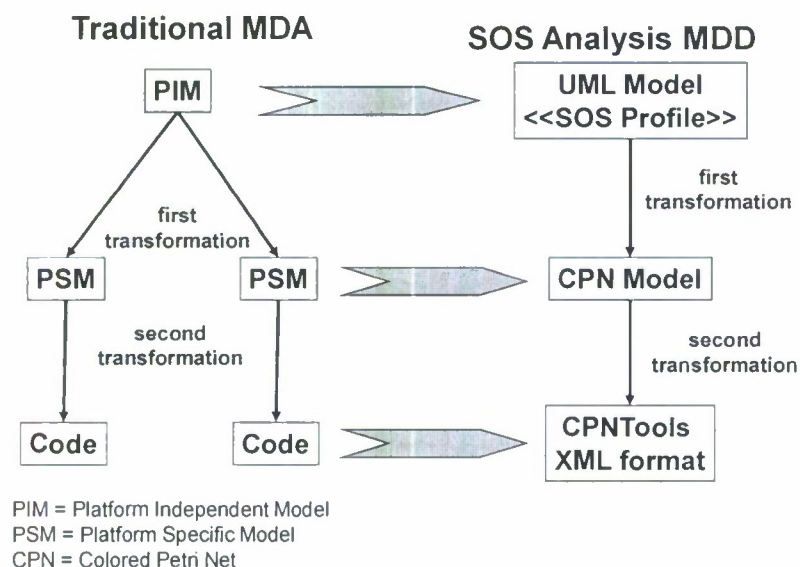


Figure 6.41: SOS Model Driven Development Process

The transformation process has four steps that follow closely the MDD environment described in Figure 6.41. These steps are subtasks of step 4.

Step 4. Transform SOSI alternatives into CPN Model.

Step 4.1. Transform automatically the UML to the CPN data model.

Step 4.2. Transform Automatically the CPN data model to CPN XML format.

Step 5. Configure and execute the CPN model.

Figure 6.42 shows the process with some the steps described in section 6.4 grayed out. Step 3 produces the Platform Independent Model (PIM) it is the static representation of the SOSI. Step 4.1 is the first transformation from the Activity Diagrams that represent the SOSICs to a PSM representing an instance of the CPN data model. Step 4.2 represents the second transformation and is the transformation from the CPN data model to the CPNtools XML format. Step 5 instruments the CPN for data gathering based on the way the Elements have been arrayed in the Nodes. The next section explains the specific tasks accomplished in each step.

Step 3 builds the UML model and is the fundamental architecture development step. All aspects of the architecture must be addressed in order to ensure that executable model can execute. These aspects are detailed in Steps 1 through 3 of the methodology. The resulting static UML representation of the SOSI alternative is the PIM for the transformation.

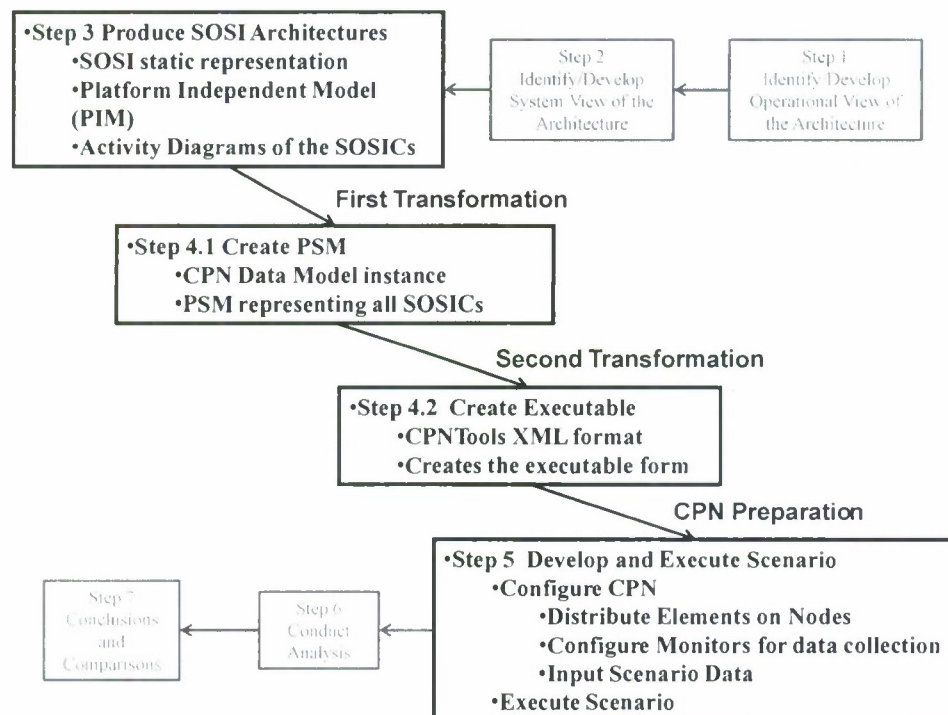


Figure 6.42. Transformation Process

The examples used in this chapter extend the example begun in section 6.4. Figure 6.43 shows the Activity Diagrams for the Element behavior. The Element Activity Diagrams are connected together to create the SOSIC Activity Diagram. Figure 6.44 shows the SOSICs that the Elements participate in. Notice that System2 is used in both SOSICs but different inputs are required. The unused inputs from each SOSIC are combined in the transformation to create a single executable that represents both activities executing simultaneously in one instance of System2. The next step addresses the first transformation in the MDD process and produces the Platform Specific Model.

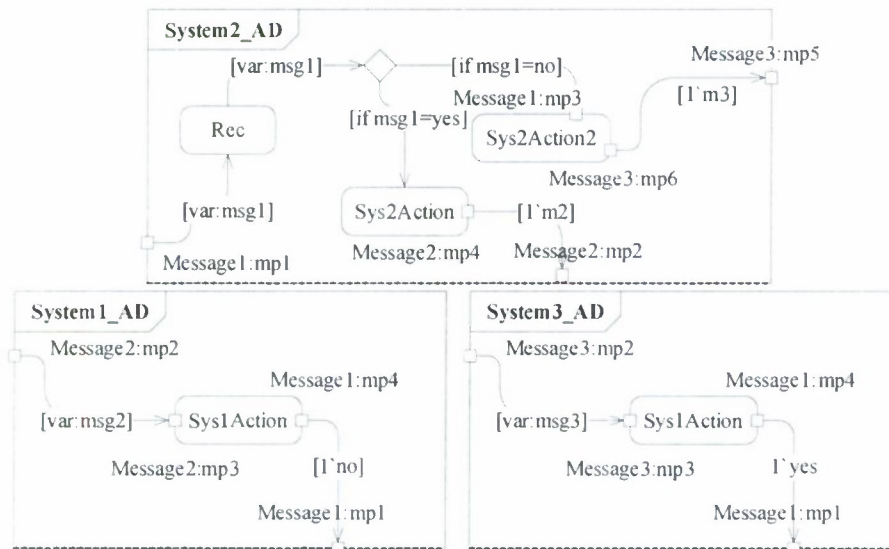


Figure 6.43. Element Activity Diagrams

Step 4.1 performs the transformation from the PIM to the PSM. The primary transformation is from the Activity Diagram to the CPN data model. Note that the transformation occurs with multiple SOSICs represented in multiple Activity Diagrams. The artifacts of the Activity Diagram are mapped to CPN constructs. The components described below are the ones that are used to model the behavior of the SOSI.

The transformation process translates the Activity Diagram components into CPN components. The basic components of the Activity Diagram used by this methodology are: Action () , Object Node () , Call Behavior Action () , Fork/Join Node () , Decision Node () , Initial Node () , Final Node () , Stop Node () , and Activity Parameter Node (). The components of the CPN are: Transition () , Place () , and Substitution Transition ().

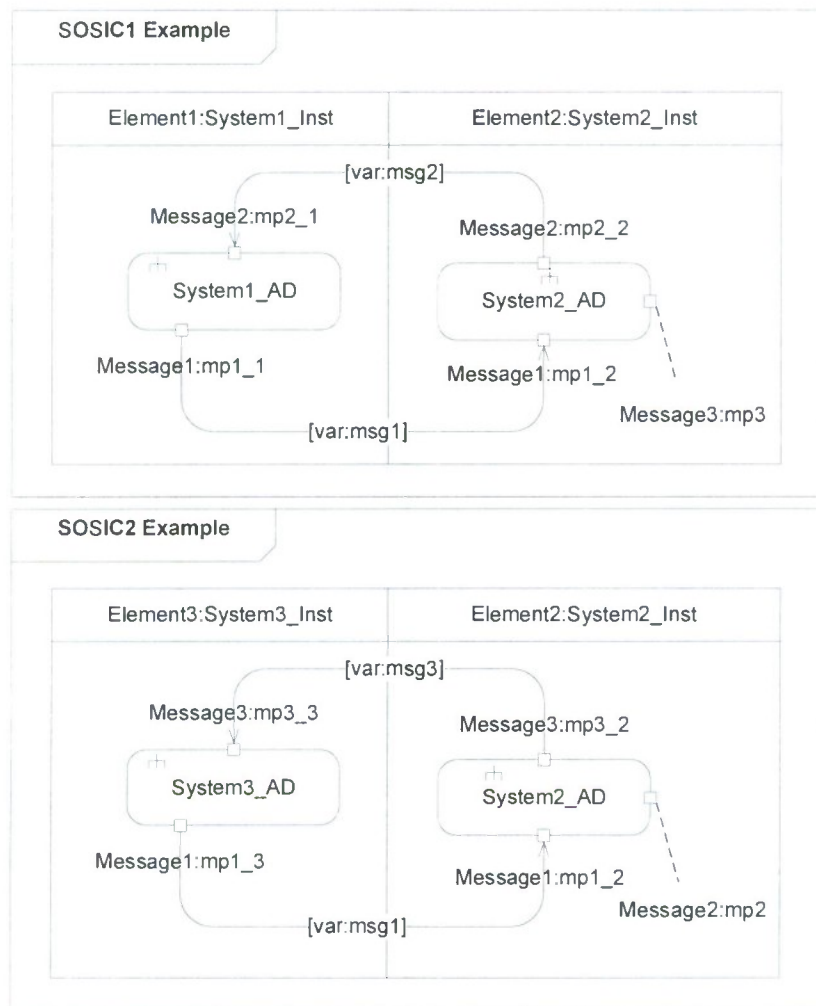


Figure 6.44 Element SOSIC Participation

An Action describes a fundamental unit of executable functionality. It represents some processing in the modeled system. [OMG, 2007] Actions are translated into Transitions in the CPN. Transitions represent actions that take specific input from places and produce specific output to places. In many cases, the Element functions defined in the SV-4 will be represented by groups of actions that accomplish a specific task.

CallBehaviorAction is an action used by the methodology to create a hierarchy of Activity Diagrams. (This action can call other behavior representations not used in this methodology like a state transition diagram or a sequence diagram). The CallBehaviorAction invokes lower level Activity Diagrams described for each Element. This Activity Diagram component translates to a Substitution Transition in the CPN which is a transition that is associated with another page in the CPN.

Object Nodes assist in describing the data that passes from Action to Action. [OMG, 2007] Four types of Object Nodes are used: Activity Parameter Node, Input/Output Ports, Data Stores, and Buffers. All of these Activity Nodes are transformed into CPN Places.

Four types of Control Nodes are used: Fork, Join, Decision, and Merge. All the Control Nodes translate into CPN transitions. Decision Nodes require accompanying arcs inscriptions that control the passing of tokens based on the value of variables represented in the token. These inscriptions are translated from the associated guards inscribed on the Decision Node output arcs represented in the Activity Diagram.

Finally, the terminal Nodes: Initial and Final, represent the beginning and end of each Activity. They are transformed into places in the CPN.

Figure 6.45 summarizes the above discussion. The Activity diagram components are shown in the top row and first column of Figure 6.45. The second row and column show the components of the CPN as well as their transformational relationship with the Activity Diagram components in the first row/column. The interior cells describe the CPN components that are used to connect the Elements described in the outside rows and columns.

source \ target		action			TYPE		call behavior		
		trans	trans	trans			subst		
action	trans						port socket		
	trans						port socket		
	trans						port socket		
TYPE					forbidden	forbidden		aux	aux
					aux	forbidden		forbidden	forbidden
call behavior	subst	port socket	port socket	port socket			port socket	forbidden	port
		forbidden	forbidden	forbidden	forbidden	forbidden	forbidden	forbidden	forbidden
					aux	forbidden	port	forbidden	aux

Figure 6.45. Activity Diagram Transformation Rules

The transformation described requires a data model for the CPN so that an Activity Diagram can be transformed into the CPN construct. The data model in Figure 6.46 is the template for the CPN PSM. All the transformation constructs described above are represented. The concept of a page, PNPage, is utilized as well as that of the Substitution Transition that facilitates creating the hierarchical representation of an Activity Diagram enabled by the CallBehaviorAction. The Place and Transition Nodes are also represented.

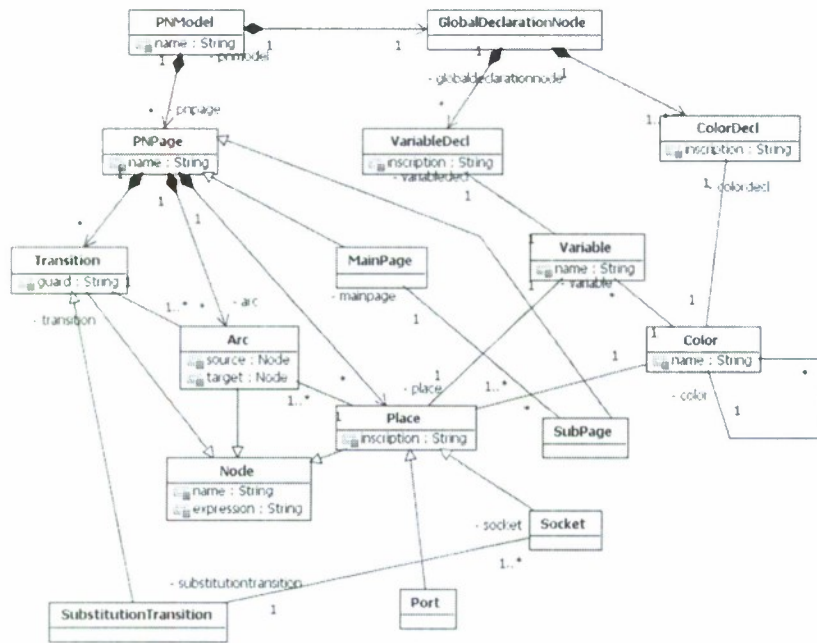


Figure 6.46. CPN Data Model

UML to CPN transformation

Figure 6.47 illustrates the concrete syntax transformation using a fragment of an Activity Diagram and the transformation rules shown in Figure 6.45. In the upper left is a fragment of an Activity Diagram for transformation. Step a) identifies the ends of the connecting arcs as actions. Actions are located in the first cell of the outside row and column of Figure 6.45. Step b) shows the transformation of the actions into transitions (as is shown in the second row and column) and a connecting place between them to hold the tokens generated by the actions. The connecting arcs and place are found at the intersection of the source and target in the interior of the matrix. Step c) creates the CPN data representation with the actions named in the transitions and the places used to hold the tokens generated by the actions. Notice that the type of the place is the type of the input and output ports on the actions. Next the transformation is applied to a simple example extended from section 6.4.

The first transformation creates the PSM from the SOSIC represented as an Activity Diagram. Figure 6.48 shows an excerpt of the SOSIC Example from Figures 6.43 and 6.44 transformed into an instance of the CPN data model representation, Figure 6.46. The PSM is an intermediate representation before finally creating the executable. The figure is a UML Object Diagram and represents an instance of the CPN data model. The boxes represent instances of classes. The transitions and places have been transformed using the rules represented in Figure

6.45. The diagram shows an instance of PNPage called Example. This represents the top level of the CPN. The Elements modeled in the SOSICs are represented as SubstitutionTransitions that compose the PNPage. The SubstitutionTransition objects are connected by Arc objects to Port objects that represent the interfaces into the subpage CPNs represented by the SubstitutionTransition objects. The other PNPage objects represent instances of System1 and System 2. These PNPage objects are composed of Transition objects that represent the Actions modeled in the Activity Diagrams for System1 and System2.

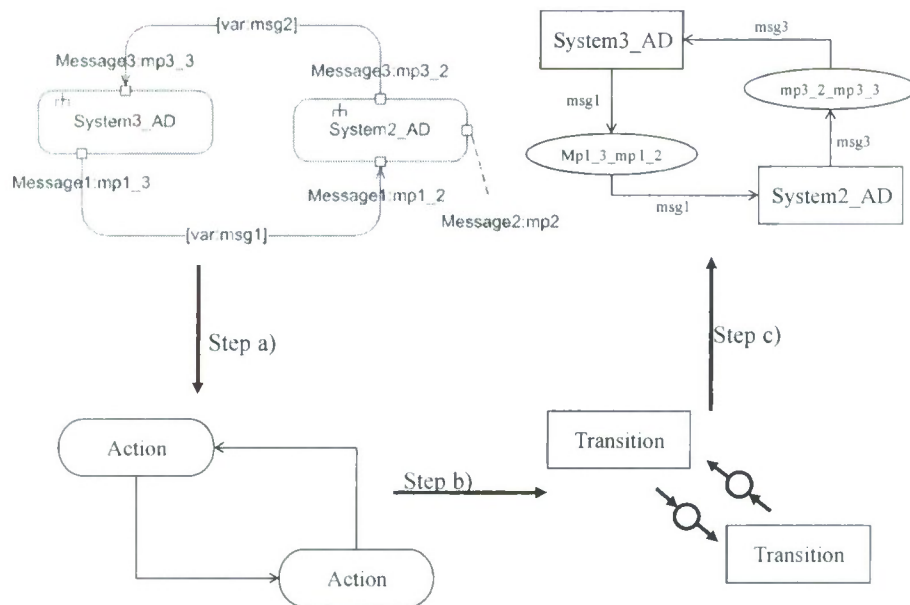


Figure 6.47. Transformation Process

Step 4.2 is the transformation of the CPN data model to the CPNtools xml representation. Figure 6.49 is the top-level CPN page created from the example Activity Diagrams. The boxes represent substitution transitions that represent the instances of the Elements: System1, System2 and System. The ovals are places that model the interfaces into the subpages that represent the Activity Diagrams of the Elements. Figure 6.50 are the subpages from the example. Using the connection between System3 and System2 as an example, the place P31 in Figure 6.49 is represented in Figure 6.50 in the CPN pages that represent System2 and System3. Place P31 is an input place for System2 and an output place for System3. The toplevel CPN page, Figure 6.49, shows this relationship between System2 and System3.

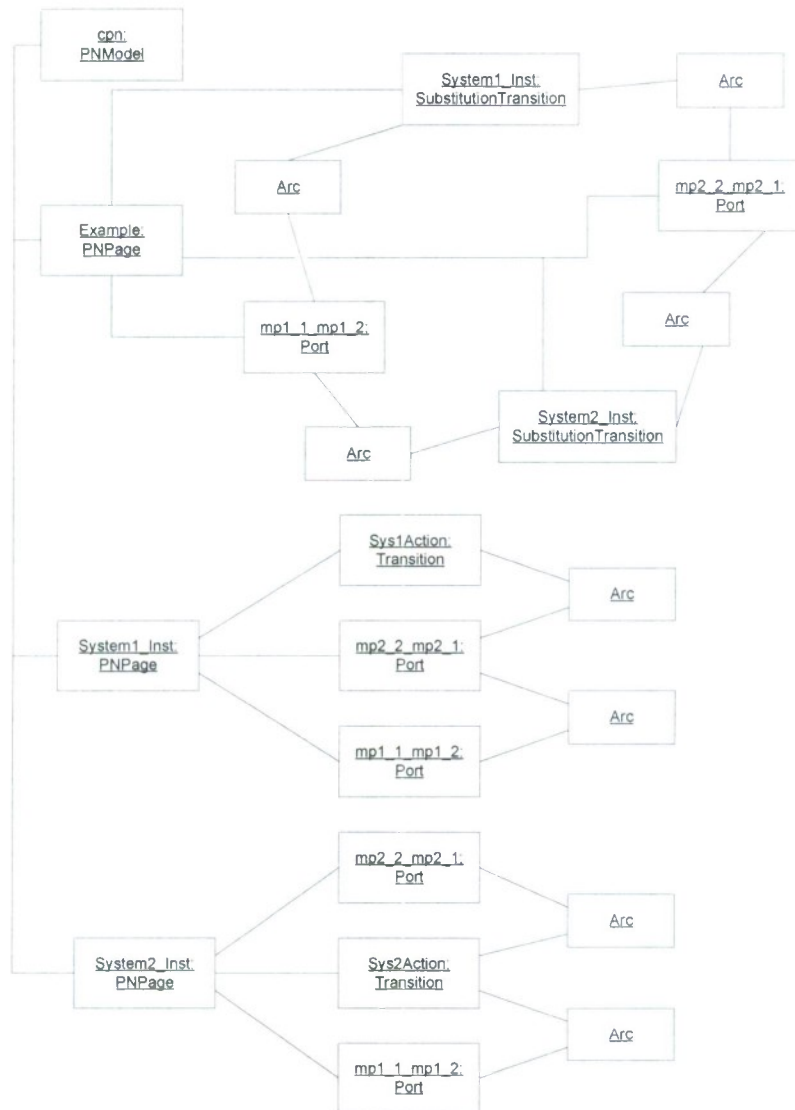


Figure 6.48. Example CPN Data Model

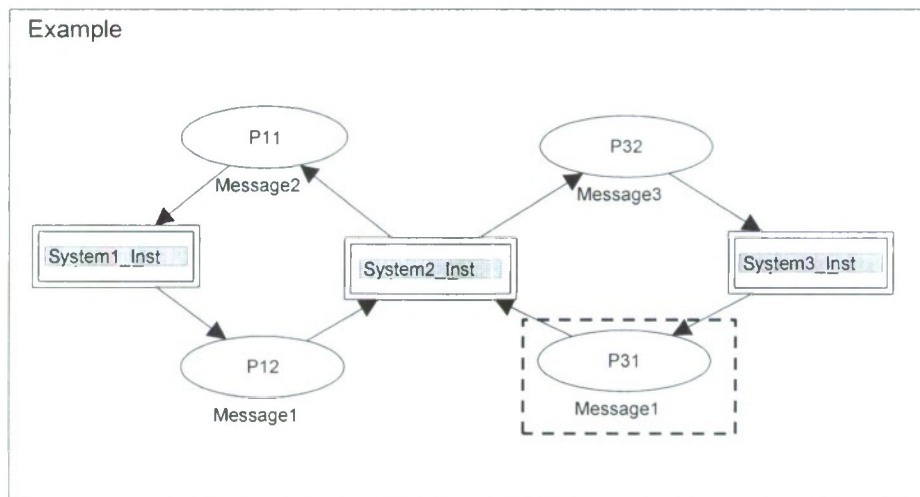


Figure 6.49. Top-level CPN Representation

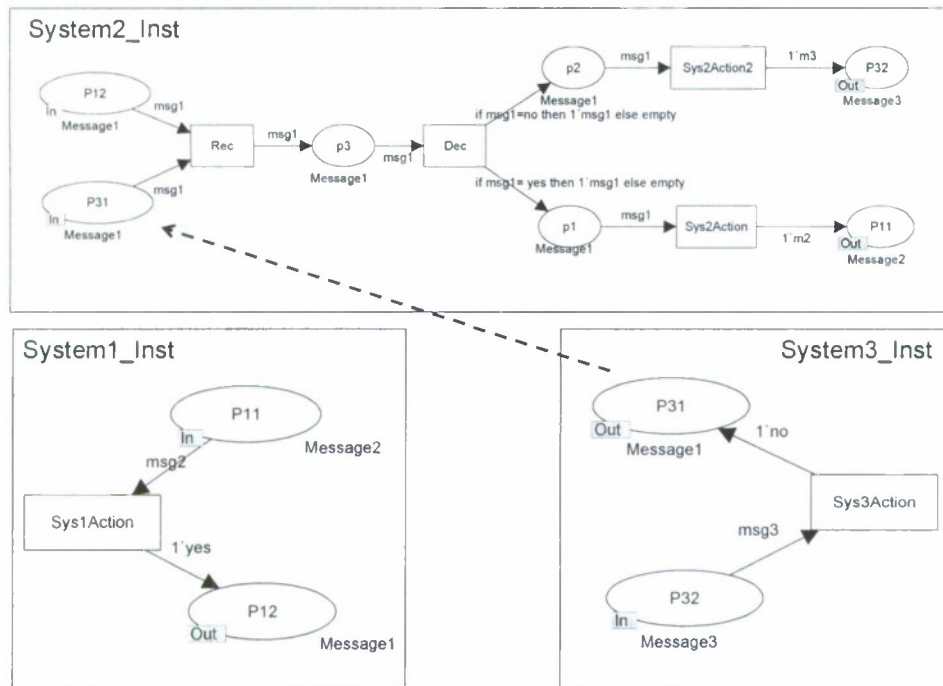


Figure 6.50. Sub-pages for Example CPN

Step 5 completes the executable model by adding monitors to count the messages passing between Nodes, and grouping the Elements into the particular Node configuration required by the SOSI alternative. CPNtools produces a report that shows the data collected by the monitors instrumenting the CPN. This report is analyzed and used to create the coupling assessments for the SOSI.

Summary

This section outlined the transformation of the UML Model into a CPN. The transformation uses a process similar to the MDA to accomplish the transformation. The example shows the ability of the transformation to create the representative CPN model given the UML Activity diagrams. The resulting CPN is a combined model of all the Activity diagrams associated with the SOSI. The next chapter completes the discussion of the SOSI measures with a case study that illustrates the use of the Adaptability and Agility measures

6.6 Conclusion

Adaptability and Agility provide a qualitative assessment of the interaction of the Elements and Nodes of the SOSI very early in the development process. This allows SOS engineers to assess the ability of the architecture to adapt to the deployed environment given that it is highly unlikely that the deployed environment will duplicate the scenarios used to test the SOSI. The combined executable model enables an analysis of the internal interaction of the Elements on a Node (Cohesion) and the degree of dependence of the Node to the rest of the SOSI (Coupling).

Degree of Reuse reveals Elements that might be over-utilized and thus inhibit the ability of the SOSI to provide the required capabilities concurrently.

The SOS architecture development methodology produces an executable model with behavior that is traceable to the static representation. The methodology ensures the rule, data and dynamic behavior models are accurately represented in the executable. Furthermore, the executable is a representation of the concurrently executing SOSICs derived from multiple system views of the architecture. The Coupling measure reveal the influence of the combined rule models on SOSI Adaptability by simulating the behavior of the Elements in concurrently executing SOSICs.

The ambiguity caused by changing adversaries, technological advancements and changing organizational structures will cause a significant amount of uncertainty as to what will be the deployed structure of the organization. The SOS engineering challenge is to assess the ability of alternative architectures to adapt to the operating environment in which it is deployed in order to provide a SOS that facilitates the level of Agility required by the organization.

This research contributes significantly in several areas. First the assessment measures Coupling, Cohesion and Degree of Reuse assess two aggregate performance characteristics of the SOSI, Adaptability and Agility. Adaptability describes the ability of a SOSI to respond to changes in the operating environment. Adaptability is modeled as a product of Cohesion and Coupling using the Cobb-Douglas [1920] form. The methodology distinguishes four cases of SOSI Adaptability:

Low Cohesion and Low Coupling = High Adaptability

Low Cohesion and High Coupling = Medium Adaptability

High Cohesion and Low Coupling = Medium Adaptability

High Cohesion and High Coupling = Low Adaptability

The case study reinforces the findings. The P2P and SOA alternatives have similar assessments for Adaptability, but the values of Coupling and Cohesion are different. The measures Coupling and Cohesion allow developers to identify traits that can be modified to improve the Adaptability of the SOSI.

Degree of Reuse and Exclusiveness assess the ability of the SOSI to execute the SOSICs concurrently. The case study illustrated the importance of the highly reused Elements and how reducing the overall reuse of Elements can improve the Exclusiveness and reduce the potential for contention for Element resources.

Adaptability and Exclusiveness combine to assess the overall Agility of a SOSI. This last measure provides an aggregate measure for assessing the ability of the SOSI to provide SOSICs concurrently and adapt to unpredicted operating environments.

Second, the methodology for combining multiple behavior models into a single combined executable significantly contributes to SOS engineer's ability to ensure the acceptability of the architectures and the ability to analyze the performance of the architectures in various scenarios. Structural architects depend on 3D representations in paper or computer generated to obtain feedback from the stakeholder about whether the proposed solution meets the needs of the organization. SOS engineers must rely on the executable model to provide a representation that allows the stakeholder to observe modeled performance and assess the appropriateness of the architecture. The model driven development environment adds validity to the process by ensuring the executable behavior is directly traceable to model artifacts in the architecture. The environment created for this methodology creates such an environment.

Third, SOS assessment requires that the SOSI be bounded for analysis and the structural and behavioral aspects of the architecture are modeled accurately. The SOS taxonomy developed for the methodology provides such a description. The SOSI is a bounded subset of the resources available to the organization. The Nodes provides structure for the SOSI, while the SOSICs model the processes that realize the capabilities for the organization.

Furthermore, the methodology provides the required architecture data early in the development process to improve early decisions concerning technologies and architecture design tradeoffs.

Finally, the Cobb-Douglass production function is used in a unique manner to relate Coupling and Cohesion for the computation of Adaptability.

There are many aspects of adaptability and agility that could be measured from the information provided by the methodology. This research concentrated on structural changes denoted by changing the configuration of the SOSI Nodes and holding all other aspects of the SOSI constant. Further work could be conducted measuring the ability of the SOSI to adapt to new SOSIC processes given a fixed set of Elements. The analysis might include performance analysis or a gap analysis that reveals shortcomings in the ability of the SOSI to provide a particular capability because a particular system function is not available in the current set of Elements. Another analysis might include the ability of the SOSI to operate in a degraded mode because certain Elements have been compromised in some fashion. Finally, there is a security aspect that should be considered to ensure that the SOSI Architecture implements the required security capabilities to ensure an uncompromised operating environment.

Another area of future work is further analysis of the ability of the SOSI to provide the operational capability described in the operational architecture view. Developers need the ability to ensure the operational concept described in the operational architecture view is actually met by the SOSI Architecture for a particular SOSI. The SV-5 assists in this arena, but only addresses the obvious modeled functions. Research in this area could reveal contradictions in the state space of the SOSI Architecture when compared to the state space of the operational architecture views. This could be true for a single capability or true when multiple capabilities are being provided.

Finally, more work is required to ensure UML semantics defined in the UML specification are formally defined. This work revealed semantics for the Activity Diagram that are not supported in the SOS environment. Such work might entail modification of current UML profiles or a new UML profile that supports the development of SOSI Architectures. Improved semantics would also assist in analyzing the UML model directly. In order for the UML to support graph analysis like invariant and state space analysis, the semantics of the language must be constrained to that reduce the ambiguity currently in the UML specification.

6. 7 References

- Alberts, D.S. and Hayes, R.E. (2003), *Power to the Edge: Command and Control in the Information Age*, 2003, Washington, D.C.: Command and Control Research Program.
- Alberts, D.S. and Hayes, R.E. (2007), *Planning: Complex Endeavors*, 2007, Washington, D.C.: Department of Defense Command and Control Research Program.
- Arisholm, E., *Dynamic Coupling Measures for Object-Oriented Software*, in *Proceedings of the 8th International Symposium on Software Metrics*. 2002, IEEE Computer Society.
- Arisholm, E., Briand, L.C. and Foyen, A., *Dynamic coupling measurement for object-oriented software*. *Transactions on Software Engineering*, 2004. 30(8): p. 491-506.
- Baldassari, M., Bruno, G. and Castella, A. (2001), *RPTOB: an Object-oriented CASE Tool for Modeling and Prototyping Distributed Systems*. *Software-Practice and Experience*, 1991. 21(8): p. 823-844.
- Baresi, L. and Pezzè, M. (2001) *On Formalizing UML with High-Level Petri Nets*. *Lecture Notes in Computer Science*, ed. Agha, G.A., Cindio, F.D. and Rozenperg, G. eds. 2001, Springer-Verlag: Berlin.
- Bernardi, S., Donatelli, S. and Merseguer, J.E. (2002). *From UML Sequence Diagrams and Statecharts to Analysable Petri Net Models*. in *3rd International Workshop on Software and Performance 2002*. Rome, Italy: ACM Press.
- Beydeda, S., Book, M. and Gruhn, V. (2005), *Model Driven Software Development*. 2005, Berlin, Germany: Springer-Verlag.
- Bieman, J. and Kang, B.K. (1998) *Measuring Design-Level Cohesion*. *IEEE Transactions on Software Engineering*, 1998. 24(2).
- Bienvenu, M.P., Shin, I. and Levis, A.H. (2000), *C4ISR Architectures: III. An Object-Oriented Approach for Architecture Design*. *Systems Engineering*, 2000. 3(4).
- Breton, E. and Bézivin, J. (2001). *Towards an Understanding of Model Executability*. in *FOIS*. 2001. Ogunquit, ME: ACM.
- Briand, L.C., Morasca, S. and Basili, V.R. (1999), *Defining and Validating Measures for Object-Based High-Level Design*. *IEEE Transactions on Software Engineering*, 1999. 25(5).
- Brown, M. (2005), *Personal Interview*. Sep 2005: Fort Belvoir.
- Brownsword, L.L. (2004), et al., *Current Perspectives on Interoperability*. *SEI Technical Note*, 2004. CMU/SEI-2004-TR-009.

- Buede, D. (2000), *The Engineering Design of Systems: Models and Methods*. 2000, New York: John Wiley and Sons, Inc.
- Calderon, M. (2005), *Model Transformation Support for the Analysis of Large-Scale Systems*, in *Software Engineering*. 2005, Texas Tech University.
- Carney, D., et al (2005) , *Some Current Approaches to Interoperability*. CMU, 2005 (CMU/SEI-2005-TN-033).
- Carney, D., Fisher, D. and Place, P. (2005), *Topics in Interoperability: System-of-Systems Evolution*. SEI Technical Note, 2005. CMU/SEI-2005-TN-002.
- Chidamber, S.R. and Kemerer, C.F. (1994), *A Metrics Suite for Object-Oriented Design*. IEEE Transactions on Software Engineering, 1994. 20(6): p. 476-493.
- Cobb, C.W. and Douglas, P.H. (1928), *A Theory of Production*. The American Economic Review, 1928. 18(1): p. 139-165.
- Cook, S.C. (2001) *On the Acquisition of Systems of Systems*. in 2001 INCOSE International Symposium. 2001. Melbourne AU.
- CPN_Group, CPN Tools 2.2.0. 2008, University of Aarhus: Denmark., <http://wiki.daimi.au.dk/cpntools/cpntools.wiki>
- DAU (2006), *Defense Acquisition University, Defense Acquisition Guidebook*, Department of Defense, Editor. 2006, DAU.
- DeLaurentis, D.A. (2005), "A Taxonomy-based Perspective for Systems of Systems Design Methods," *Systems, Man and Cybernetics*, 2005 IEEE International Conference on , vol.1, no., pp. 86-91 Vol. 1, 10-12 Oct. 2005
- Dickerson, C.E., et al.(2004), *Using Architectures for Research, Development, and Acquisition*. O.O.T.A.S.O.T.N.R.D.A. ACQUISITION), Editor. 2004, Defense Technical Information Center.
- DODAF(2007a), *Department of Defense Architecture Framework Version 1.5 Vol 1*, 2007. Defense Information Security Agency.
- DODAF(2007a), *Department of Defense Architecture Framework Version 1.5 Vol 2*, 2007. Defense Information Security Agency.
- DODAF(2007a), *Department of Defense Architecture Framework Version 1.5 Vol 3*, 2007. Defense Information Security Agency.
- Eshuis, R. and Wieringa, R. (2001), *A Comparison of Petri Net and Activity Diagram Variants*. in 2nd Int. Collaboration on Petri Net Technologies for Modelling Communication Based Systems. 2001: DFG Research Group "Petri Net Technology".
- Eshuis, R. and Wieringa, R. (2001), *A Real-Time Execution Semantics for UML Activity Diagrams*, in *Fundamental Approaches to Software Engineering : 4th International Conference, FASE 2001 : Held as Part of the Joint European Conferences on Theory and Practice of Software, ETAPS 2001, Genova, Italy, April 2-6, 2001, Proceedings*. 2001. p. 76.
- Farkas, J. (1902), *Theorie der einfachen Ungleichungen*. Journal Fur reine und angew, 1902. Mathematik(124).

- Fowler, M.(2003), Who Needs an Architect? IEEE Software, vol. 20, no. 5, pp. 11-13, Sept/Oct.
- Fowler, M.(2004), UML Distilled, Object Technology Series, ed. Addison-Wesley. 2004: Pearson Education, Inc.
- Genrich, H. and Lautenbach, K. (1979), The Analysis of Distributed Systems by Means of Predicate/Transition-nets, in Semantics of Concurrent Computation. 1979. p. 123-146.
- Grady, J.O.(1994), System Integration. Systems Engineering Series. 1994, Boca Raton: CRC Press. 256.
- Greenwood, R.M., et al.(1995), Active Models In Business. in Business IT Conference. 1995. Manchester.
- Hansen, K.M.(2001), Towards a Coloured Petri Net Profile for the Unified Modeling Language - Issues, Definition, and Implementation. 2001: University of Aarhus.
- He, X. and Ding, Y. (2001), Object Orientation in Hierarchical Predicate Transition Nets. LNCS, 2001. 2001(2001).
- Hillion, H.P.(1986), Performance Evaluation of Decisionmaking Organizations, in Laboratory for Information and Decision Systems. 1986, MIT: Cambridge.
- Hong, J.-E. and Bae, D.-H. (2000), Software Modeling and Analysis Using a Hierarchical Object-oriented Petri Net. Information Sciences, 2000. 130(1-4): p. 133-164.
- Hassoun, Y., Counsell, S. and Johnson, R. (2005), Dynamic Coupling Metric: Proof of Concept. IEE Proceedingd in Software Engineering, 2005. 152(6).
- IEEE (2000), Recommended Practice for Architectural Description of Software-Intensive Systems. 2000, IEEE.
- JCS(2007), Joint Chiefs of Staff, Dictionary of Military and Associated Terms, Department of Defense, Editor. 2007, Joint Chiefs of Staff.
- Jensen, K. (1991), High-Level Petri nets: Theori and Application. 1991, Berlin, Gernany: Springer-Verlag.
- Jensen, K.(1992), Coloured Petri Nets: Basic Concepts, Analysis Methods and Practical Use. Monographs on Theoretical Computer Science, ed. W. Brauer, G. Rozenberg, and A. Salomaa. Vol. 1. 1992, New York: Springer-Verlag.
- Jensen, K.(1993), Coloured Petri Nets. Discrete Event Systems: A New Challenge for Intelligent Control Systems, IEE Colloquium on,, 1993(4): p. 1-3.
- Jensen, K.(1995), Coloured Petri Nets: Basic Concepts, Analysis Methods and Practical Use. Monographs on Theoretical Computer Science, ed. W. Brauer, G. Rozenberg, and A. Salomaa. Vol. 2. 1995, New York: Springer-Verlag.
- Jensen, K.(1997), Coloured Petri Nets: Basic Concepts, Analysis Methods and Practical Use. Monographs on Theoretical Computer Science, ed. W. Brauer, G. Rozenberg, and A. Salomaa. Vol. 3. 1997, New York: Springer-Verlag.
- Kasunic, M. and Anderson, W. (2004), Measuring Systems Interoperability: Challenges and Opportunities. CMU/SEI-2004-TN-003, 2004.

- Kleppe, A., Warmer, J. and Bast, W. (2003), *MDA Explained The Model Driven Architecture: Practice and Promise*. Object Technology Series, ed. Addison-Wesley. 2003: Pearson Education Group, Inc.
- Lehman, M. M.(1996), *Laws of Software Evolution Revisited*. in EWSPT. 1996. Nacy, France: Springer-Verlag.
- Leite, M. (1998), *Interoperability Assessment*. in MORS Symposium. 1998. Monterey, CA.
- Levis, A.H. and Wagenhals, L.W. (2000), *C4ISR Architectures: 1. Developing a Process for C4ISR Architecture Design*. Systems Engineering, 2000. 3(4).
- López-Grao, J.P., Merseguer, J. and Campos, J. (2004), *From UML Activity Diagrams to Stochastic Petri nets: Application to Software Performance Engineering*. SIGSOFT Softw. Eng. Notes, 2004. 29(1): p. 25-36.
- Maier, M.W.(1996), "Architecting Principles for Systems-of-Systems". in 6th Annual International Council on System Engineering (INCOSE) Symposium Proceedings. 1996.
- Maier, M.W. (1998), *Architecting principles for systems-of-systems*. Systems Engineering, 1998. 1(4): p. 267-284.
- Maqbool, S. (2005), *Transformation of the Core Scenario Model and Activity Diagrams into Petri Nets*, in Computer Science. 2005, University of Ontario: Ottawa. p. 172.
- Merseguer, J., et al. (2002), *A Compositional Semantics for UML State Machines Aimed at Performance Evaluation*. 2002.
- NIST (1993),National Institute for Standards and Technology, *Integration Definition For Function Modeling (IDEF0)*, FIPS 183, Federal Information Processing Standards Publications.
- Novak, J.D. and Cañas, A.J. (2006) *The Theory Underlying Concept Maps and How to Construct Them.*, Technical Report IHMC CMapTools 2006-01, 2006; Available from: <http://cmap.ihmc.us/Publications/ResearchPapers/TheoryUnderlyingConceptMaps.pdf>.
- OMG.(2006a), Object Management Group, *UML Profile for DODAF and MODAF Request for Proposal*, 2006 [cited. 1 May 2007] <http://www.omg.org>
- OMG.(2006b), Object Management Group, *System Modeling Language version 1.0*, 2006 [cited. 1 May 2007] , <http://www.omg.org>
- OMG.(2006c), Object Management Group, *Meta Object Facility Core Specification Version 2.0*, 2006 [cited. 1 May 2007] <http://www.omg.org>
- OMG.(2005),Object Management Group, *XML Metadata Interchange Version 2.1*, 2005 [cited. 1 May 2007] <http://www.omg.org>
- OMG.(2007a), Object Management Group, *Unified Modeling Language: Infrastructure*. 2007 [cited 2008 Feb 2008]. <http://www.omg.org>
- OMG.(2007b), Object Management Group, *Unified Modeling Language: Superstructure*. 2005 [cited. 1 May 2007] <http://www.omg.org>
- Peterson, J.(1981), *Petri Net Theory and the Modeling of Systems*. 1981, Upper Saddle River, NJ: Prentice Hall.

- Petri, C.A.(1966), Communcation with Automata. Defense Technical Information Center, 1966.
- Petriu, D.C. and Shen, H. (2002), Applying the UML Performance Profile: Graph Grammar-Based Derivation of LQN Models from UML Specifications, in Computer Performance Evaluation : Modelling Techniques and Tools 12th International Conference, TOOLS 2002 London, UK, April 14-17, 2002. Proceedings. 2002. p. 159.
- Pettit, R.(2003), Analyzing dynamic Behavior of concurrent Object-Oriented Software Designs, in Information and Software Systems Eengineering. 2003, George Mason university: Fairfax, VA.
- Pooley, R. and King, P. (1999), Using UML to derive stochastic Petri net models. Petri Nets Performance Models '99, 1999.
- Rechtin, E.(1991), Systems architecting: Creating & building complex systems, Prentice Hall, Englewood Cliffs, NJ, 1991.
- Rechtin, E.(1992), The Art of Systems Architecting, IEEE Spectrum (October 1992), pp. 66–69.
- Rechtin, E. and Maier M.(1996), The Art of Systems Architecting, CRC Press, Boca Raton, FL, 1996.
- Sage, A.P. and Cuppan, C. D. (2001), On the Systems Engineering and Management of Systems of Systems and Federations of Systems,. Information, Knowledge, and Systems Management, 2001. 2(4).
- Saldhana, J. and Shatz, S. (2000), UML Diagrams to Object Petri Net Models: An Approach for Modeling and Analysis. proceedings of the International Conference on Software Engineering and Knowledge Engineering(SEKE), 2000: p. 103-110.
- Selic, B., Gullekson, G. and Ward, P. (1994), Real-Time Object-Oriented Modeling. 1994: John Wiley and Sons.
- Sommerville, I. (2004), Software Engineering. 7 ed. 2004, Essex: Pearson Education Limited.
- Störrle, H.(2005), Towards a Petri-net Semantics of Data Flow in UML 2.0 Activities, in Technical Report 0504. 2005, Ludwig-Maximilians-Universität München, Institut für Informatik,.
- Sunyé, G., et al.(2001), Using UML Action Semantics for Executable Modeling and Beyond. Advanced Information Systems Engineering: 13th International Conference, CAiSE 2001, Interlaken, Switzerland, June 4-8, 2001, Proceedings, 2001. 2068: p. 433.
- TheOpenGroup (2004), The Open Group Architecture Framework, Version 8.1 'Enterprise Edition'. 2004.
- Tolk, A.(2003), Beyond Technical Interoperability – Introducing a Reference Model for Measures of Merit for Coalition Interoperability. in 8th CCRTS. 2003. Washington, DC: National Defense University.
- Várró, D. and Pararicza, A. (2003), UML Action Semanitics for Model Transformation Systems. Peridica Politehnica, 2003. 47(3): p. 167-186.
- Wagenhals, L.W., Haider, S. and Levis, A.H. (2003), Synthesizing Executable Models of Object Oriented Architectures. Systems Engineering, 2003. 6(4).

- Wagenhals, L.W., et al.(2000), C4ISR Architectures II: A Structured Analysis Approach for Architecture Design. Systems Engineering, 2000. 3(4).
- Whittle, J.(2002), Transformations and Software Modeling Languages: Automating Transformations in UML. LNCS, 2002. 2460.
- Yacoub, S.M., Ammar, H.H. and Robinson. T. (1999), Dynamic Metrics for Object Oriented Designs. in Software Metrics Symposium. 1999. Boca Raton, Florida.
- Yourdon, E. and Constantine, L., Structured Design. 1979, Englewood Cliffs, NJ: Prentice Hall.

SECTION 7

Case Study: Expeditionary Strike Group¹

Stewart W. Liles and Alexander H. Levis

7.1 Introduction

The case study is an idealized military example with diverse capabilities that must be executed concurrently in an unpredictable operating environment. The case study requires an assessment of multiple SOSI architectures in order to decide how to configure the organization for its upcoming deployment.

7.2 Scenario

The fictional mission in this case is as follows. On a small island in the Pacific called Efcratia the US maintains a ground station that receives data down-linked from national security assets. It also has had ready access to the port facilities. The population in Efcratia is diverse. The majority is Moslem but with a significant minority that is Christian (Catholic). The government and the population of Efcratia are generally pro-US, but there exists a small vocal opposition to US presence on the island. More recently, in response to world events, a local instantiation of a terrorist organization, the Shining Crescent, has established a presence on the island and is fomenting anti-US attitudes.

The recent earthquake and the resulting tsunami caused substantial damage to the infrastructure of the island and destroyed many of the government buildings in Efcratia's capital – the main port city. It has also caused damage to the airport so transport planes cannot land – only small planes. As a result of the tsunami and the destruction, there is anarchy on the island. Consequently, in addition to the dire need for humanitarian assistance and disaster relief, there is also need for rapid re-establishment of public order and for Efcratia's government to function and provide services.

The US Government, through the Pacific Command (PACOM), has decided to send an ESG that was in the area with two primary objectives: (a) provide some protection to the humanitarian assistance and disaster relief that is being sent to the island through the port city; and (b) protect the ground station from possible politically or financially motivated attack. The ESG X receives the orders while at sea on its way to the Southwest Asia area of operations.

We need to develop SOSI Architecture alternatives and present our assessment of the alternatives to the commander. Figure 7.1 shows the operational concept graphic. This graphic shows the island of Efcratia and identifies the various Nodes that will be used to structure the ESG for its operations. The Tarawa, Austin and Harper's Ferry ships represent command ships

¹ This section consists of the slightly edited Chapter 6 of the Ph.D. thesis of LCOL Stewart W. Liles, USA.

that can act as Nodes for the ESG. The Satellite Node is used to Link geographically separated Elements together with common communication facilities. The other Nodes represented are the Beach, Ground Station, and Port.

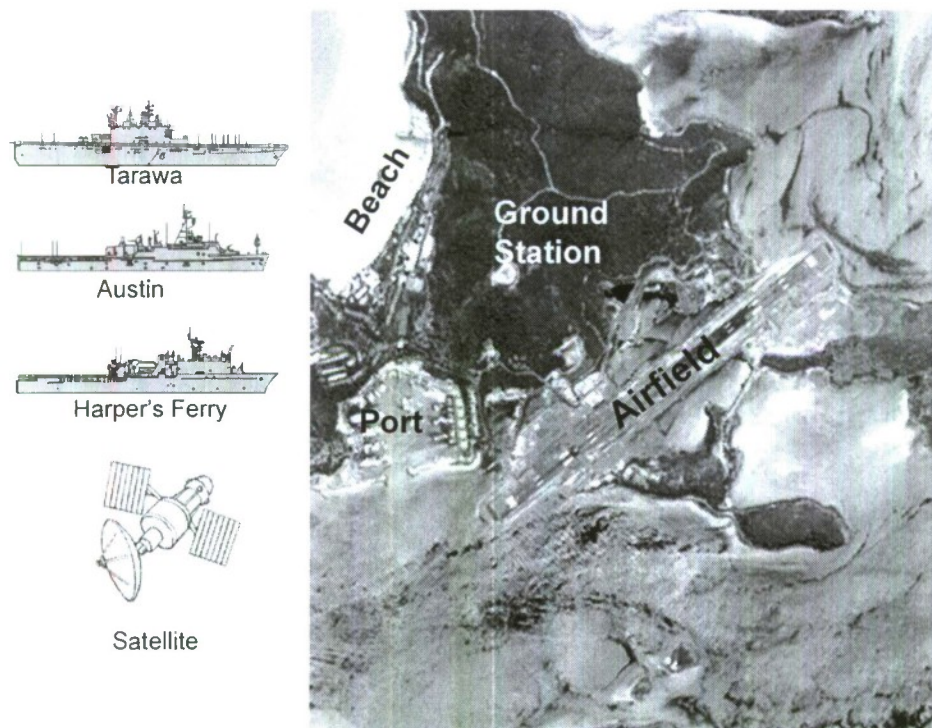


Figure 7.1. ESG Operational Concept Graphic (OV-1)

Given the mission scenario and the provided operational architectures that represent the required capabilities, assess the architecture alternatives for their ability to adapt to unplanned configurations. The resulting SOSI Architectures are built from the perspective of the lead System Engineer.

7.3 Operational Architectures

Step 1 of the methodology identifies the Operational Architectures that describe the required capabilities. This section illustrates the required capabilities in simplified operational architectures using DODAF products to describe the kind of data required. The assessment process requires four operational views for each capability: the Operational Node Connectivity Diagram, OV-2; the Operational Activity Model, OV-5; Operational Rules Model, OV-6a; and the Logical Data Model, OV-7.

The OV-2 shows the particular roles that are represented in the capability and the data that is passed between the roles. Figure 7.2 through 7.4 show the OV-2s for each capability. Figure 7.2 is the OV2 for the Planning and Coordination Capability. A Requestor initiates the Operational activity by sending a Request to a Coordinator. The Coordinator then sends an Order to the

Planners and the Planners respond with a revised Order. The Order is then sent to the appropriate Executor. The Executor coordinates with the Requestor and sends Status to the Coordinator.

Figure 7.3 is the Blue Force Tracking (BFT) Capability. The capability begins with the Reporter sending new BluePLI (Blue Position Location Information) to the Distributor. The Distributor then sends BluePLI Messages to all connected Receivers.

Figure 7.4 is the Process and Disseminate Intelligence Information Capability. This capability begins with personnel or equipment being sensed by a Sensor. Then a Sensor sends an Input to the Controller. Input types are Blip, Signal, and Sighting. The controller sends a SpotReport to the Analyzer. The Analyzer uses multiple SpotReports to synthesize opposing force locations. The Analyzer then passes the RedPLI (Red Position Location Information) to a Distributor. The Distributor distributes the RedPLI to Receivers.

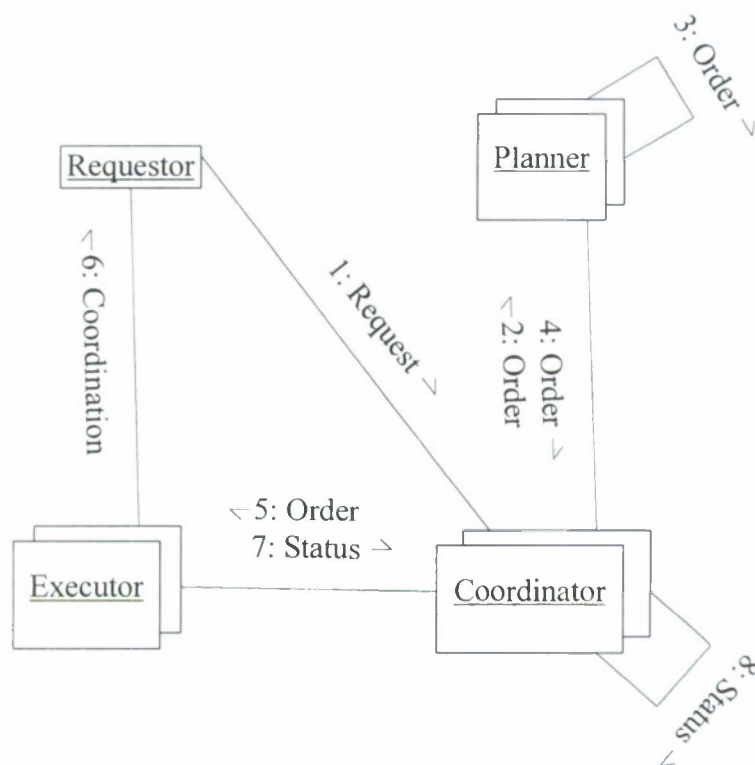


Figure 7.2. Planning and Coordination OV-2

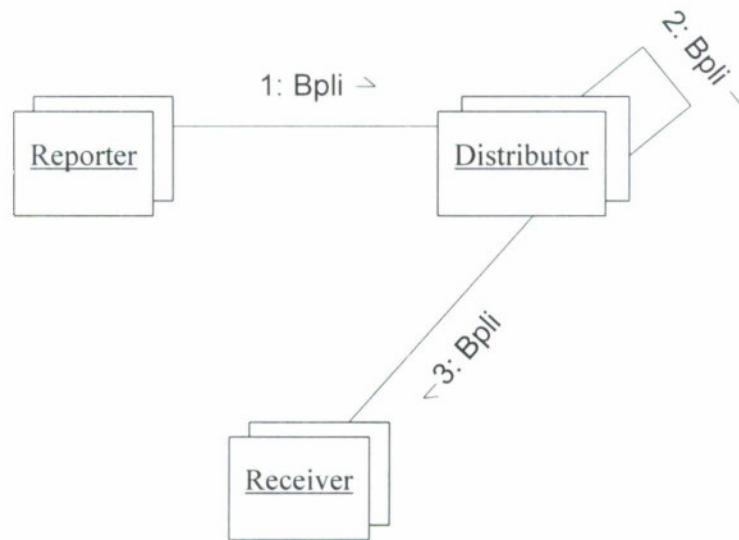


Figure7.3 Blue Force Tracking OV-2

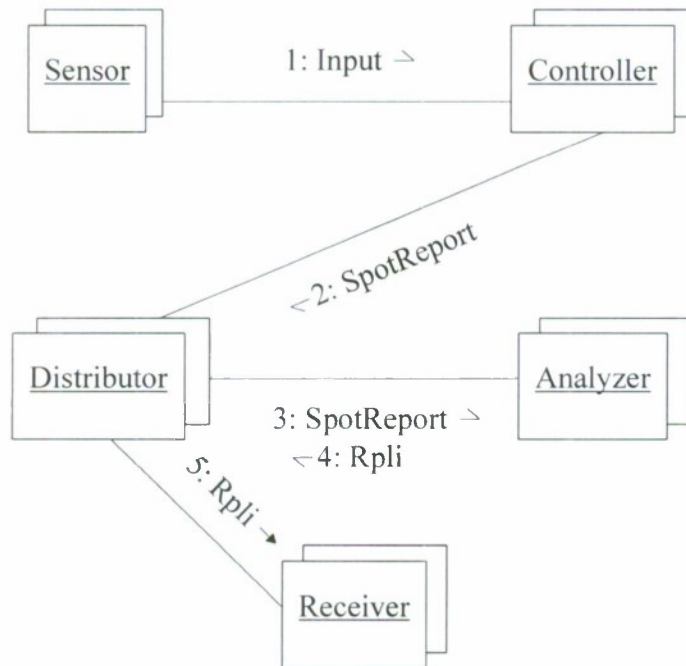


Figure 7.4. Process and Disseminate Intelligence Information OV-2

The Operational Information Exchange Matrix, OV-3, further defines the data exchanges identified in the OV-2 by describing the attributes of the messages passed between the represented roles.

Table 7.1 represents the OV-3s for the required capabilities. For example, a Request message is sent from the Requester to a Coordinator. A Request message is of type text and is 32 characters long.

Table 7.1. ESG Operational Information Exchange Matrix OV-3

Planning and Coordination Capability				
	Sender	Receiver	Type	Length
Request	Requester	Coordinator	Text	32
Order	Coordinator	Planner	Text	Variable
Order	Planner	Coordinator	Text	Variable
Status	Executor	Coordinator	Text	48
Coordination	Executor	Requestor	Text	32
Blue Force Tracking Capability				
	Sender	Receiver	Type	Length
BluePLI	Reporter	Distributor	Text	32
BluePLI	Distributor	Receiver	Text	32
Process and Disseminate Intelligence Information				
	Sender	Receiver	Type	Length
Input	Sensor	Controller	Text	48
SpotReport	Controller	Distributor	Text	48
SpotReport	Distributor	Analyzer	Text	48
RedPLI	Analyzer	Distributor	Text	48
RedPLI	Distributor	Receiver	Text	48

Operational activities describe the actions that each role executes to create the capability. The Operational Rules Models, OV-6a, Table 7.2, document the rules governing the behavior of operational activities. The rules are modeled in the Operational Activity Model, OV-5. For example the Coordinator role has an activity called ProcessRequest (Coordinator:ProcessRequest). This operational activity implements the rule, If Request = req_sec then Order = ord_sec, which means if a request for security is received send an Order for security.

Table 7.2. ESG Operational Rules Model OV-6a

Planning and Coordination Capability	
Requester:SendRequest	Send all generated Request
Requester:RecieveCoordination	Receive all Coordination messages
Coordinator:ReceiveRequest	Send all Request
Coordinator:ProcessRequest	If Request = req_sec then Order = ord_sec
Coordinator:ProcessRequest	If Request = req_sup then Order = ord_sup
Coordinator:SendOrder	Send all Order
Coordinator:DistributeOrder	Send all Order to connected Executor
Coordinator:ReceiveStatus	Receive and Store all Status reports
Planner:ReceiveOrder	Receive all Order
Planner:ProcessOrder	If Order = ord_pass then stop
Planner:ProcessOrder	If Order = ord_sec or ord_sup then SendOrder
Planner:SendOrder	Send all Order
Executor:ReceiveOrder	If Order then send Coordination
Executor:ComputeStatus	Send Order then send Status
Blue Force Tracking Capability	
Reporter:ComputeBpli	Compute location and sendBpli
Reporter:SendBpli	Send all BluePLI
Distributor:ReceiveBpli	If BluePLI $\diamond$ bpli_pass DistributeBpli
Distributor:DistributeBpli	Send all BluePLI
Receiver:ReceiveBpli	Receive all BluePLI
Receiver:ReceiveBpli	Store all BluePLI
Process and Disseminate Intelligence Information Capability	
Sensor:Sense	If Sense then SendInput
Sensor:SendInput	Send all Input
Controller:ProcessInput	If Input = input_pers then SpotReport = sr_pers

Controller:ProcessInput	If Input = input_equip then SpotReport = sr_equip
Controller:SendSpotReport	Send all SpotReport
Analyzer:ProcessSpotReport	If SpotReport = sr_pers then RedPLI = rpli_pers
Analyzer:ProcessSpotReport	If SpotReport = sr_equip then RedPLI = rpli_equip
Analyzer:SendRedPLI	Send all RedPLI
Distributor:DistributeRedPLI	If RedPLI =rpli_pass then empty
Receiver:RecieveRpli	Receive and Store all RedPLI

The OV-7 describes the messages passed between Elements and the operations expected of the roles defined in the OV-2. The OV-7s for each capability are shown in Figs. 7.5, 7.6 and 7.7. The body of each Message is represented as a String. The contents of the Message body are interpreted based on the type of Message received. Each role type is defined for each capability.

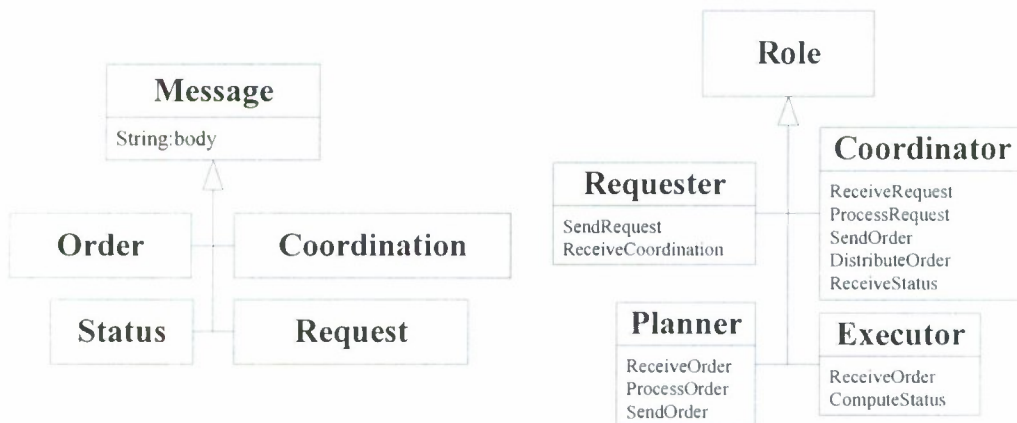


Figure 7.5. Planning and Coordination Capability OV-7

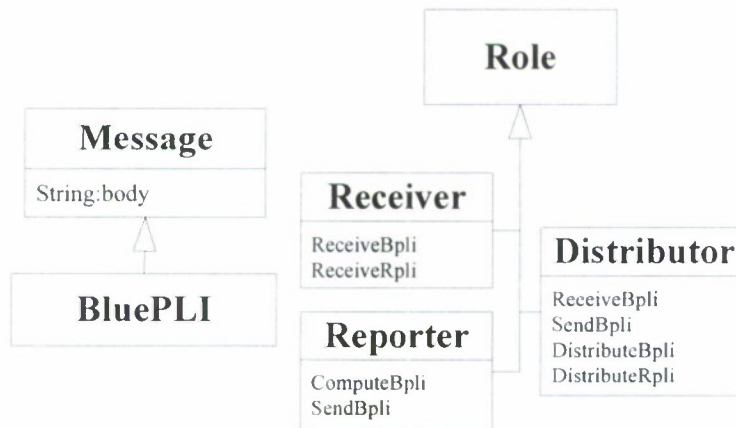


Figure 7.6. Blue Force Tracking Capability OV-7

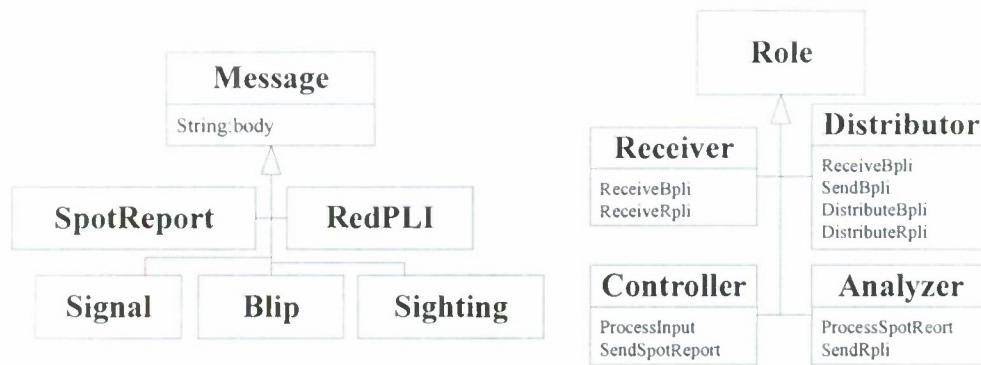


Figure 7.7. Process and Disseminate Intelligence Information Capability OV-7

The OV-6a, OV-2, and OV-7 are integrated in the Operational Activity Model, OV-5. The rules are realized in the actions shown in the UML Activity Diagrams representing each capability. Figure 7.8 is the Activity Diagram for Planning and Coordination followed by Figure 7.9 and Figure 7.10 for Blue Force Tracking and Process and Disseminate Intelligence Information, respectively. The operational roles from the OV-2s are identified in the swim lanes of each Activity Diagram. The activities modeled here are identified in the rule model and the data model. The Activity Model shows how the activities are connected and what data is passed between them.

7.4 System Architectures

Step 2 of the assessment process identifies the applicable system architecture views that realize the required capabilities defined in the operational architecture views identified in step 1. There is a one-to-many relationship between the operational architecture views and the system architecture views for this case study. The system architecture views use different architecture approaches to realize the capabilities described in the operational architecture views. There are three patterns described in the system architectures, peer-to-peer, centralized-server and service oriented architecture. Figure 7.11 shows the relationships between the operational architecture views and the system architecture views. The three architecture patterns resulted in three different SOSI Architectures that created related groups of SOSI alternatives. The P2P system architecture view will be presented in detail, the remaining approaches will only contribute to the results presented in step 6 and 7.

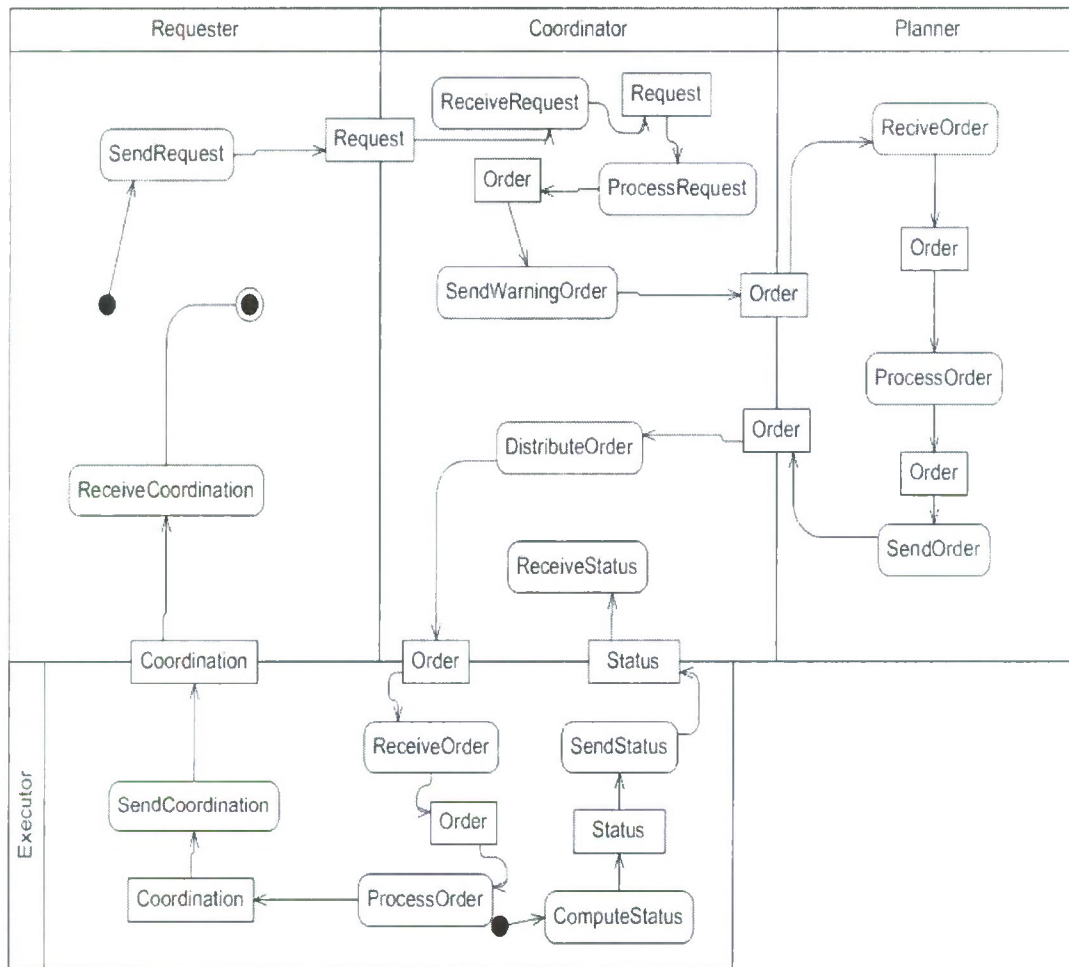


Figure 7.8. Planning and Coordination OV-5

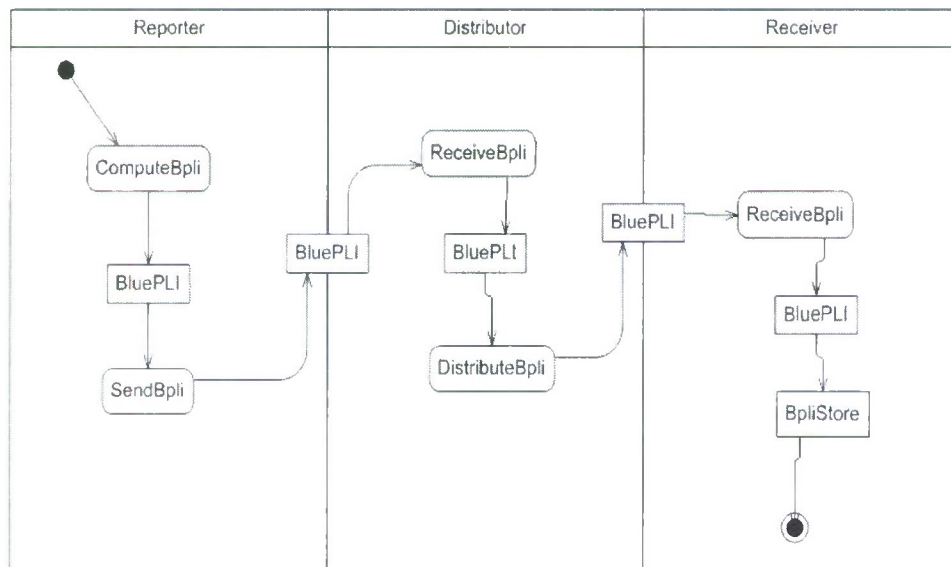


Figure 7.9. Blue Force Tracking OV-5

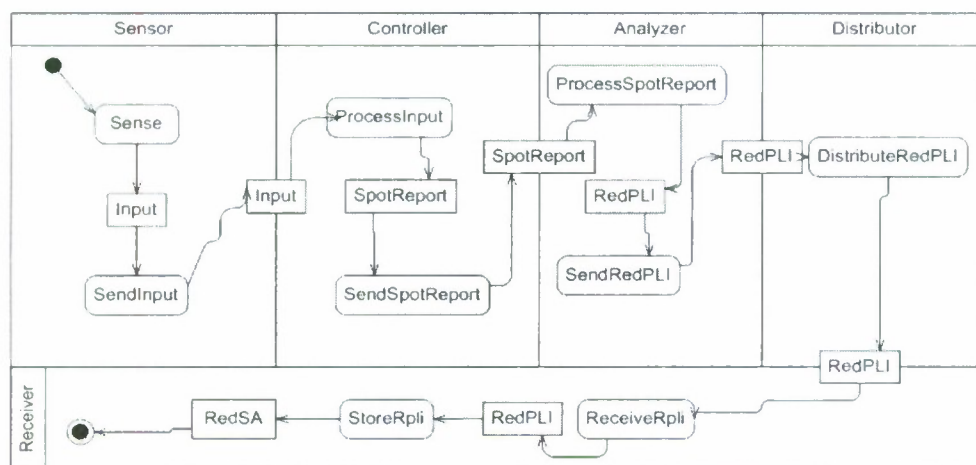


Figure 7.10. Process and Disseminate Intelligence Information OV-5

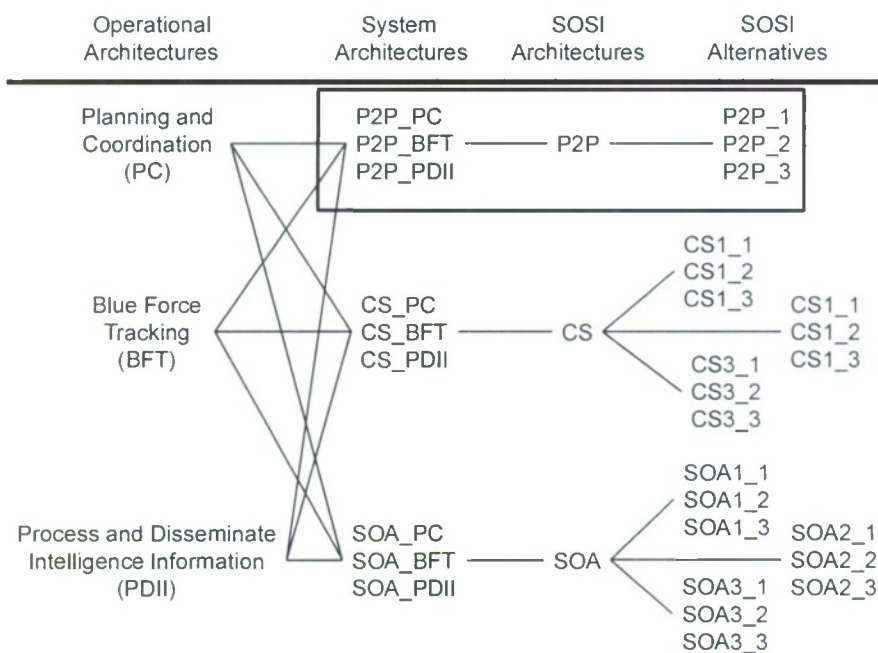


Figure 7.11. Relationship Among Architecture Views

The DODAF system architecture views required by the methodology follow: System Interface Description (SV-1), Systems Functionality Description (SV-4), Operational Activity to System Function Traceability Matrix (SV-5), System Data Exchange Matrix (SV-6), Systems Rules Model (SV-10a) and the Physical Schema (SV-11). There are three system architecture views, one for each capability described by the operational architecture views. The System Views will be grouped by DODAF product.

The SV-1 represents the system Elements used to realize the capability. It also represents the nodes the Elements are assigned and the messages passed between them. The SV-1s are represented as modified UML Communications Diagrams by adding the node assignments for the Elements. The SV-1s are shown in Figures 7.12, 7.13, and 7.14. The roles from the corresponding operational architecture are shown in the angle brackets for each Element instance. The node each Element is assigned to is identified by the box. There are four types of Elements: Tactical Level Command and Control System (TLC2S), Operational Level Command and Control System (OLC2S), Blue Force Tracking (BFT) and Intelligence Control System (ICS). These Elements interact in the following system architecture views to realize the capabilities described in methodology step 1.

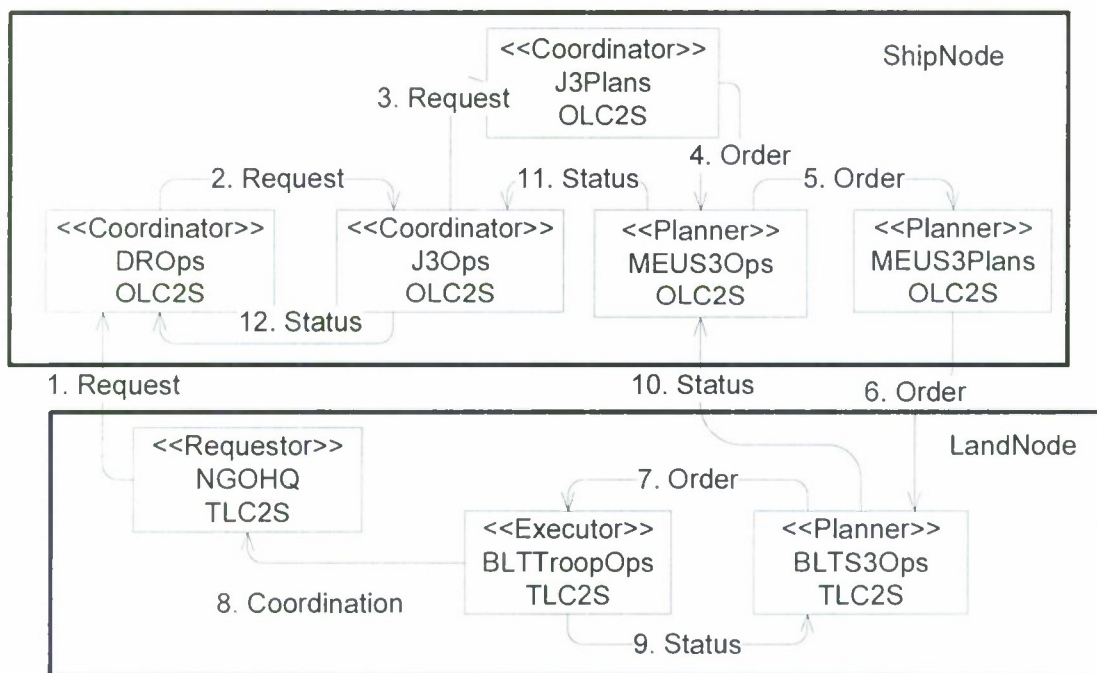


Figure 7.12. Planning and Coordination SV-1

The system architecture views are further defined by the SV-6. It provides the attributes of the Messages exchanged between the Systems. The SV-6s for are shown in Tables 7.3, 7.4, and 7.5. Each exchange between a Sender and Receiver shown in the SV-1 is a row in the SV-6. For example, Request Messages are sent from the TLC2S Elements to the OLC2S element. The message is text and its length is 32 characters.

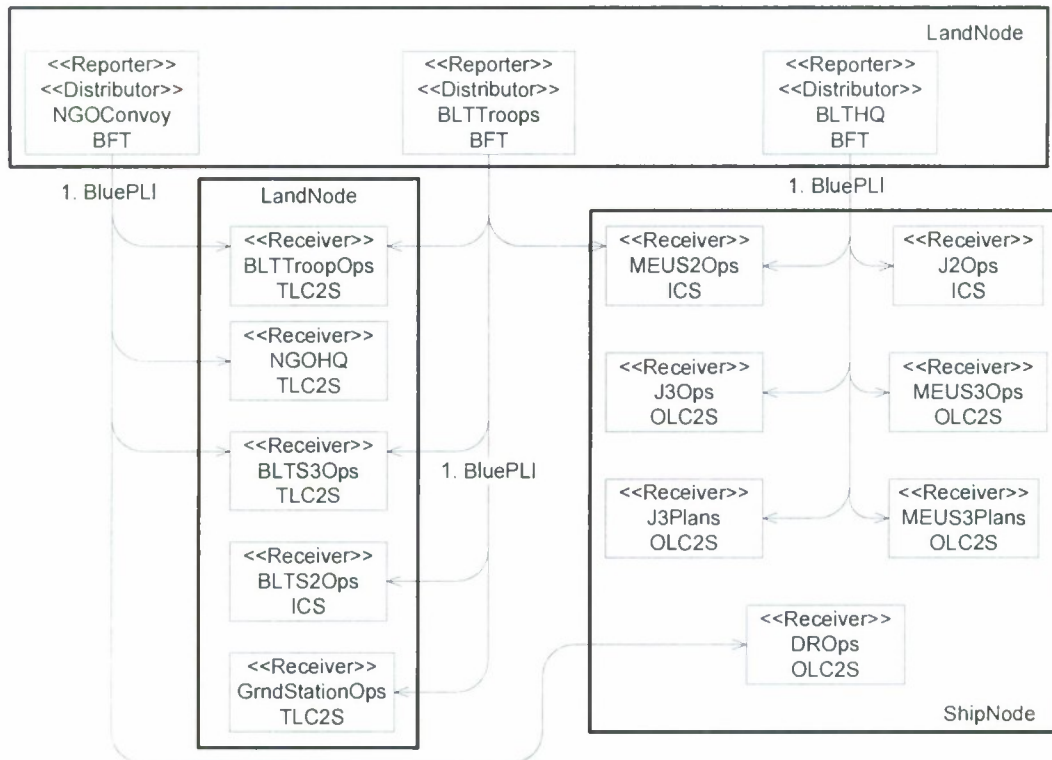


Figure 7.13. Blue Force Tracking SV-1

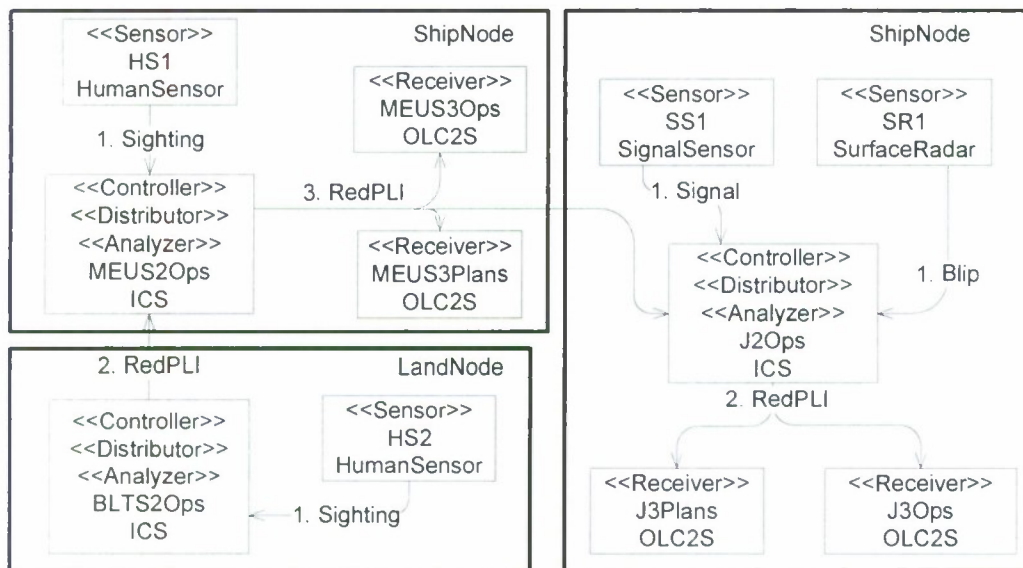


Figure 7.14. Process and Disseminate Intelligence Information SV-1

Table 7.3. Planning and Coordination SV-6

	Sender	Receiver	Type	Len
Request	TLC2S	OLC2S	Text	32
Request	OLC2S	OLC2S	Text	32
Order	TLC2S	TLC2S	Text	*
Order	OLC2S	OLC2S	Text	*
Order	OLC2S	TLC2S	Text	*
Coordination	TLC2S	TLC2S	Text	32
Status	TLC2S	TLC2S	Text	56
Status	OLC2S	OLC2S	Text	56
Status	OLC2S	TLC2S	Text	56

Table 7.4. Blue Force Tracking SV-6

	Sender	Receiver	Type	Len
BluePLI	BFT	OLC2S	Text	32
BluePLI	BFT	TLC2S	Text	32
BluePLI	BFT	ICS	Text	32
BluePLI	BFT	BFT	Text	32

Table 7.5. Process and Disseminate Intelligence Information SV-6

	Sender	Receiver	Type	Len
Sighting	HumintSensor	ICS	Text	48
Sighting	SigintSensor	ICS	Text	48
Signal	SurfaceRadar	ICS	Text	48
RedPLI	ICS	OLC2S	Text	48
RedPLI	ICS	TLC2S	Text	48

The SV-10a describes the rules that define the behavior of the system functions. In this case, the rules describe the action that the system takes upon receiving a particular type of message. Tables 7.6, 7.7, and 7.8 are the rule models for the system architectures. Table 7.6 shows that if a Request message is received send a Request message.

Table 7.6. Planning and Coordination SV-10a

TLC2S
if Request then Request
if Order $\diamond$ ord_warning then SendOrder
if Status then SendStatus
OLC2S
if Request then Request
if Order $\diamond$ ord_warning then SendOrder
if Status then SendStatus

Table 7.7. Blue force Trackin SV-10a

TLC2S
if BluPLI then PassBpli
OLC2S
if BluPLI then PassBpli
BFT
if BluePLI $\diamond$ rpli_pass the rpli

Table 7.8. Process and Disseminate Intelligence Information SV-10a

TLC2S
if RedPLI then SendRpli
OLC2S
if RedPLI then SendRpli
ICS
if SpotReport = sr_pers then rpli_pers
if SpotReport = sr_equip then rpli_equip
if RedPLI = rpli_pass then empty
if Blip = blip_pers then sr_pers
if Blip = blip_equip then sr_equip
if Signal = signal_pers then sr_pers
if Signal = signal_equip then sr_equip
if Sighting = sighting_pers then sr_pers
if Sighting = sighting_equip then sr_equip

The SV-11, shows the physical schema for each system architecture. Figure 7.15, 7.16, and 7.17 show the physical schemas for the system architectures. Each one describes the Messages and Systems that realize the capability.

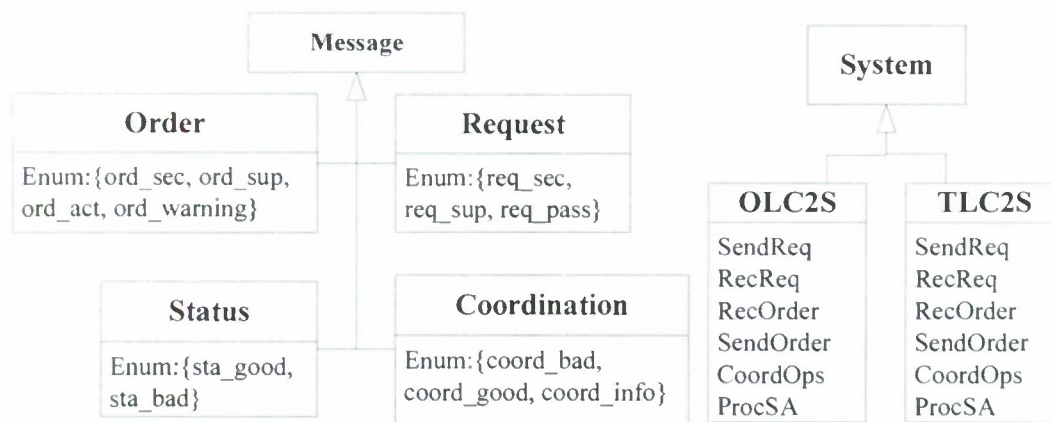


Figure 7.15. Planning and Coordination SV-11

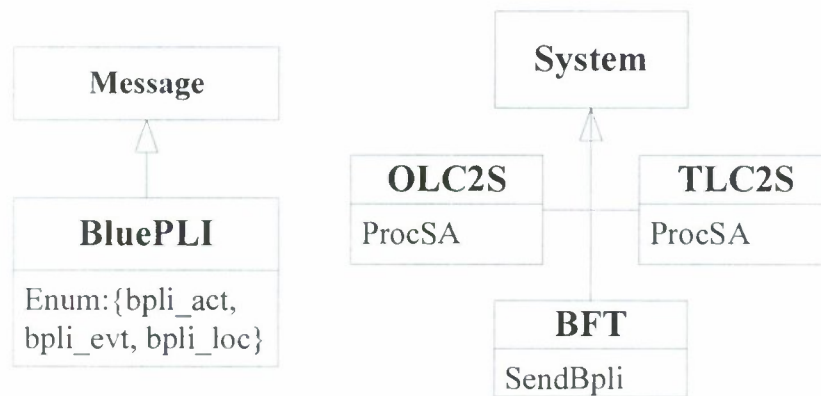


Figure 7.16. Blue Force Tracking SV-11

The SV-4 is represented in this methodology as an UML Activity Diagram. The activity diagram uses the information from all the system views presented to model the dynamic behavior of the system architecture view. The data represented in the SV-11 and SV-6 is represented by the ports showing the Message types passed between systems. The rules defined in the SV-10a are implemented in the Activity Diagram. The diagrams shown here represent the top level of a hierarchy of activity diagrams. The fork icon in means there is a lower level activity diagram the further represents the behavior of the system. Figures 7.18, 7.19, and 7.20 represent the SV-4s for the system architectures.

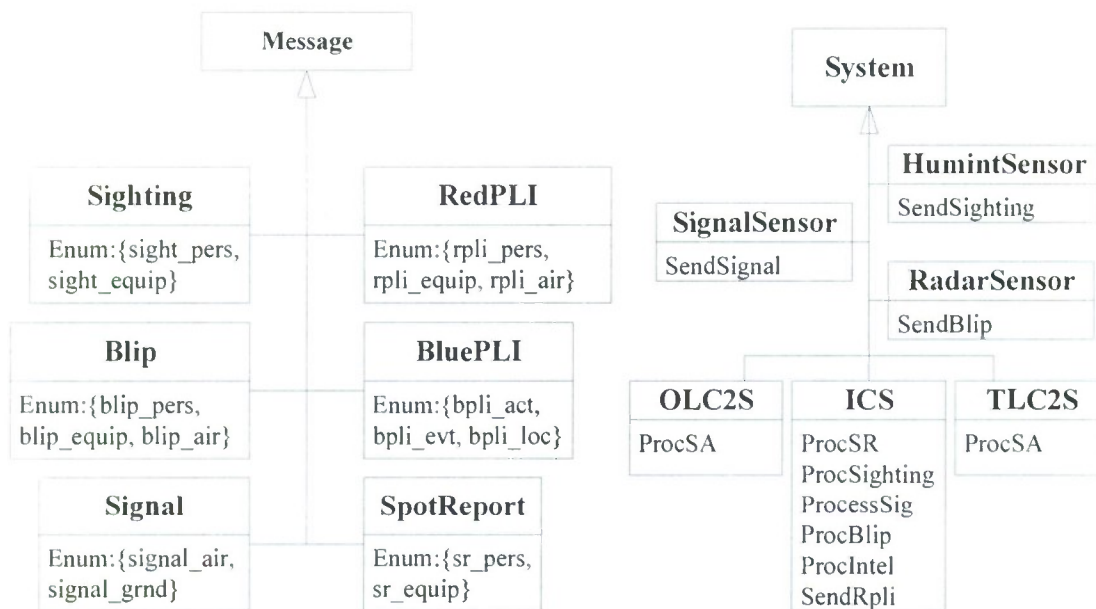


Figure 7.17 Process and Disseminate Intelligence information SV-11

Figure 7.21 is an example of one of the lower activity diagrams that describes the behavior of a system. It represents the behavior of the TLC2S. Notice that the actions defined in the diagram map to the functions described in the SV-6, SV-5 and SV-11.

The SV-5 is a matrix that maps the operational activities shown in the columns on the left with the system functions shown in the rows at the top. This matrix provides the traceability from the system architecture view back to the operational architecture view. It is used to ensure that all the operational activities are realized by a system function. Using Table 7.9 as an example, the operational role Coordinator has an activity ProcessRequest. The TLC2S and OLC2S systems both realize this operational activity. The name of the system function that realizes the activity in both systems is Rec Req (ReceiveRequest). The SV-5s for each capability are represented in Tables 7.9, 7.10, and 7.11.

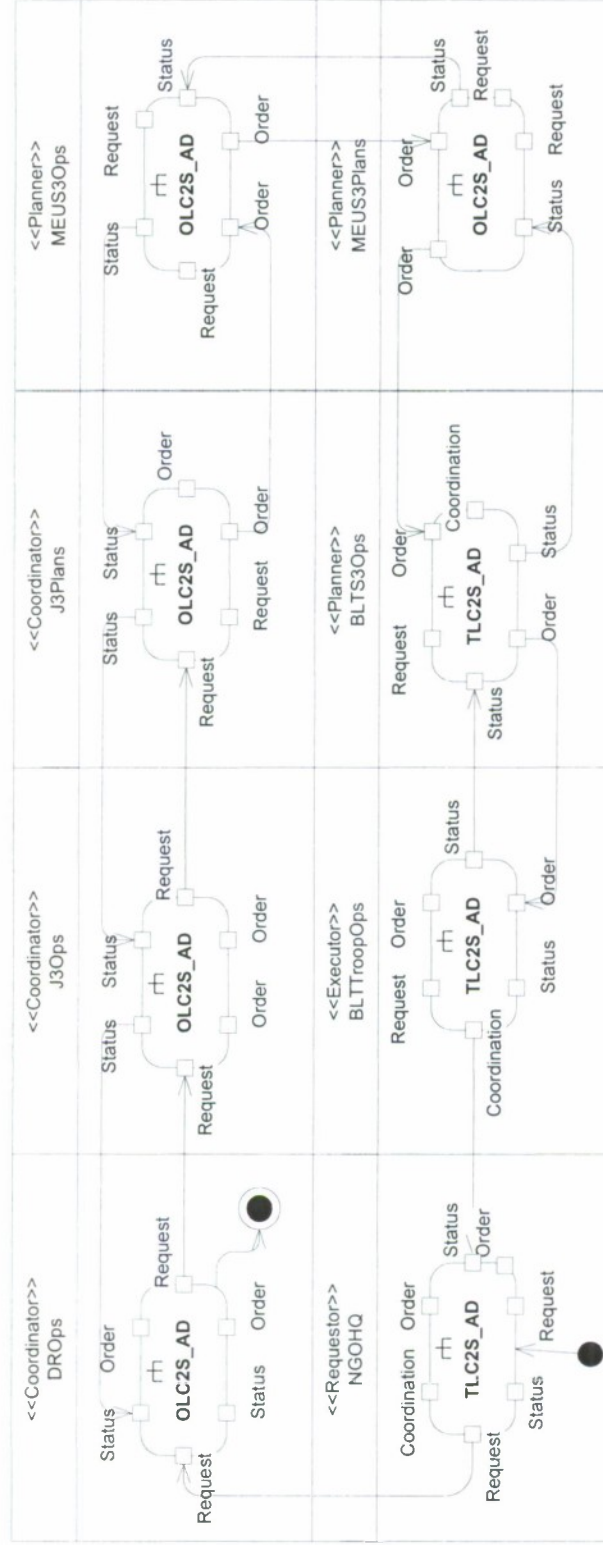


Figure 7.18. Planning and Coordination SV-4

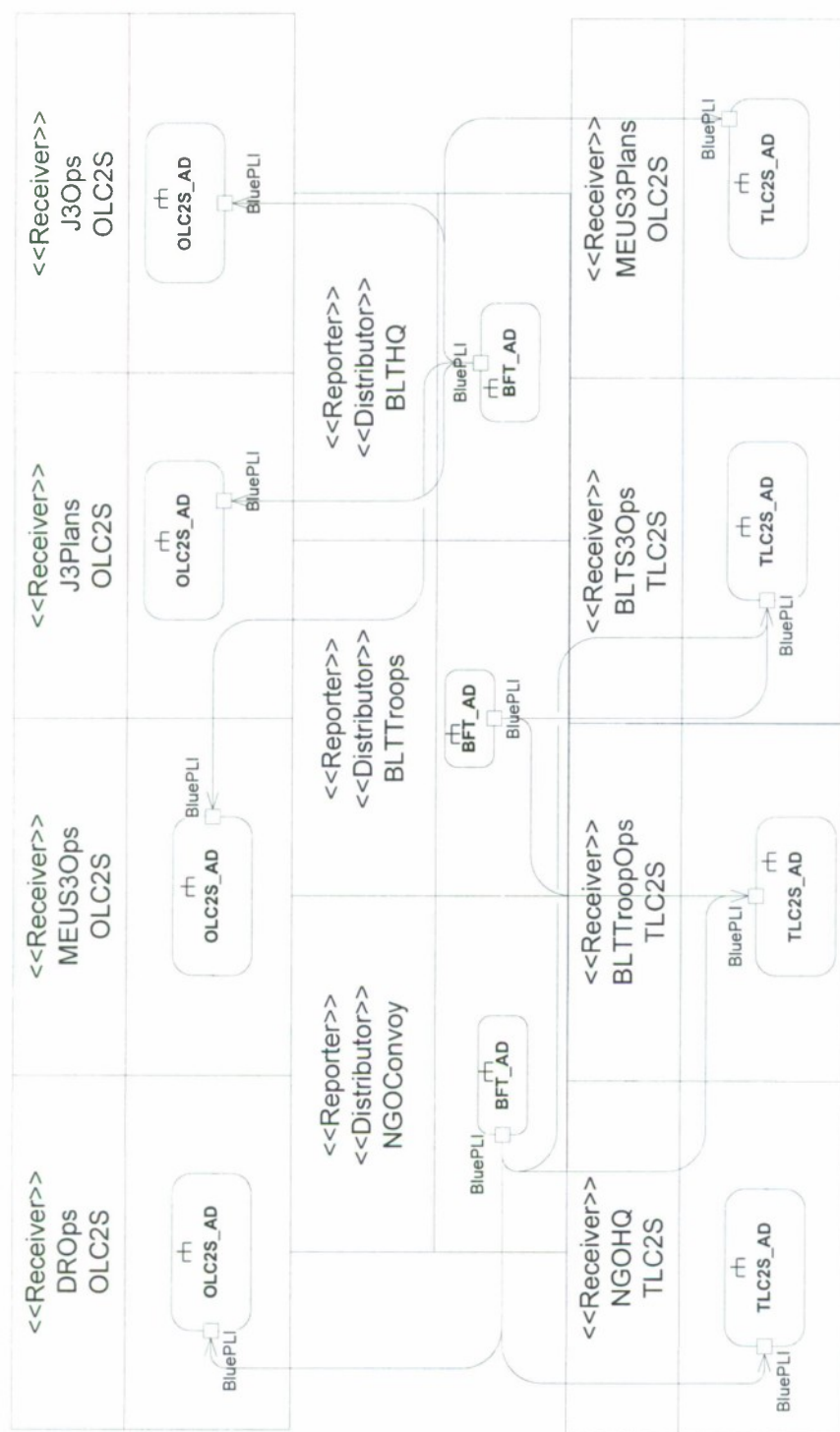


Figure 7.19. Blue Force Tracking SV-4



Figure 7.20. Process and Disseminate Intelligence Information SV-4

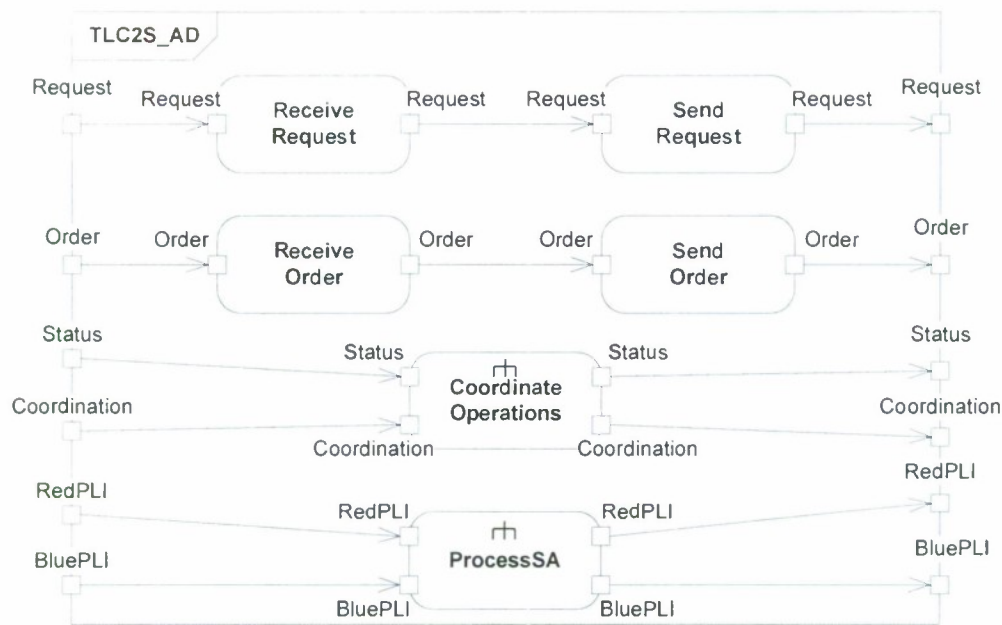


Figure 7.21. Tactical Level Command and Control System Activity Diagram

7.5 System of System Instance Architecture

Step 3 of the methodology merges the system architecture views into SOSI Architectures that represent the different approaches used by the system architecture views and requirements described by the implementing organization. There are four SOSICs created from the system architecture views. They are Conduct Security Operations, Conduct Support Operations, Blue Force Tracking and Process and Disseminate Intelligence Information. Conduct Security Operations and Conduct Support Operations are both realizations of the Planning and Coordination Capability. The ESG requires diverse capabilities. By modeling both SOSICs the utilization of Element resources that must be applied to disparate tasks can be assessed.

The system architecture views identified in step 2 yielded three different architectural approaches: P2P, CS and SOA. The P2P SOSI Architecture will be shown in detail while only the results of the analysis will be shown for the CS and SOA SOSI Architectures. The P2P SOSI Architecture fulfills the requirement of the operational view using a peer to peer architecture concept. In P2P there are no central servers. Each Element is connected to its peers in a predetermined fashion. Figure 7.22 shows a diagram that represents the P2P SOSI without Nodes assigned. The lines represent connections between Elements that facilitate the sending and receiving of messages defined in the SOSICs.

Table 7.9. Planning and Coordination SV-5

	System	TLC2S					OLC2S				
Role	Activity/ Function	Send Req	Rec Req	Rec Ord	Send Order	Coord Ops	Send Req	Rec Req	Rec Ord	Send Ord	Coord Ops
Requestor	SendRequest	X	X				X	X			
	Receive Coordination					X					X
Coordinator	Receive Request		X					X			
	ProcessRequest		X				X				
	SendOrder				X					X	
	Distribute Order				X					X	
	RecieveStatus					X					X
Planner	RecieveOrder			X					X		
	ProcessOrder			X					X		
	SendOrder				X					X	
Executor	ReceiveOrder			X							
	ProcessOrder				X						
	Send Coordination					X					
	Compute Status					X					X

Table 7.10. Blue Force Tracking SV-5

	System	TLC2S	OLC2S	BFT
Role	Activity/Function	Proc SA	Proc SA	Send Bpli
Reporter	ComputeBpli			X
	SendBpli			X
Distributor	ReceiveBpli	X	X	
	DistributeBpli	X	X	
Receiver	RecieveBpli	X	X	

Table 7.11. Process and Disseminate Intelligence Information SV-5

	System	TLC2S	OLC2S	ICS						Sensor
Role	Activity /Function	Proc SA	Proc SA	Proc SR	Proc Sight	Proc Sig	Proc Blip	Proc Intel	Send Rpli	SR SS HS
Sensor	Sense									X
	SendInput									X
Controller	ProcessInput				X	X	X			
	SendSpotReport			X						
Analyzer	ProcessSpotReport			X						
	SendRedPLI			X				X		
Distributor	DistributeRedPLI							X	X	
Receiver	ReceiveRpli	X	X					X	X	
	StoreRpli	X	X					X	X	

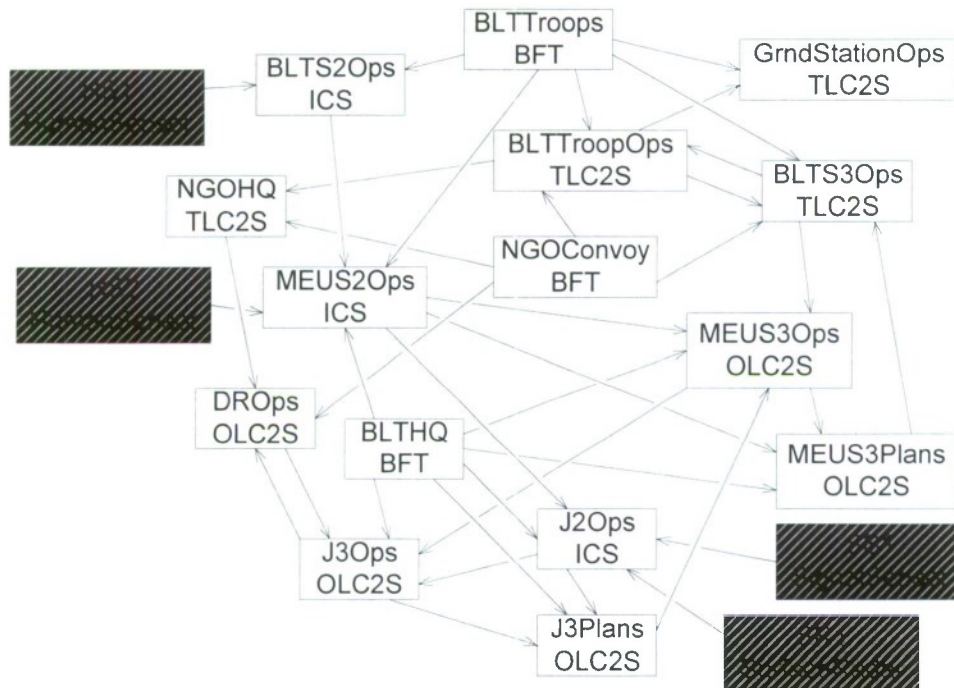


Figure 7.22. Peer-to-Peer Architecture

The second SOSI Architecture used for comparison in the case study is the centralized-server (CS) architecture. As the name implies there are central servers that facilitate the passage of information from one Element to another. Where the P2P architecture used direct connection between Elements, the CS architecture relies on a server to pass information from one Element to another. There are three CS SOSI groups. They differ in the number of servers up to three servers in the SOSI. Figure 7.23 shows the architectures for the one server SOSI.

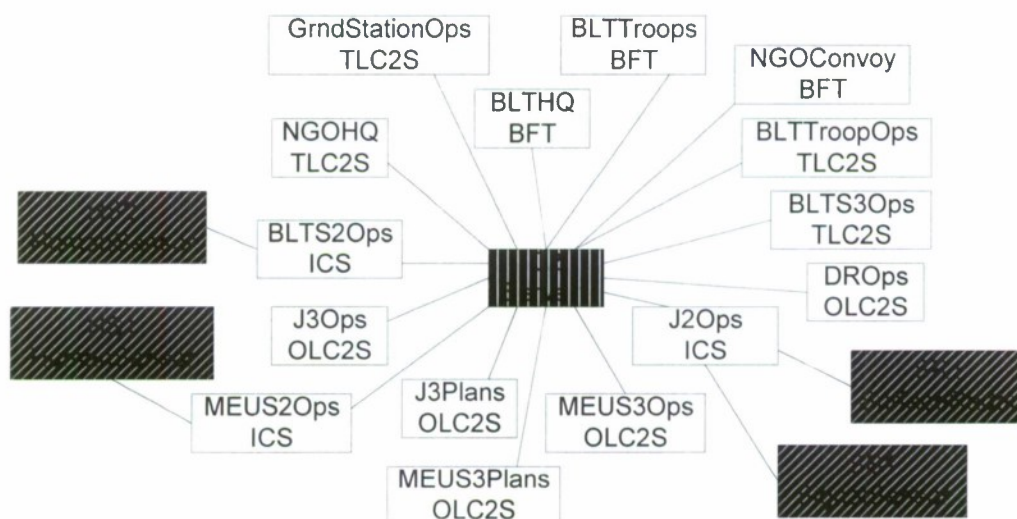


Figure 7.23. Client Server SOSI with One Server

The last SOSI Architecture represents a Service Oriented Architecture. This type of architecture is characterized by instances of services that accomplish specific tasks for the organization and facilitate communication between the Elements. The SOA SOSI groups are differentiated by the number of instances of each service type. There are five services defined: Planning, Coordination; Request, ISR and BFT. There are three different SOSI groups that use SOA architecture configuration. They are differentiated by the number of instances of each service. SOA1 SOSI, Figure 7.24, shows a single instance of each service.

The P2P SOSI Architecture will be used as the example for the remaining steps of the methodology. Figure 7.25 is part of the SV-11 for the P2P SOSI Architecture. It shows the relationships between the Nodes and Elements. Another diagram completes the SV-11 by showing the Message types. The Expeditionary Strike Group System (ESGS) is the top level class and represents the SOSI as a whole. There are two types of Nodes, ShipNode and LandNode. There are four types of Elements: Tactical Level Command and Control System (TLC2S), Operational Level Command and Control System (OLC2S), Intelligence Control System (ICS) and Blue Force Tracker (BFT). Finally there are three Elements that represent sensors in the SOSI: SurfaceRadar (SR), SignalSensor (SS) and HumanSensor (HS).

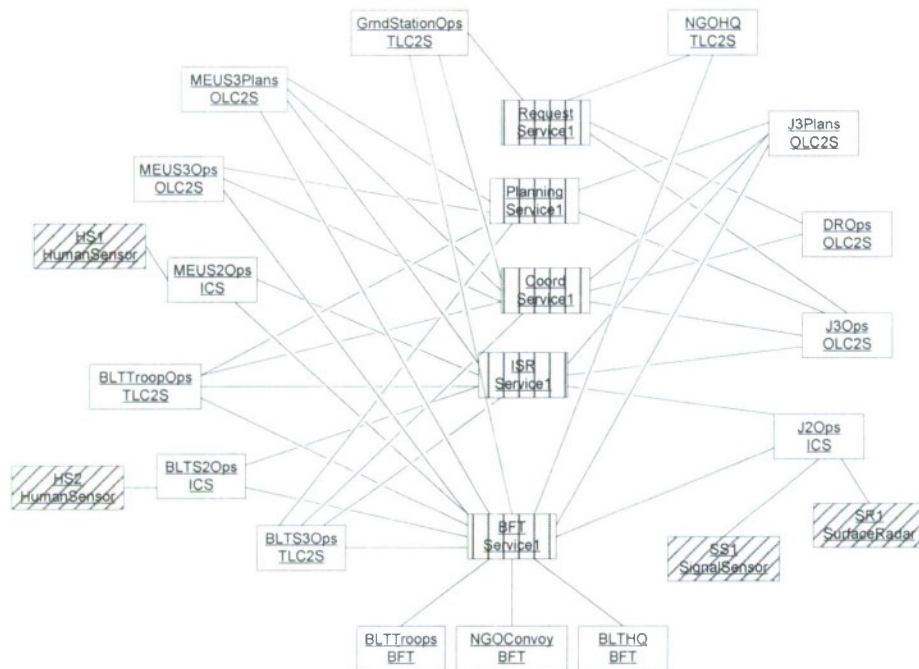


Figure 7.24. Service Oriented Architecture with One Instance of Each Service

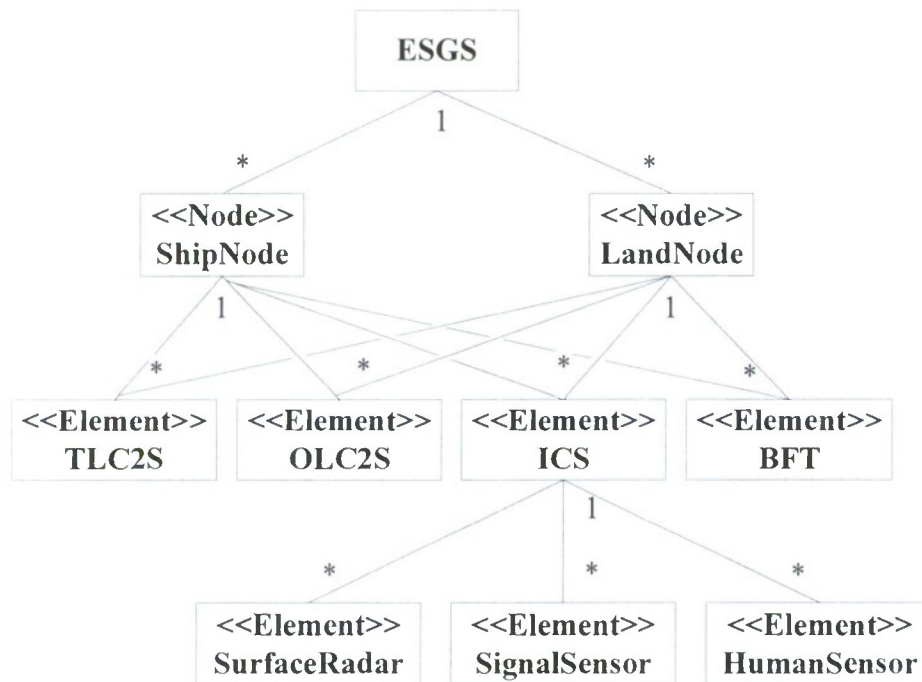


Figure 7.25. P2P SOSI Architecture SV-11, part 1

The methodology requires the SV-1, SV-4, SV-5, SV-6, SV-10a and SV-11. The case study looks at three SOSI alternatives of the P2P SOSI Architecture differentiated by the Node

configuration shown in the SV-1s. The SV-1 changes for each node configuration. Figures 7.26, 7.27, and 7.28 show the three different Node configurations used for evaluation of the P2P SOSI. The measures will reveal that certain configurations are more adaptable to change than others.

The first Node partition, Figure 7.26 partitions the Elements by echelon. There are two nodes on the island that represent the supported population, Ground Station and Port. The Ground Station represents the communication station described in the scenario. The Port Node represents a connection to the government of Efratia. The Tarawa Node is the ESG command ship. The Harper's Ferry Node is the Marine Expeditionary Unit (MEU) Headquarters command ship. The Beach Node is the location of the Battalion Landing Troops that are controlled by the other echelons.

The second Node partition, Figure 7.27, is functional. The Tarawa Node has the operations Elements and the Harper's Ferry Node has the Planning Elements. The rest of the Elements are arrayed over 6 other Nodes. The third partition, Figure 7.28 groups all the elements strictly by echelon. This is different from the first partition because the MEU and BLT are all on the same Node in this partition.

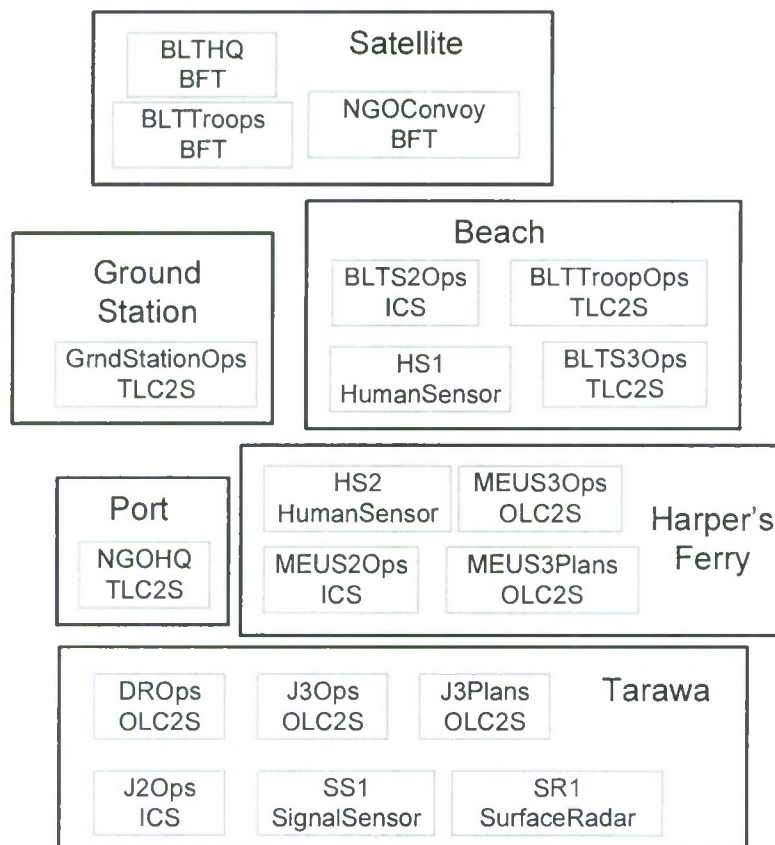


Figure 7.26. P2P_1 SV-1 Six Nodes

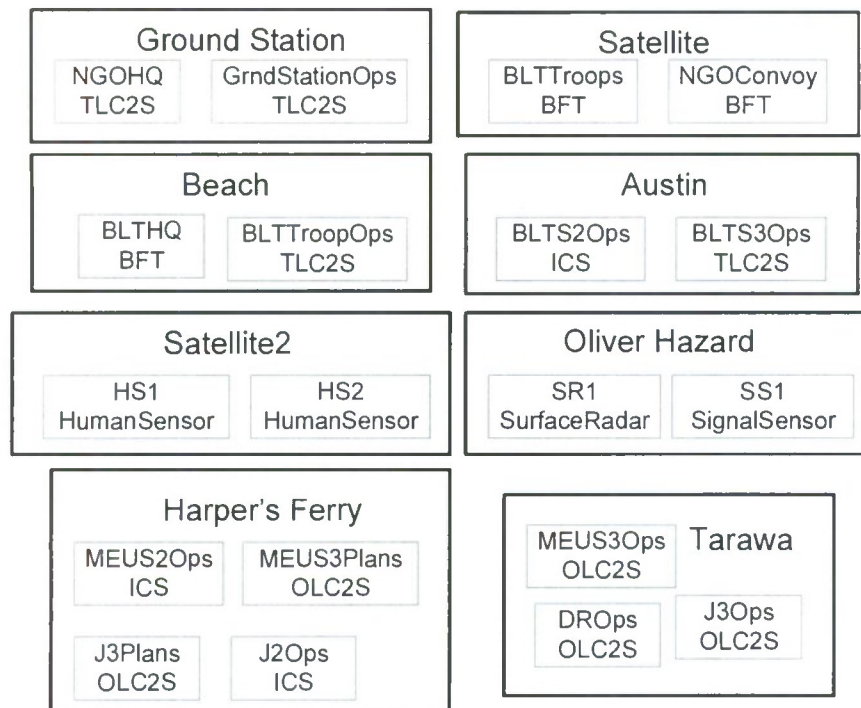


Figure 7.27. P2P_2 SV-1 Eight Nodes

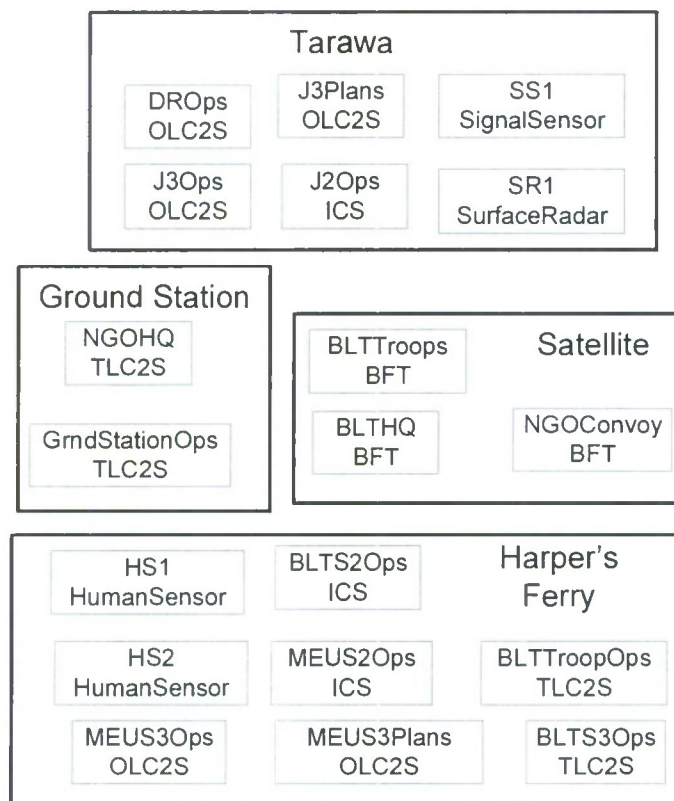


Figure 7.28. P2P_3 SV-1 Four Nodes

The SOSI Architecture SV-6, Table 7.12, is the result of merging the SV-6s from the P2P system architecture views. This view is focused on the interfaces between Elements. For example, Element TLC2S sends Request messages which are received by Element OLC2S. A Request message is text and 32 characters in length. These interfaces will be represented in the SOSICs modeled as SV-4s.

Table 7.12. P2P Systems Data Exchange Matrix, SV-6

	Sender	Receiver	Type	Len
Request	TLC2S	OLC2S	Text	32
Request	OLC2S	OLC2S	Text	32
Order	TLC2S	TLC2S	Text	*
Order	OLC2S	OLC2S	Text	*
Order	OLC2S	TLC2S	Text	*
Coordination	TLC2S	TLC2S	Text	32
Status	TLC2S	TLC2S	Text	56
Status	OLC2S	OLC2S	Text	56
Status	OLC2S	TLC2S	Text	56
BluePLI	BFT	OLC2S	Text	32
BluePLI	BFT	TLC2S	Text	32
BluePLI	BFT	ICS	Text	32
BluePLI	BFT	BFT	Text	32
Sighting	HumintSensor	ICS	Text	48
Sighting	SigintSensor	ICS	Text	48
Signal	SurfaceRadar	ICS	Text	48
RedPLI	ICS	OLC2S	Text	48
RedPLI	ICS	TLC2S	Text	48

The SV-10a, Table 7.13, merges the rules implemented by the various Elements of the system architecture views. For example the ICS Element has a rule: if Blip_pers then sr_pers. The rule means that when an ICS instance receives a blip_pers message it should send out a sr_pers SpotReport message.

The SV-11, Figure 7.29, is the merged data model from the system architecture views. It completes the SV-11 from Figure 7.25. In this case the various message contents are represented by enumerated values. For example, Status can be either good or bad. The enumerated values are sta_good and sta_bad. In this view the Elements show the system functions that are modeled in each.

Table 7.13. P2P Systems Rule Model, SV-10a

TLC2S
if Request then Request
if Order $\Diamond$ ord_warning then SendOrder
if Status then SendStatus
if SpotReport then PassSpotReport
if BluPLI then PassBpli
if RedPLI then SendRpli
OLC2S
if Request then Request
if Order $\Diamond$ ord_warning then SendOrder
if Status then SendStatus
if SpotReport then PassSpotReport
if BluPLI then PassBpli
if RedPLI then SendRpli
BFT
if BluePLI $\Diamond$ rpli_pass the rpli
ICS
if SpotReport = sr_pers then rpli_pers
if SpotReport = sr_equip then rpli_equip
if RedPLI = rpli_pass then empty
if Blip = blip_pers then sr_pers
if Blip = blip_equip then sr_equip
if Signal = signal_pers then sr_pers
if Signal = signal_equip then sr_equip
if Sighting = sighting_pers then sr_pers
if Sighting = sighting_equip then sr_equip

The SV-4 describes the SOSICs. The SOSIC remain the same for each P2P SOSI alternative. The Elements may occupy various Nodes but the interconnections between Elements defined by the SOSIC remains the same. The activity diagrams used to model the SV-4 implement the rule model and data model defined above. They also implement the data exchanges defined in the SV-6 and the data types represented in the SV-11.

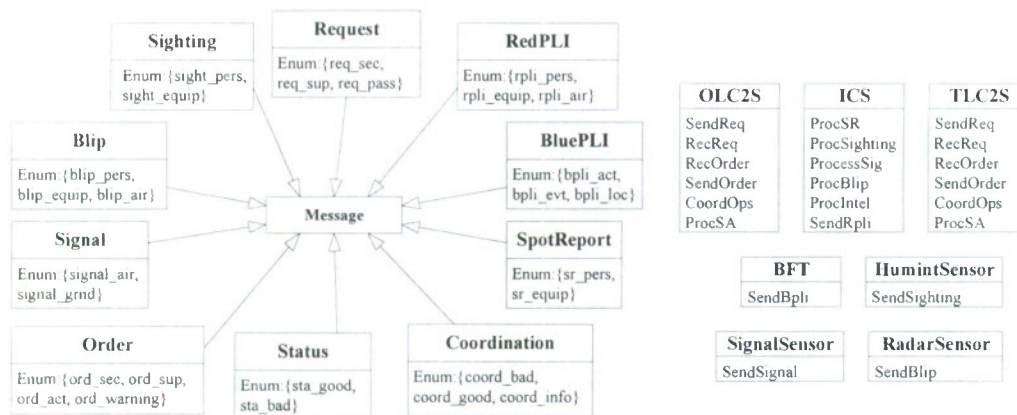


Figure 7.29. P2P SOSI Architecture SV-11

Figures 7.30, 7.31, 7.32, and 7.33 are the Activity diagrams that represent the SV-4s for each SOSIC. Each partition represents a specific Element instance. Each partition contains the Activity diagram for the Element type of the instance. Notice that there are many unused Element interfaces in each SV-4. There is reuse of Elements among the SOSICs and each SOSIC may use different interfaces. The combined CPN captures this reuse of Element instances and ensures the CPN models all the interfaces connected in the SOSICs.

For example, Element6 represents the MEUS3Ops instance of an OLC2S Element. Element6 appears in every SOSIC, identified by the box.. All the interfaces used by Element6 will be modeled. Those interfaces in the OLC2S_AD that are not used by Element6 in any SOSIC will be stubbed out in the transformed CPN.

The SV-5 merges the SV5s from the P2P system architectures. Table 7.14 is the merged SV-5. There are four Systems that are represented. The Operational Activities are distributed across the elements. This system view helps trace system functions modeled in the executable back to the operational architecture representing the capability.







Figure 7.33. Conduct Security Operations SOSIC

Table 7.14. P2P SOSI Architecture SV-5

		TLC2S						OLC2S						ICS						BFT		Sensor
		Send Req	Rec Req	Rec Ord	Send Ord	Coord Ops	Proc SA	Send Req	Rec Req	Rec Ord	Send Ord	Coord Ops	Proc SA	Proc Sight	Proc Sig	Proc Blip	Proc Intel	Send Rpli	Send Bpli	Send SR	Send HS	
Requestor	SendRequest	X	X					X	X													
	ReceiveCoord					X						X										
Coordinator	RecieveRequest		X						X													
	ProcessRequest		X					X														
	SendOrder				X						X											
	DistributeOrder				X						X											
	ReceiveStatus					X						X										
Planner	ReceiveOrder		X						X													
	ProcessOrder		X						X													
	SendOrder				X						X											
Executor	ReceiveOrder		X																			
	ProcessOrder				X																	
	SendCoord					X																
	ComputeStatus					X						X										
	SendStatus						X					X										
Reporter	ComputeBpli																		X			
	SendBpli																		X			
Distributor	ReceiveBpli						X						X									
	DistributeBpli						X						X									
Receiver	RecieveBpli						X						X									
Sensor	Sense																				X	
	SendInput																				X	
Controller	ProcessInput													X	X	X						
	SendSpotReport																					
Analyzer	ProcSpotReport												X									
	SendRedPLI												X				X					
Distributor	DistriRedPLI																X					
Receiver	ReceiveRpli																X					
	StoreRpli																X					

Step 4 transforms the SOSICs from Step 3 into the executable model. This methodology transforms the activity diagrams that represent the SOSICS into a hierarchical CPN that represents the Elements and the behavior of the Elements. The Elements are allocated to particular Nodes based on the SV-1s. The following diagrams show the CPN created for the P2P_1 alternative. Figure 7.34 represents the top level CPN graph. The Nodes have been transformed into substitution transitions that represent the subpages shown in the following six figures. The ovals are places that represent the inputs and outputs of the Nodes. The Elements are represented as substitution transitions and represent the Node configurations from the SV-1s.

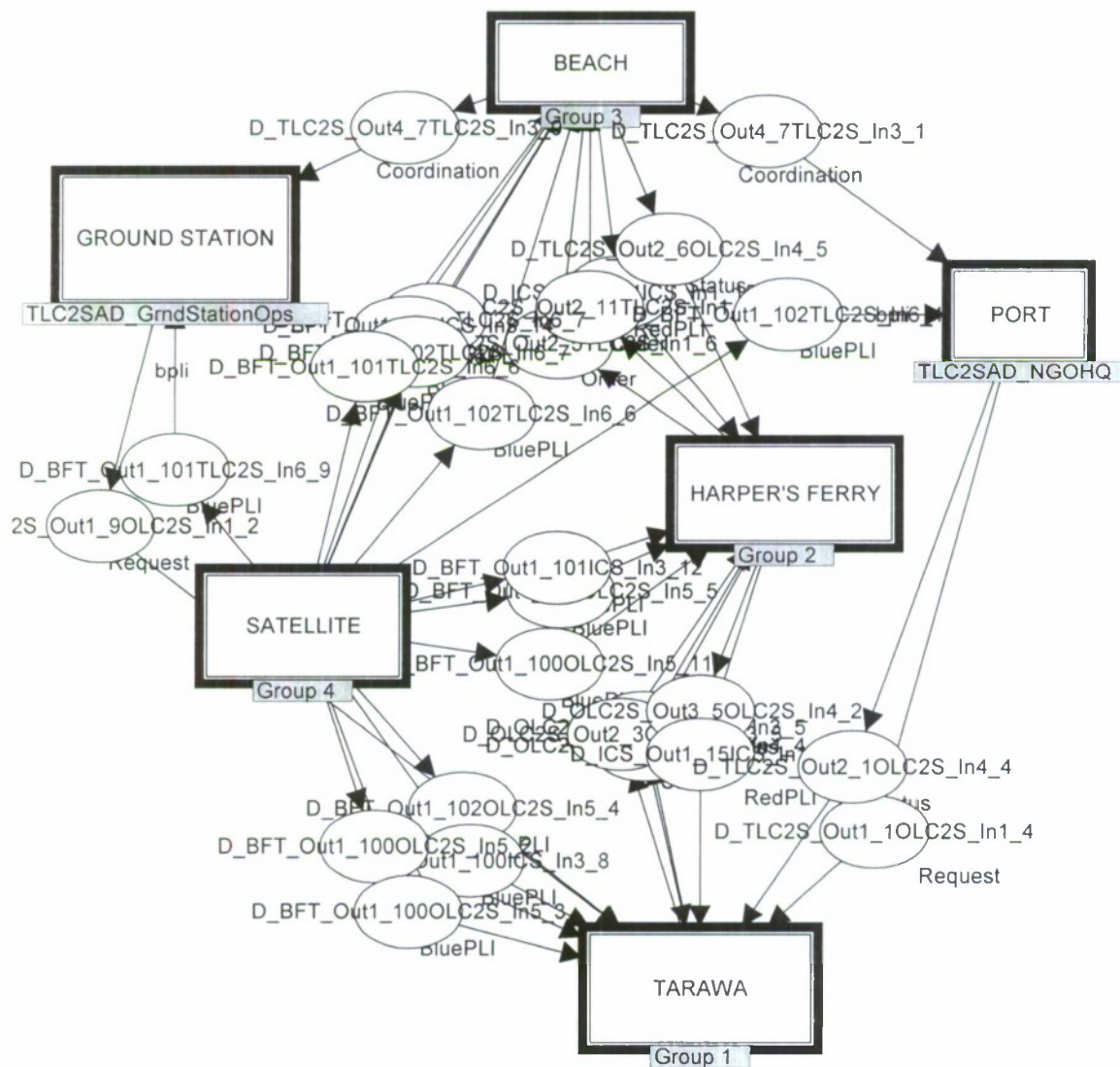


Figure 7.34. P2P_1 SOSI Alternative Top Level CPN Representation

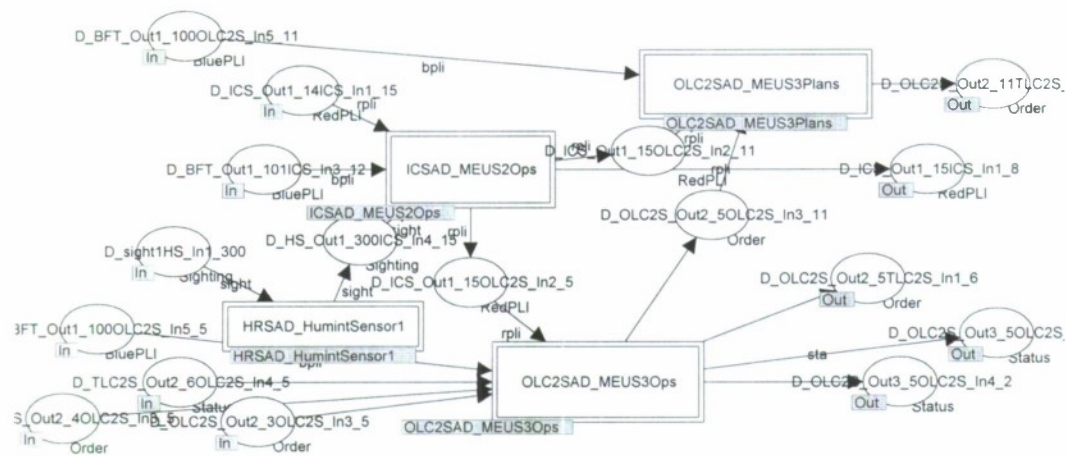


Figure 7.38. P2P_1 SOSI Harper's Ferry Node CPN

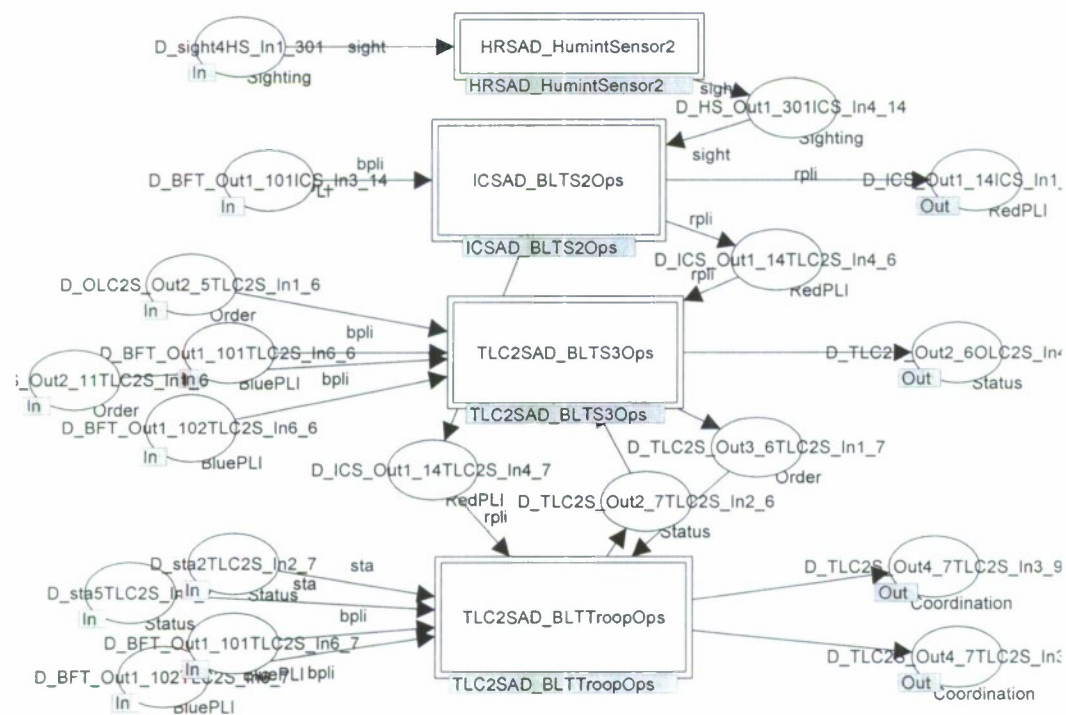


Figure 7.39. P2P_1 SOSI Beach Node CPN

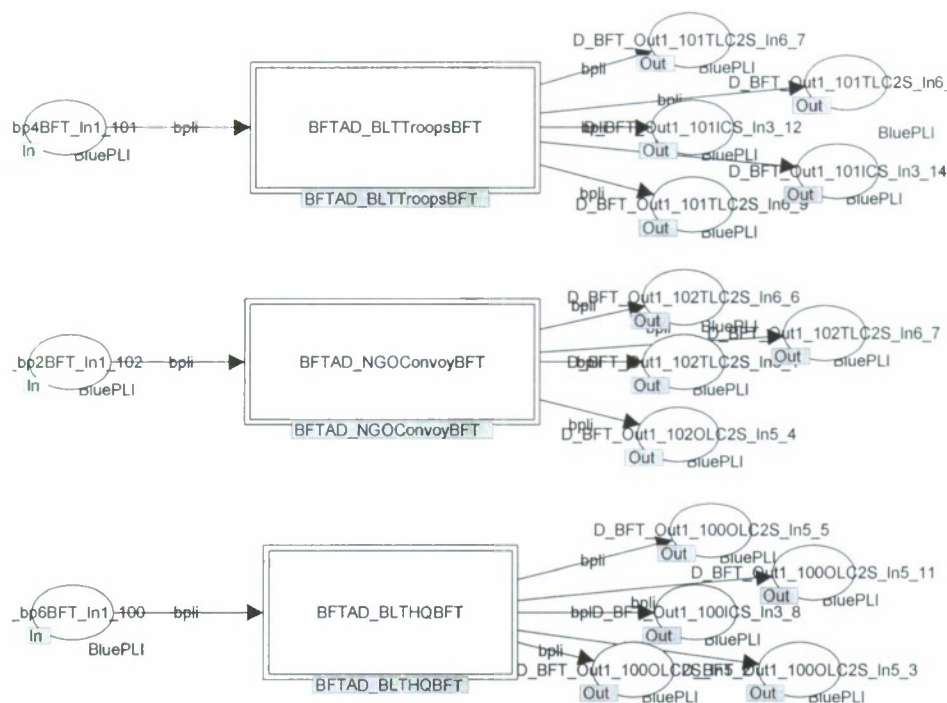


Figure 7.40. P2P_1 SOSI Satellite Node CPN

7.6 Assessment Measure Calculations for P2P SOSI

This section shows the calculation of the SOSI performance measures for the first SOSI alternative of the P2P SOSI Architecture. There are three alternative SOSI configurations: Figures 7.26, 7.27, and 7.28. In this example, the Elements are distributed by echelon over six Nodes. Then Adaptability and Agility are calculated for each SOSI alternative. The example starts with the calculation of Overlap which is the same for all SOSI alternatives in the group.

Cohesion

The CPN for each SOSI reflects the Node configuration. The Cohesion measure was made on each resulting Node structure using a graph analysis of the CPN that represents each Node. All Nodes are shown in the figures above. The Cohesion measurements for each Node are summarized in Table 7.15. The Tarawa Node has 12 inputs and 2 outputs. The number of possible connections is 24. The number of paths connecting inputs and outputs is 66 for Node Cohesion of 2.75. The SOSI Cohesion is the average Node Cohesion with a value of 1.22.

Table 7.15. Cohesion SOSI P2P_1

Figure	Node	Name	I	Q	x	z	Coh
Fig. 7.35	1,1	Port	3	1	3	3	1.0
Fig. 7.36	1,2	Ground Station	3	1	3	3	1.0
Fig. 7.37	1,3	Tarawa	12	2	24	66	2.75
Fig. 7.38	1,4	Harper's Ferry	8	5	40	35	.88
Fig. 7.39	1,5	Beach	10	4	40	54	1.35
Fig. 7.40	1,6	Satellite	3	14	42	14	.33
	SOSI						1.22

Coupling

Coupling was calculated using the CPN that was created for the P2P SOSI group. The CPN model was modified for each SOSI alternative to model the three different Node configurations. Then the monitors were added that count the number messages that are exchanged between Nodes. After execution of the CPN, the results of the data collected by the monitors is summarized into the Coupling results shown in Table 7.16. The sending Nodes are the columns and the receiving Nodes are the rows. The Port Nodes sends 15 messages to the Ground Station Node. There are 6 Nodes which makes the number of possible Links equal to $(6*5)/2 = 15$ Links. The SOSI Coupling is the average of the Node Coupling with a value of 3.21.

Table 7.16. SOSI P2P_1 Coupling Results

Node	Port	Ground Station	Tarawa	Harper's Ferry	Beach	Satellite	Coupling	
Port	x	15					15/15	1
Ground Station	20	x	50				70/15	4.67
Tarawa		30	x		30		60/15	4
Harper's Ferry	40	30	50	x	10	10	140/15	9.33
Beach	2				x		2/15	.13
Satellite	2					x	2/15	.13
							SOSI	3.21

Degree of Reuse and Exclusiveness

The Degree of Reuse calculations for the P2P SOSI group appear below. Tables 7.17 and 7.18 show the data and results of the Degree of Reuse and Exclusiveness calculations. The P2P SOSI alternative group average Degree of Reuse is 2.08. This results in an Exclusiveness measure of 0.48. That means that there reuse among the Elements of the SOSI. There is potential for contention of Element resources, the Elements that are members of three or four SOSIC warrant scrutiny. The highest degree of reuse is 4. This analysis highlights to developers the potential importance of the highly reused Elements.

Table 7.17. P2P Degree of Reuse Data

	1	2	3	4	5	6	7	8	9	11	14	15	100	101	102	200	300	301	400
BFT	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x				
ISR		x	x		x			x		x	x	x				x	x	x	x
GRN		x	x		x	x	x		x	x									
NGO	x	x	x	x	x	x	x												
	2	4	4	2	4	3	3	2	2	3	2	2	1	1	1	1	1	1	1

Table 7.18. P2P Degree of Reuse and Exclusiveness Results

Degree of Reuse			
Avg Element	2.08	High Reuse	4.00
Exclusiveness			0.48

7.7 Case Study Results

To accomplish the methodology comparisons, three SOSI alternatives (vary Node configuration) were developed for each SOSI group. The same SOSI Architecture can describe SOSIs that have different Element sets therefore the SOSI groups are SOSI alternatives that share the same Element set. SOSI groups share the same SOSIC definitions. Every SOSI alternative has the same set of end Elements and Sensor Elements but differ by the infrastructure Elements that differentiate the architecture patterns. There are three alternative SOSI Architecture concepts that are compared in the case study: peer-to-peer (P2P), centralized-server (CS) and service oriented architecture (SOA). The P2P SOSI Architecture generated three SOSI alternatives for comparison. The CS and SOS SOSI Architectures have three SOSI groups each with three SOSI alternatives for each group for a total of nine SOSI alternatives for CS and SOA SOSI Architectures.

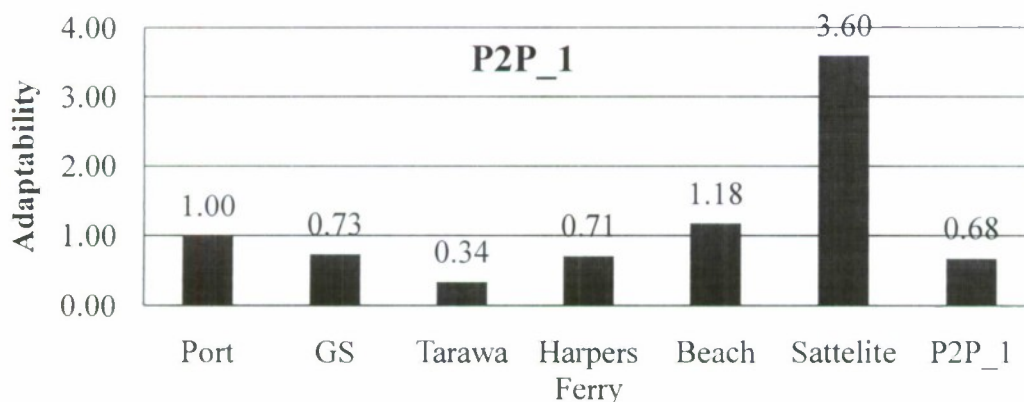
7.8 P2P Results

This section describes the results for the P2P SOSI Architecture. Adaptability is computed for each Node in each P2P alternative. The Node results reveal Nodes that assess low Adaptability relative to the other Nodes. This can show SOS architects the Nodes that significantly affecting overall SOSI Adaptability. This can assist in focusing development effort to improve the assessed Adaptability.

Fig. 7.41 shows the Adaptability results for the P2P alternatives by node. The last result in each graph is the SOSI Adaptability. In P2P_1 the Adaptability of the Satellite Node is dramatically higher than the other nodes. The cohesion and coupling of that Node is much lower than the other Nodes. The Satellite Node is shown in Fig. 7.40. The Satellite Node has high Adaptability because it has low Coupling and low Cohesion. Changes to this Node will result in less impact on the SOSI than changes to the Tarawa Node that has lower Adaptability.

The SOSI results also show the impact of the highly reused Elements on Adaptability. The P2P SOSI group has three Elements that are used by all four SOSIC. The Nodes with lowest Adaptability in P2P_1 and P2P_2, Tarawa and Harper's Ferry Node, respectively, contain one or more of the Elements that are highly reused. The highly reused Elements are spread over more Nodes in P2P_2 so the Adaptability scores on each Node are relatively higher than the Nodes of the other P2P alternatives. This reinforces the claim that Degree of Reuse affects the overall Agility of the SOSI.

Figure 7.42 shows the Adaptability results for the P2P SOSI Architecture. P2P_1 and P2P_3 were partitioned by echelon. The Adaptability scores show that the grouping by echelon is not as Adaptable as the grouping by function, P2P_2. This is illustrated by the higher overall Adaptability of P2P_2 when compared to P2P_1 and P2P_3.



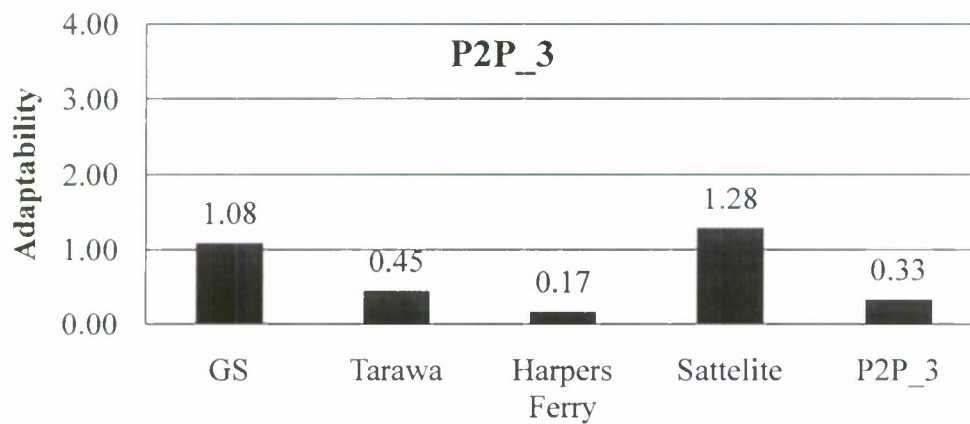
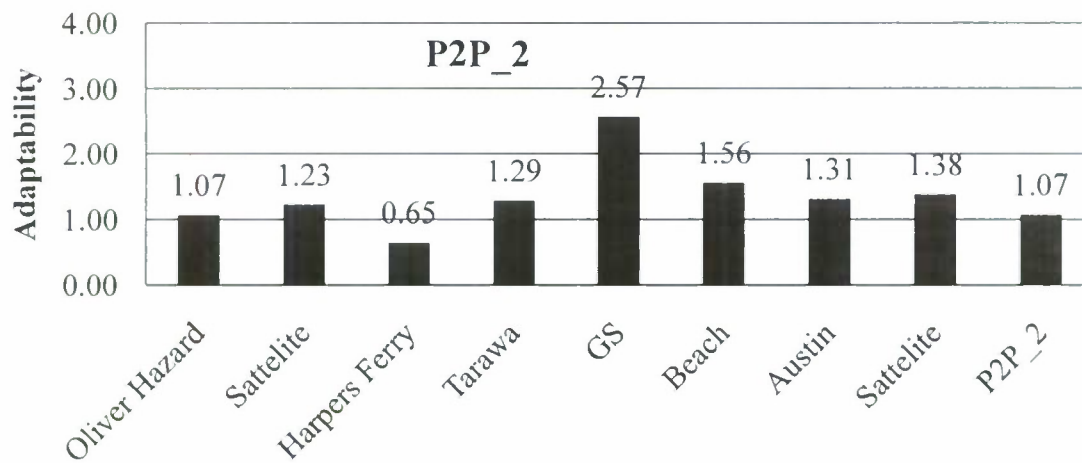


Figure 7.41. P2P SOSI Architecture Results

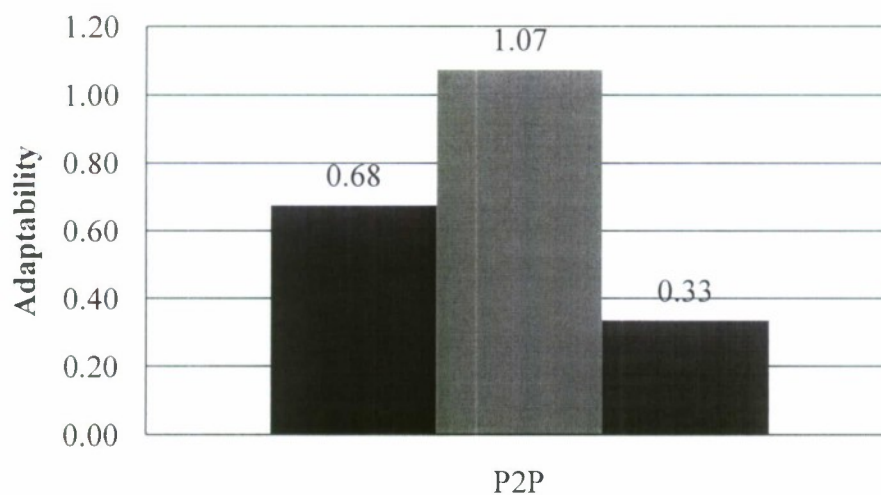


Figure 7.42. P2P SOSI Group Adaptability

7.9 CS Results

This section discusses the results for the CS SOSI Architecture. The CS SOSI Architecture groups use server Elements to facilitate communication. CS1 adds a single Server Element, while CS2 and CS3 add two and three Server Elements, respectively.

Figures 7.43, 7.44, and 7.45 show a sample of the for each SOSI group: CS1, CS2 and CS3, respectively. The SOSI groups show the effects of high cohesion and excessive coupling. The Nodes with very low Adaptability in the diagrams are the Nodes that contain Server Elements. The Server Elements have a high Degree of Reuse and are very interconnected with the other Elements in the Node which increases Cohesion. The Nodes with servers are highly coupled because all the Elements in the SOSI are connected to the server Elements which increases the number of Messages sent between the Node with the server and the other Nodes in the SOSI. Nodes with higher Adaptability have fairly low cohesion thus the increased Adaptability. This shows that the changes to any other Node than the one with the Server will result in low impact of change on the SOSI as a whole. But, if the Server Element is moved to another Node or otherwise incapacitated then the impact of the change on the SOSI would be significant. The aggregate measure of Adaptability is low for each CS SOSI Group alternative. The results for CS2 and CS3 also show the impact of the Server Elements.

For illustrative purposes, Figures 7.46 and 7.47 show the CPN for Tarawa and Harpers Ferry Nodes. The Tarawa Node is an example of a high cohesion node and the Harper's Ferry Node is an example of a low cohesion Node. The Tarawa Node is an example of the cohesion that is present when a Server Element is a member of a Node. The Harper's Ferry Node is an example of Nodes with server Elements in the CS alternatives. Notice there is little communication directly between Elements on the Harper's Ferry Node because all the communication between Elements is brokered by the server Element. The Elements on the Harper's Ferry Node could be easily moved to other Nodes with much impact on the SOSI. Changing the Node with the Server would cause significant impact on the SOSI which is illustrated by the very low Adaptability of the Tarawa Node.

Figure 7.48 Shows the Adaptability results for all CS alternatives. Adaptability assessment is better for the alternatives with the most server Elements. This shows the effects of reduces reuse and reduced coupling because the SOSIs with more server Elements have lower Node Coupling than the SOSIs with only one server. However, the Cohesion on the Nodes with servers is still very high and reduces the overall Adaptability of the alternatives.

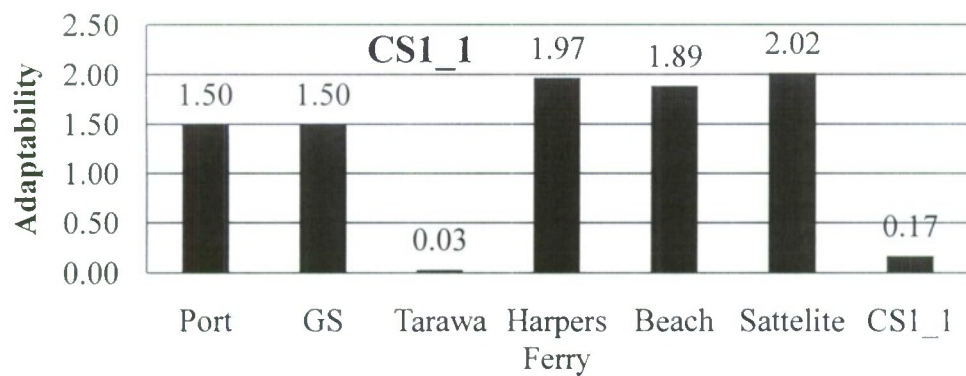


Figure 7.43. CS1 SOSI Group Adaptability

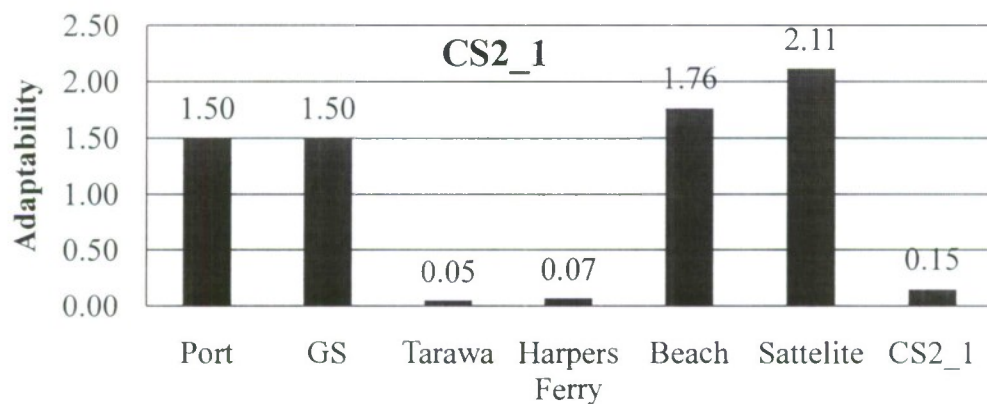


Figure 7.44. CS2 SOSI Group Adaptability

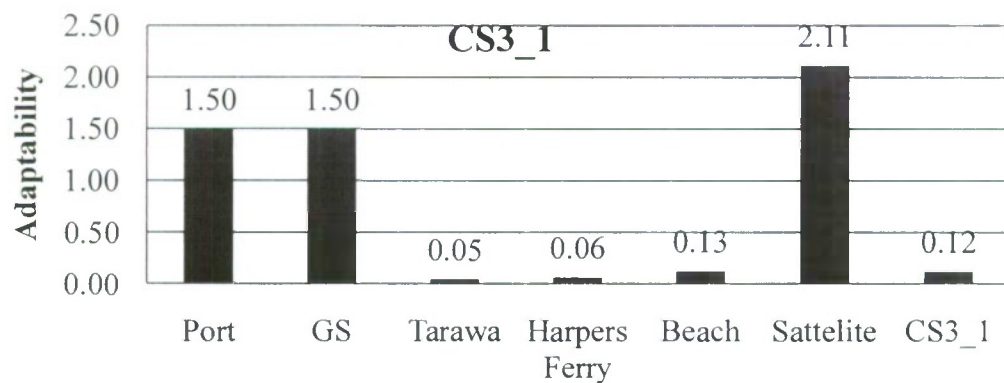


Figure 7.45. CS3 SOSI Group Adaptability

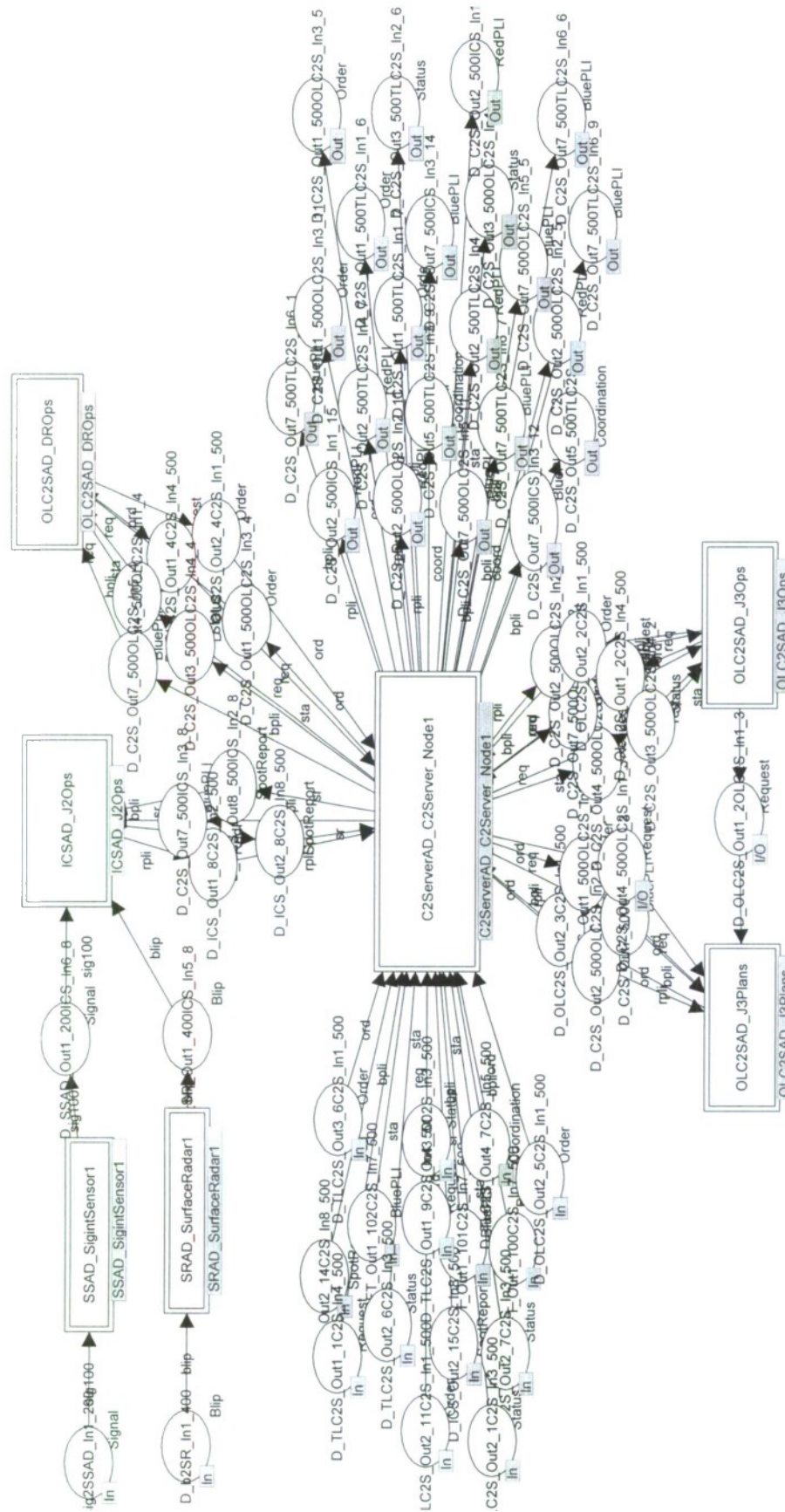


Figure 7.46. High Cohesion Node Example CS Architecture

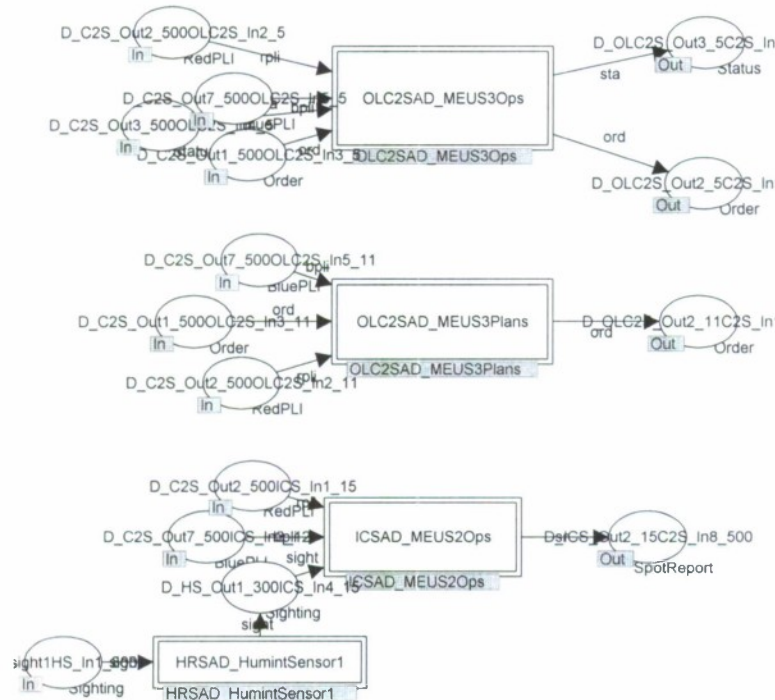


Figure 7.47. CS Low Cohesion Node

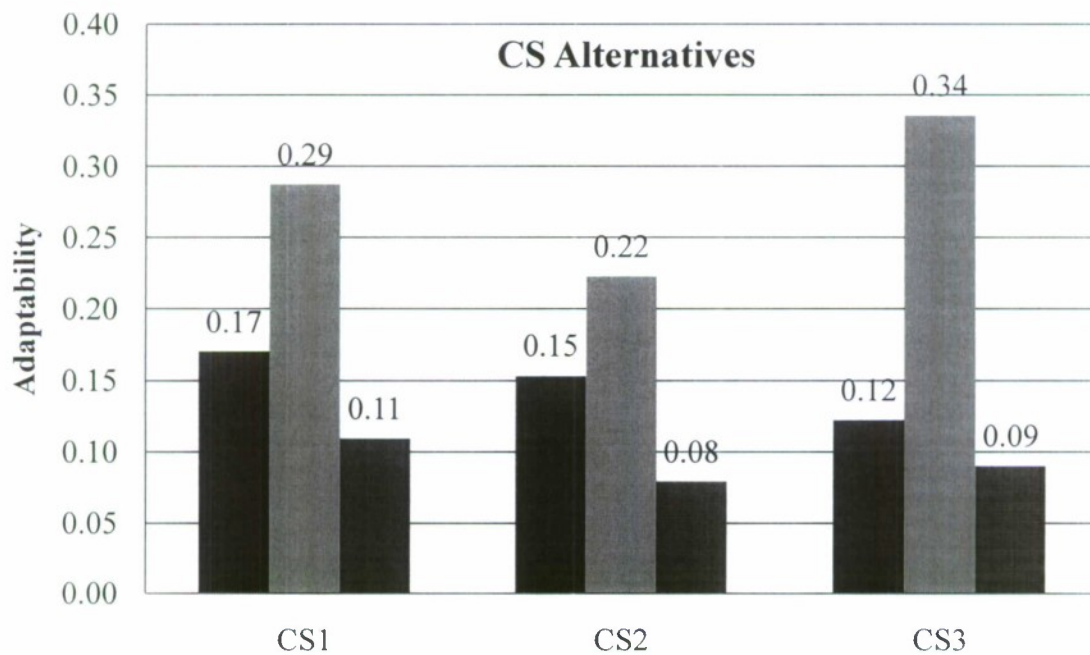


Figure 7.48. CS SOSI Architecture Adaptability

7.10 SOA Results

The SOA SOSI Architecture uses services to facilitate the communication between Elements and accomplish each SOSIC. Each SOA SOSI group has different number instances of each of five services: BFTService, ISRService, PlanningService, RequestService and CoordinationService.

The Adaptability results in Figures 7.49, 7.50, and 7.51 are all examples from each SOA SOSI group. The Adaptability is lowest on the Nodes that contain the Service Elements. Adaptability increases as the number of service Elements increases. This is because the Cohesion of the Nodes is going down as the number of Service Elements increase because the number of connections to each service decreases as the number of Service elements increases.

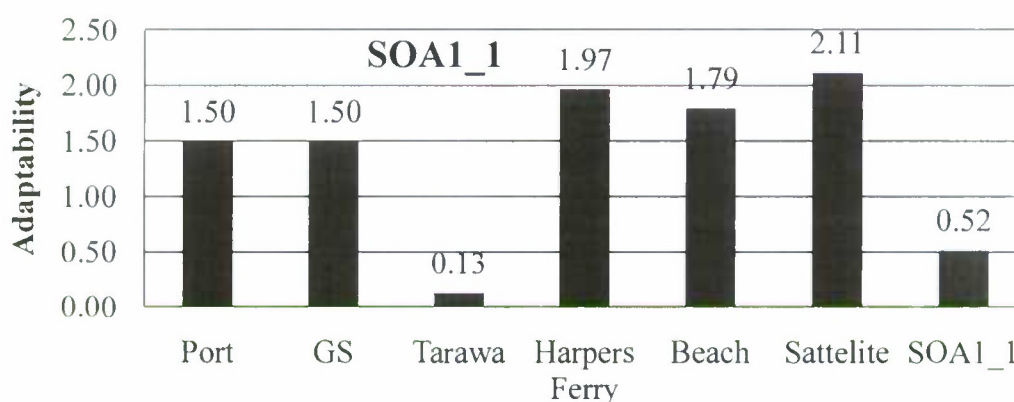


Figure 7.49. SOA1_1 SOSI Adaptability

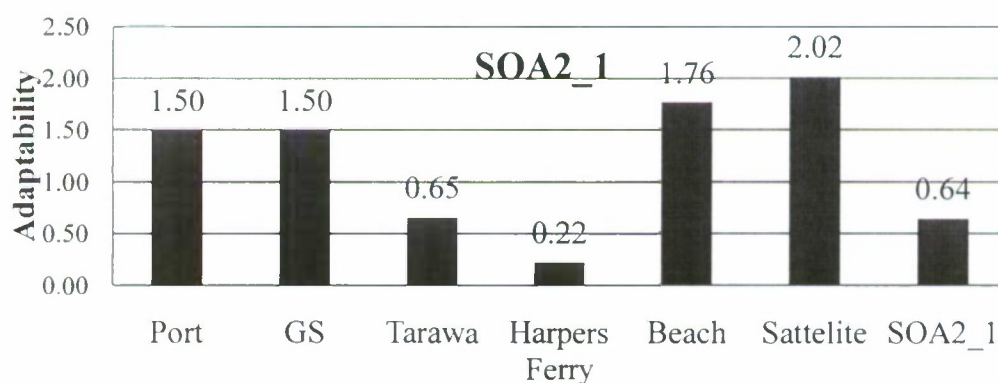


Figure 7.50. SOA2_1 SOSI Adaptability

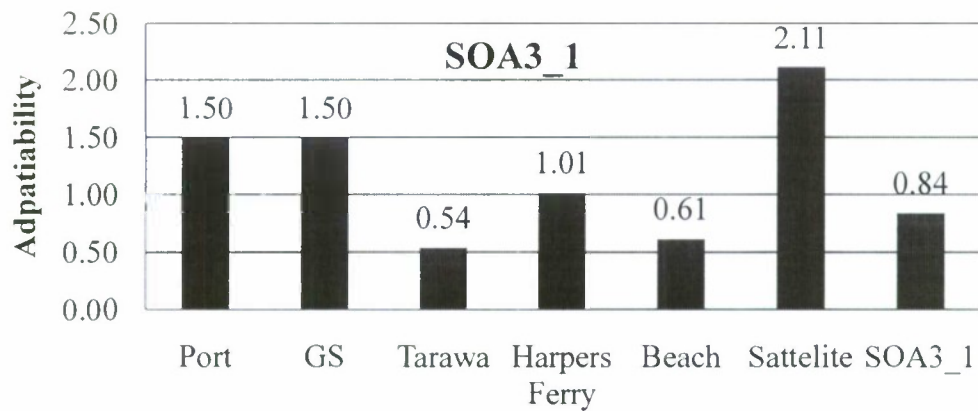


Figure 7.51. SOA3_1 SOSI Adaptability

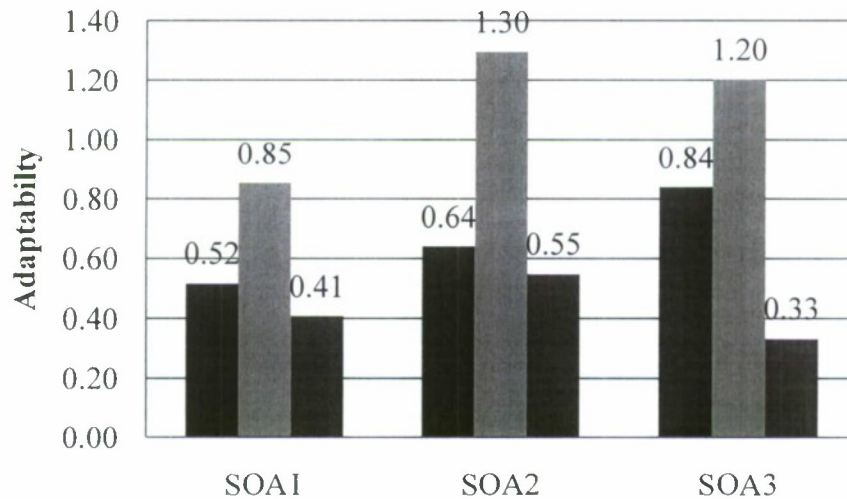


Figure 7.52. SOA SOSI Architecture Adaptability

7.11 Overall Results

This section compares the assessed Adaptability and Agility of the SOSI groups. Figure 7.54 shows Adaptability for all the SOSI alternatives. The results for CS are clearly lower than the P2P and SOA alternatives.

Figure 7.55 shows the Exclusiveness results for each SOSI group. The Degree of reuse is higher among the CS alternatives because the servers are used by every SOSIC. Degree of Reuse is lower for the SOA alternatives because the services are members of at most two SOSICs which is less than the Degree of Reuse for the server Elements which is four.

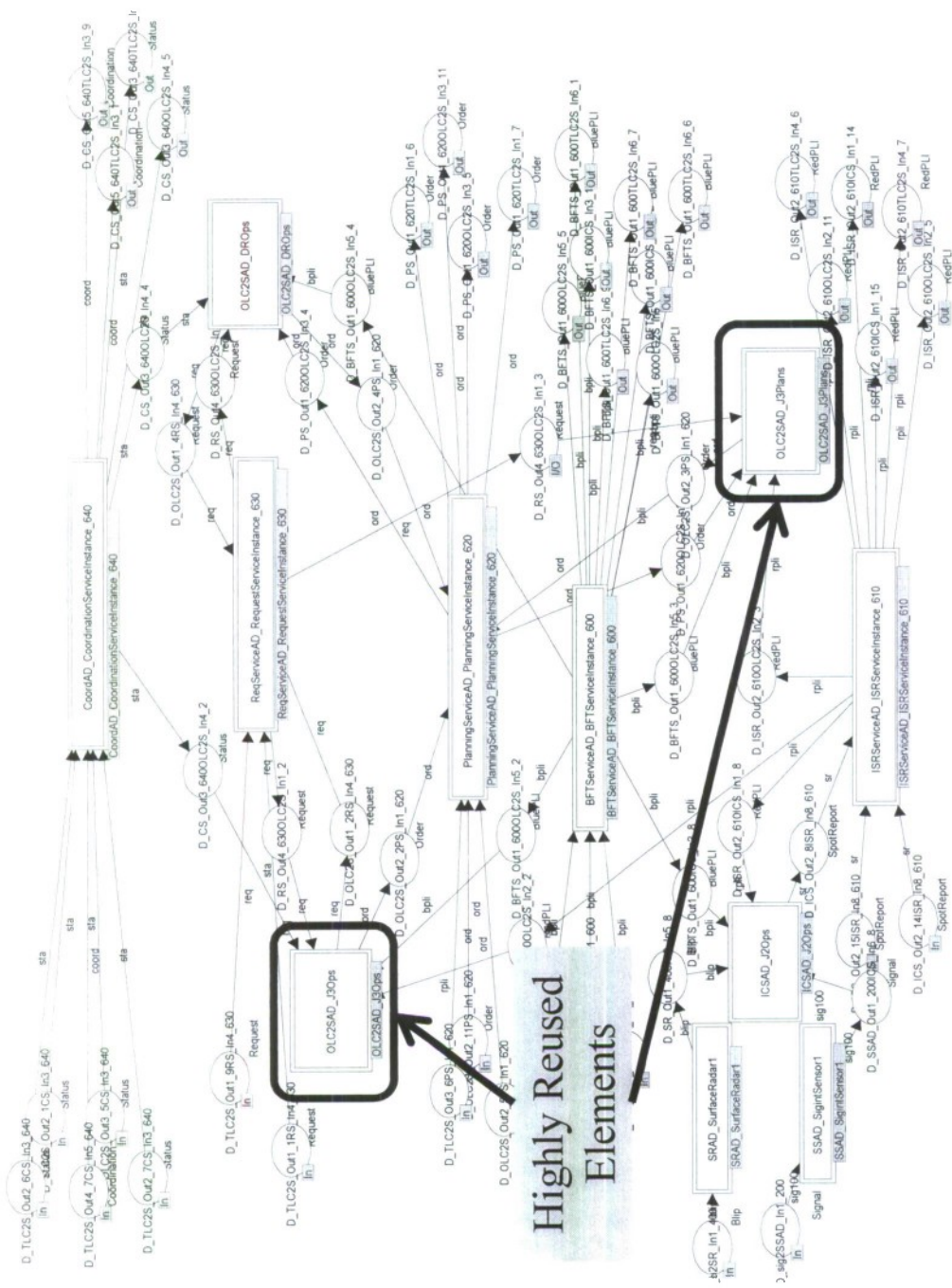


Figure 7.53 Impact of Highly Reused Elements

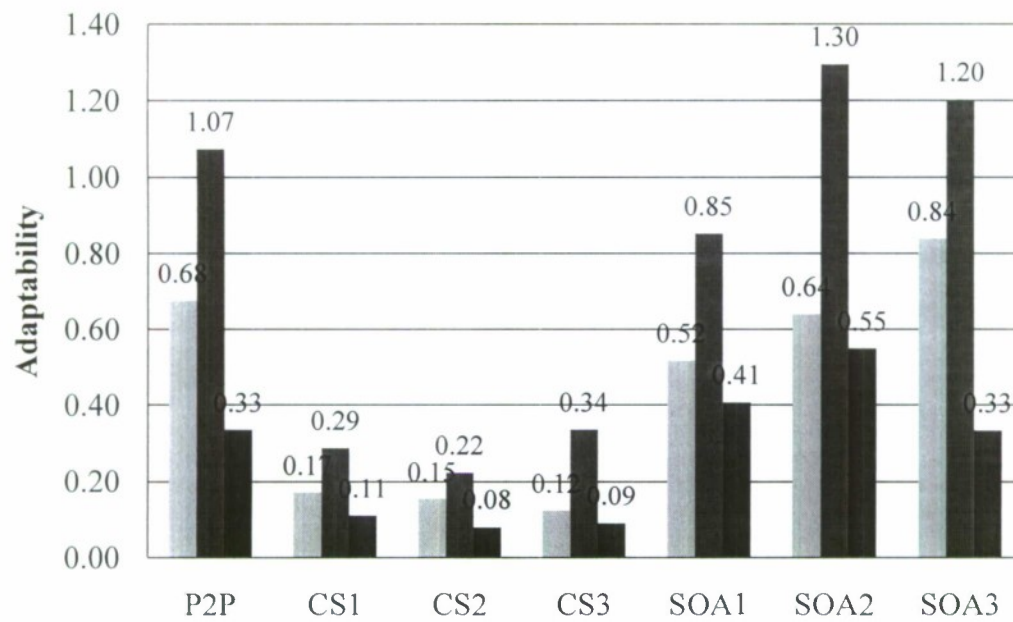


Figure 7.54. Case Study Adaptability Results

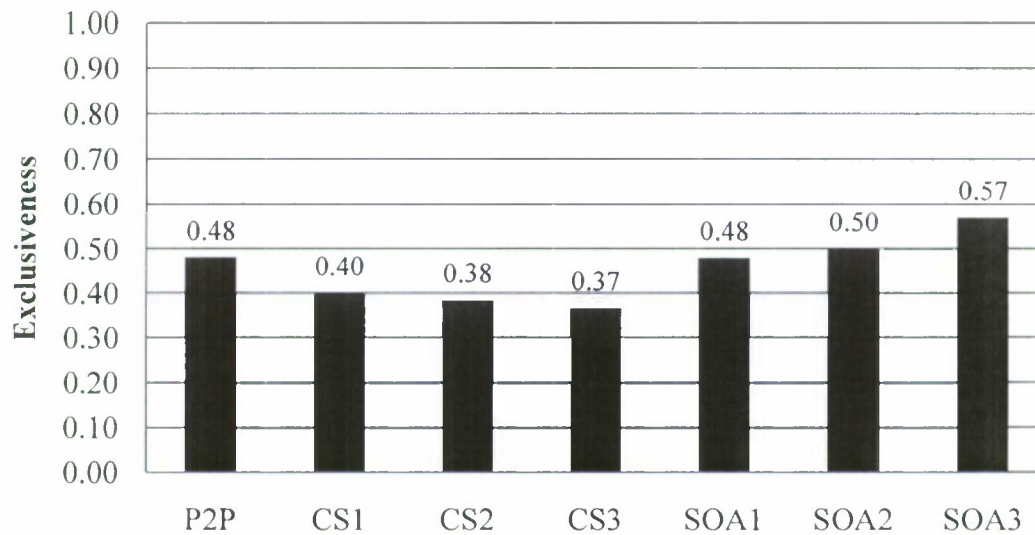


Figure 7.55 Case Study Exclusiveness Results

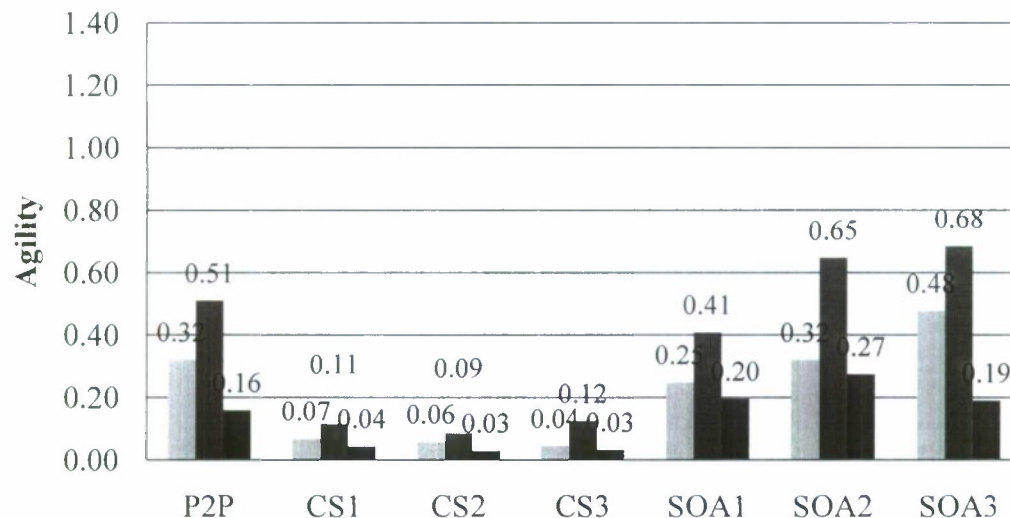


Figure 7.56. Case Study Agility Results

Exclusiveness for the P2P alternatives is driven by the end Elements that are members of each SOSIC. There are three Elements that are members of all four SOSIC. This reduces the Exclusiveness of the SOSI and reduces the Agility of the SOSI.

Agility is the final assessment measure and the results show that the P2P and SOA alternatives assess higher for Agility than the CS alternatives. The results show that low Exclusiveness, especially in the CS alternatives, reduces the overall Agility of the SOSI alternatives.

Figure 7.53 has the high reuse Elements circled. Notice how these Elements connect four of the Service Elements. The highly reused Elements participate in all the SOSICs, therefore they are directly connected to all but one of the Service Elements and indirectly connected to the other. This situation causes the relationship of the inputs and outputs to increase thus increasing cohesion and reducing the Adaptability of the Node. While low compared to the other Nodes, the level of Adaptability on the nodes with Services is still significantly higher than their CS alternative counterparts. This is a result of the ability of the Services to be distributed across more Nodes where a Server can occupy only one Node but may accomplish many of the tasks in a single Element that may be accomplished by multiple Service Elements. This reduces the level of the Cohesion between Elements on the Node and the level of Coupling between the Nodes; the end result being higher Adaptability.

7.12 Conclusions

Step 7 presents the case study results. The ESG operates with a high level of uncertainty based on the multiple missions that it must be able to accomplish and the diverse operating environments that it expected to operate in. The assessment is used to illustrate the effects of

architecture decisions on the Adaptability and Agility of the ESG to increase the confidence of the ESG commander that the SOSI that supports his organization can adapt to unpredicted operating environments.

The assessment reveals to developers that the P2P and SOA alternatives are more adaptable than the CS alternatives. The CS alternatives have low Adaptability because they contain Nodes that have extremely high coupling and cohesion compared to the other Nodes in the SOSI alternatives. The primary reason for the low Adaptability is the server Elements increase the number of highly reused Elements. The server Elements and the highly reused Elements cause extremely high coupling and cohesion on the Nodes they are assigned to dramatically reducing the Adaptability of the SOSI alternative. Fig. 7.57 summarizes the results of the case study. Based on the assessment, SOA and P2P have similar Adaptability measures but for different reasons. The SOA alternatives showed higher coupling than the P2P alternatives. This result is surprising because the conventional wisdom is that SOA implementations will have lower coupling. The Coupling measure identifies data dependence; therefore the SOA paradigm may reduce the dependence of an Element on a particular instance of a service but not the dependence on the data generated by the service.

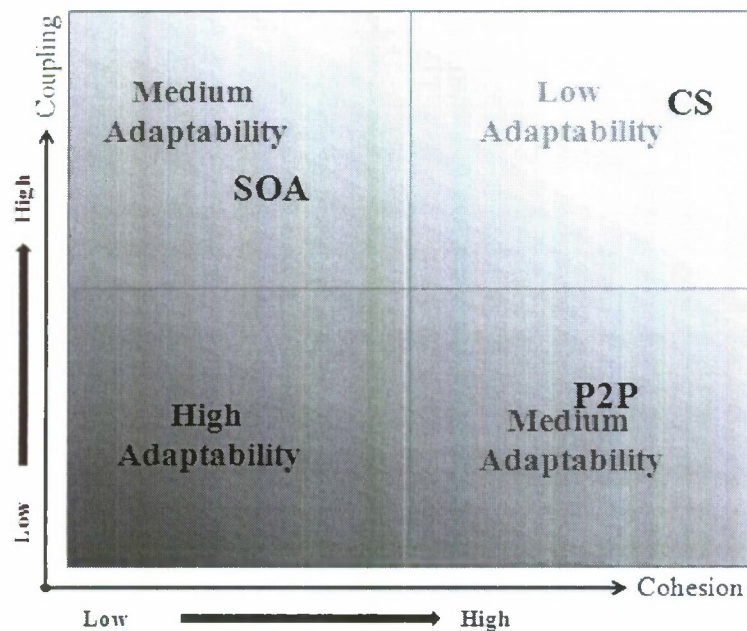


Figure 7.57. Summary Graphic of Case Study Results

Additionally, the P2P alternatives had higher Cohesion than the SOA alternatives. This is because the services in the SOA alternatives diffused the interaction between the highly reused Elements and the other Elements in SOSI. In the P2P alternatives, the highly reused elements cause an increase in the number of paths through the Node because the highly reused Elements are connected to more Elements than in the SOA alternatives. Furthermore, both SOA and P2P

offer higher Adaptability than the CS alternatives because the CS alternatives displayed much higher Coupling and Cohesion than the P2P or SOS alternatives because the server Elements were highly reused and connected to every other Element on the Node. This made every Element in the SOSI dependent on a Server Element for its data. Finally, all the SOSI alternatives possessed highly reused Elements that reduced the Exclusiveness measure and had a corresponding effect on Agility for all the alternatives. The SOA alternatives had the best Exclusiveness measure because the total number of Elements is increased by the number of service Elements.

