ON THE CHARACTERIZATION AND ANALYSIS OF SYSTEM OF SYSTEMS ARCHITECTURES

by

Stewart Whitfield Liles

A Dissertation
Submitted to the
Graduate Faculty
of
George Mason University
In Partial fulfillment of
The Requirements for the Degree
of
Doctor of Philosophy
Information Technology

Committee:

Dr. Alexander H. Levis, Dissertation Director

Dr. J. Mark Pullen, Committee Chairman

Dr. Lee E. Wagenhals, Committee Member

Dr. Andrew G. Loerch, Committee Member

Dr. Daniel Menascé, Senior Associate Dean for Research and Graduate Studies

Dr. Lloyd J. Griffiths, Dean, The Volgenau School of Information Technology and Engineering

Date: August 20, 2008

Fall Semester 2008
George Mason University
Fairfax, VA 22030
On the Characterization and Analysis of System of Systems Architectures

A dissertation submitted in partial fulfillment of the requirements for the degree of Doctor of Philosophy at George Mason University

By

Stewart Whitfield. Liles
Master of Science
Naval Post-graduate School, 1998

Director: Alexander H. Levis
Department of Electrical and Computer Engineering

Fall Semester 2008
George Mason University
Fairfax, VA
DEDICATION

This dissertation is dedicated to all the people who have helped me in the effort. Of course my wife, Carol and daughter, Haley for their support during the high tempo work periods. I also dedicate the work to my fellow soldiers and officers who have deployed, some multiple times, while I have been taking courses and sleeping every night in my own bed. Their sacrifices and their families’ are recognized by me and my committee. Finally, this work is dedicated to my parents who have supported and nurtured me physically and intellectually my entire life.
ACKNOWLEDGEMENTS

My advisors, of course, who supported me throughout the process. Dr. Alex Levis provided encouragement and advice. His lectures on architecture based system integration provided the germ for the ideas explored in the research. Dr. Mark Pullen provided support with work space and computing resources. His support prior to and during the coursework phase is greatly appreciated. Dr. Lee Wagenhals provided excellent comments late in the process and I could not have completed the work without his support. Dr. Andy Loerch listened when I had questions and provided excellent counsel concerning the PhD process, in general, and my research, specifically. Dr. Jon Whittle provided much counsel for the model transformation aspects of the work. Dr. David Alberts generously provided me time to discuss the project and provided excellent advice that guided the development of the methodology and performance measures. Additionally, I acknowledge LTC Tina Hartley for her friendship and support, especially with math tutoring. Finally, I also acknowledge the U.S. Army and its Advance Civil Schooling program. I would not have been able to complete a doctorate without their support of time and resources.
# TABLE OF CONTENTS

List of Tables ........................................................................................................ viii
List of Figures ...................................................................................................... x
Abstract .............................................................................................................. xiv

## CHAPTER 1 INTRODUCTION ............................................................................ 1
  1.1 Background .................................................................................................. 1
  1.2 Motivation .................................................................................................. 2
    1.2.1 System of Systems ............................................................................. 3
    1.2.2 Architecture Modeling ................................................................. 5
  1.3 Research Goals .......................................................................................... 5
    1.3.1 Problem and Thesis Statements ..................................................... 6
    1.3.2 Hypothesis ...................................................................................... 6
    1.3.3 Contributions ................................................................................. 6
  1.4 Document Organization .............................................................................. 7

## CHAPTER 2 RELATED WORK ........................................................................... 8
  2.1 System of Systems ..................................................................................... 8
    2.1.1 System of Systems Taxonomy ......................................................... 10
  2.2 Architecture Modeling .............................................................................. 12
    2.2.1 Department of Defense Architecture Framework ......................... 13
    2.2.2 Unified Modeling Language .......................................................... 19
    2.2.3 Model Driven Development ............................................................ 23
    2.2.4 Executable Modeling Languages ................................................... 26
    2.2.5 UML to Colored Petri Net (CPN) Transformation ......................... 30
LIST OF TABLES

Table 2.1. System of Systems (SOS) Characteristics [Maier, 1996] ................................. 9
Table 2.2. Taxonomy for Describing a System of Systems [DeLaurentis, 2005] ....... 11
Table 2.3. All View Products [DODAF, 2007b] ............................................................. 15
Table 2.4. Operational View Products [DODAF, 2007b] ............................................... 16
Table 2.5. System View Products [DODAF, 2007b] .................................................... 17
Table 2.6. Technical Standards View Products [DODAF, 2007b] ............................... 18
Table 3.1. Cohesion Example .......................................................................................... 67
Table 3.2. Degree of Reuse Example Data ................................................................. 70
Table 3.3. Degree of Reuse Calculation for SOS f .................................................... 70
Table 4.1. Operational Information Exchange Matrix, OV-3 ...................................... 84
Table 4.2. Operational Rules Model, OV-6a ............................................................... 84
Table 4.3. Systems Data Exchange Matrix, SV-6 ....................................................... 89
Table 4.4. Operational Activity to Systems Function Traceability Matrices, SV-5s .... 91
Table 4.5. Systems Rules Models, SV-10a’s ............................................................... 91
Table 4.6. SOSI Architecture SV-6 ............................................................................. 94
Table 4.7. SOSI Architecture Rules Models, SV-10a .................................................. 96
Table 6.1. ESG Operational Information Exchange Matrix OV-3 ................................ 119
Table 6.2. ESG Operational Rules Model OV-6a ......................................................... 121
Table 6.3. Planning and Coordination SV-6 ................................................................. 128
Table 6.4. Blue Force Tracking SV-6 ............................................................................. 128
Table 6.5. Process and Disseminate Intelligence Information SV-6 .............................. 129
Table 6.6. Planning and Coordination SV-10a ............................................................ 129
Table 6.7. Blue force Tracking SV-10a ....................................................................... 130
Table 6.8. Process and Disseminate Intelligence Information SV-10a ....................... 130
Table 6.9. Planning and Coordination SV-5 ............................................................... 137
Table 6.10. Blue Force Tracking SV-5 ......................................................................... 137
Table 6.11. Process and Disseminate Intelligence Information SV-5 ....................... 138
Table 6.12. P2P Systems Data Exchange Matrix, SV-6 .............................................. 146
Table 6.13. P2P Systems Rule Model, SV-10a ............................................................. 147
Table 6.14. P2P SOSI Architecture SV-5 ............................................................... 153
Table 6.15. Cohesion SOSI P2P_1 ........................................................................ 158
Table 6.16. SOSI P2P_1 Coupling Results ............................................................ 159
Table 6.17. P2P Degree of Reuse Data ................................................................. 160
Table 6.18. P2P Degree of Reuse and Exclusiveness Results .............................. 160
<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Figure 1.1</td>
<td>System of Systems [Brown, 2005]</td>
<td>4</td>
</tr>
<tr>
<td>Figure 2.1</td>
<td>Graphical View of DeLaurentis Taxonomy</td>
<td>13</td>
</tr>
<tr>
<td>Figure 2.2</td>
<td>Department of Defense Architecture Registry System [DODAF, 2007]</td>
<td>18</td>
</tr>
<tr>
<td>Figure 2.3</td>
<td>Role of UML Common Core [OMG, 2007a]</td>
<td>21</td>
</tr>
<tr>
<td>Figure 2.4</td>
<td>UML-MOF Meta-Levels [OMG, 2007b]</td>
<td>22</td>
</tr>
<tr>
<td>Figure 2.5</td>
<td>Four Layer Meta-Model Hierarchy [OMG, 2007b]</td>
<td>23</td>
</tr>
<tr>
<td>Figure 2.6</td>
<td>Model Driven Architecture Fundamental Concept [OMG, 2007b]</td>
<td>26</td>
</tr>
<tr>
<td>Figure 3.1</td>
<td>SOSI Taxonomy</td>
<td>37</td>
</tr>
<tr>
<td>Figure 3.2</td>
<td>System of Systems (SOS) Boundary</td>
<td>41</td>
</tr>
<tr>
<td>Figure 3.3</td>
<td>SOS Taxonomy Relationships Venn Diagram</td>
<td>44</td>
</tr>
<tr>
<td>Figure 3.4</td>
<td>Simple Information Flow Path</td>
<td>48</td>
</tr>
<tr>
<td>Figure 3.5</td>
<td>Cohesion Example</td>
<td>49</td>
</tr>
<tr>
<td>Figure 3.6</td>
<td>Node with Loops</td>
<td>50</td>
</tr>
<tr>
<td>Figure 3.7</td>
<td>Cobb-Douglas Production Function</td>
<td>55</td>
</tr>
<tr>
<td>Figure 3.8</td>
<td>Adaptability Plot</td>
<td>56</td>
</tr>
<tr>
<td>Figure 3.9</td>
<td>Adaptability Contour</td>
<td>56</td>
</tr>
<tr>
<td>Figure 3.10</td>
<td>Contour Calculation</td>
<td>57</td>
</tr>
<tr>
<td>Figure 3.11</td>
<td>Effects of Elasticity Constant on Adaptability</td>
<td>58</td>
</tr>
<tr>
<td>Figure 3.12</td>
<td>Relationship of Cohesion and Coupling to Adaptability</td>
<td>59</td>
</tr>
<tr>
<td>Figure 3.13</td>
<td>Agility Plot</td>
<td>62</td>
</tr>
<tr>
<td>Figure 3.14</td>
<td>Working Example for Adaptability Calculations with Nodes</td>
<td>63</td>
</tr>
<tr>
<td>Figure 3.15</td>
<td>CPN for n_{1,1}</td>
<td>65</td>
</tr>
<tr>
<td>Figure 3.16</td>
<td>Colored Petri Net (CPN) for n_{1,2}</td>
<td>66</td>
</tr>
<tr>
<td>Figure 3.17</td>
<td>Colored Petri Net (CPN) for n_{1,3}</td>
<td>66</td>
</tr>
<tr>
<td>Figure 3.18</td>
<td>Coupling Results for SOSI f_1</td>
<td>67</td>
</tr>
<tr>
<td>Figure 3.19</td>
<td>Adaptability Results for SOSI f_1</td>
<td>68</td>
</tr>
<tr>
<td>Figure 3.20</td>
<td>SOSI f_1 with Three SOSIC</td>
<td>69</td>
</tr>
<tr>
<td>Figure 3.21</td>
<td>Example Results for Adaptability and Agility</td>
<td>71</td>
</tr>
<tr>
<td>Figure 4.1</td>
<td>Methodology</td>
<td>75</td>
</tr>
</tbody>
</table>
Figure 6.17. Process and Disseminate Intelligence information SV-11 ....................... 132
Figure 6.18. Planning and Coordination SV-4 ............................................................. 133
Figure 6.19. Blue Force Tracking SV-4 ................................................................. 134
Figure 6.20. Process and Disseminate Intelligence Information SV-4 ....................... 135
Figure 6.21. Tactical Level Command and Control System Activity Diagram .......... 136
Figure 6.22. Peer-to-Peer Architecture ...................................................................... 139
Figure 6.23. Client Server SOSI with One Server .................................................... 140
Figure 6.24. Service Oriented Architecture with One Instance of Each Service ......... 141
Figure 6.25. P2P SOSI Architecture SV-11, part 1 .................................................... 142
Figure 6.26. P2P_1 SV-1 Six Nodes ........................................................................ 143
Figure 6.27. P2P_2 SV-1 Eight Nodes ...................................................................... 144
Figure 6.28. P2P_3 SV-1 Four Nodes ...................................................................... 145
Figure 6.29. P2P SOSI Architecture SV-11 ............................................................ 148
Figure 6.30. P2P SOSI Architecture Process and Disseminate Intelligence Information SOSIC ................................................................. 149
Figure 6.31. Conduct Support Operations SOSIC ................................................... 150
Figure 6.32. Blue Force Tracking SOSIC ................................................................ 151
Figure 6.33. Conduct Security Operations SOSIC ................................................... 152
Figure 6.34. P2P_1 SOSI Alternative Top Level CPN Representation ....................... 155
Figure 6.35. P2P_1 Port Node CPN ........................................................................ 156
Figure 6.36. P2P_1 SOSI Ground Station Node CPN .............................................. 156
Figure 6.37. P2P_1 SOSI Tarawa Node Colored Petri Net ....................................... 156
Figure 6.38. P2P_1 SOSI Harper's Ferry Node CPN .............................................. 157
Figure 6.39. P2P_1 SOSI Beach Node CPN ............................................................. 157
Figure 6.40. P2P_1 SOSI Satellite Node CPN ......................................................... 158
Figure 6.41. P2P SOSI Architecture Results .......................................................... 163
Figure 6.42. P2P SOSI Group Adaptability ............................................................. 164
Figure 6.43. CS1 SOSI Group Adaptability .............................................................. 166
Figure 6.44. CS2 SOSI Group Adaptability .............................................................. 166
Figure 6.45. CS3 SOSI Group Adaptability .............................................................. 166
Figure 6.46. High Cohesion Node Example CS Architecture ................................... 167
Figure 6.47. CS Low Cohesion Node ...................................................................... 168
Figure 6.48. CS SOSI Architecture Adaptability ..................................................... 168
Figure 6.49. SOA1_1 SOSI Adaptability ................................................................. 171
Figure 6.50. SOA2_1 SOSI Adaptability ................................................................. 171
Figure 6.51. SOA3_1 SOSI Adaptability ................................................................. 172
Figure 6.52. SOA SOSI Architecture Adaptability .................................................. 172
Figure 6.53. Impact of Highly Reused Elements ..................................................... 173
Figure 6.54. Case Study Adaptability Results ................................................... 174
Figure 6.55. Case Study Exclusiveness Results ................................................... 174
Figure 6.56. Case Study Agility Results ......................................................... 175
Figure 6.57. Summary Graphic of Case Study Results ....................................... 176
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 should be assessed for their ability to deploy in more than one configuration. Past architecture assessments and performance characteristics are primarily system focused and do not address the dynamics of the interacting constituent systems of the SOS. This dissertation provides two measures, Adaptability and Agility,
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.

A case study is presented to illustrate the application of the assessment methodology and 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.
CHAPTER 1
INTRODUCTION

1.1 BACKGROUND

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. It focuses on 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.

1.2 MOTIVATION

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 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.

### 1.2.1 System of Systems

Figure 1.1 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 Chapter Three.

Figure 1.1. System of Systems [Brown, 2005]
1.2.2 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 architecture views describe organizational roles that interact to provide a particular capability. System architecture views 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.

1.3 RESEARCH GOALS

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.

1.3.1 Problem and Thesis Statements

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.

The corresponding thesis statement is:

A method can be developed to analyze and evaluate characteristics that describe unique qualities of a system of systems.

1.3.2 Hypothesis

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

1.3.3 Contributions

There are three primary contributions of this research:

- Analytical measures that describe unique technical aspects of the system of systems.
Methodology for combining behavior models to create an executable model for analysis.

SOS architecture assessment of SOS characteristics given the analytical measures and the methodology.

1.4 DOCUMENT ORGANIZATION

This dissertation is organized around the technologies required to implement the concepts described above. Chapter Two presents related research in the primary domains that were used to address the problem. Chapter Three presents the SOS characteristics that were developed to assess the SOS architecture alternatives. Chapter Four presents the methodology used to develop and assess the SOS architecture alternatives using the measures described in Chapter Three. Chapter Five presents the process for transforming the static architecture representation into a dynamic representation for analysis. Chapter Six details a case study that provides evidence concerning the validity of the assessment methodology, SOS characteristics, and the executable models used to evaluate them. Chapter Seven concludes with the contributions of this thesis and ideas for future research concerning SOS architecture development and analysis.
CHAPTER 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.

2.1 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 Table 2.1). 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 for the purpose of this research is offered in Chapter 3.

Table 2.1. System of Systems (SOS) Characteristics [Maier, 1996]

<table>
<thead>
<tr>
<th>Characteristics</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Operational Independence of the Elements</strong></td>
<td>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.</td>
</tr>
<tr>
<td><strong>Managerial Independence of the Elements</strong></td>
<td>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.</td>
</tr>
<tr>
<td><strong>Evolutionary Development</strong></td>
<td>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.</td>
</tr>
<tr>
<td><strong>Emergent Behavior</strong></td>
<td>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.</td>
</tr>
<tr>
<td><strong>Geographic Distribution</strong></td>
<td>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.</td>
</tr>
</tbody>
</table>
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 in 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].

2.1.1 System of Systems 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 2.2.

### Table 2.2. Taxonomy for Describing a System of Systems [DeLaurentis, 2005]

<table>
<thead>
<tr>
<th>Category</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Resources</td>
<td>The entities (systems) that give physical manifestation to the system-of-systems</td>
</tr>
<tr>
<td>Stakeholders</td>
<td>The non-physical entities that give intent to the SOS operation through values</td>
</tr>
<tr>
<td>Operations</td>
<td>The application of intent to direct the activity of physical and non-physical entities</td>
</tr>
<tr>
<td>Policies</td>
<td>The external forcing functions that impact the operation of physical and non-physical entities</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Level</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Alpha (α)</td>
<td>Base level of entities in each category, further decomposition will not take place.</td>
</tr>
<tr>
<td>Beta (β)</td>
<td>Collections of α-level systems (across categories), organized in a network.</td>
</tr>
<tr>
<td>Gamma (γ)</td>
<td>Collections of β-level systems (across categories), organized in a network.</td>
</tr>
<tr>
<td>Delta (δ)</td>
<td>Collections of γ-level systems (across categories), organized in a network.</td>
</tr>
</tbody>
</table>

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 (Figure 2.1), 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.

2.2 ARCHITECTURE MODELING

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.

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.

Figure 2.1. Graphical View of DeLaurentis Taxonomy

2.2.1 Department of Defense Architecture Framework

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 (FOSs), 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 View primarily addresses the operational nodes and the data that must pass between them for operational success. The Systems View addresses the specific physical systems that support the exchange of information between operational nodes. The Technical Standards View describes 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. Table 2.3, Table 2.5, and Table 2.6 show the various DODAF Architecture View products: All View, Operational View, Systems View, and Technical Standards View.

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 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.

Table 2.3. All View Products [DODAF, 2007b]

<table>
<thead>
<tr>
<th>Product</th>
<th>Framework Product Name</th>
<th>General Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AV-1</td>
<td>Overview and Summary Information</td>
<td>Scope, purpose, intended users, environment depicted analytical findings</td>
</tr>
<tr>
<td>AV-1</td>
<td>Integrated Dictionary</td>
<td>Architecture data repository with definitions of all terms used in all products</td>
</tr>
</tbody>
</table>
### Table 2.4. Operational View Products [DODAF, 2007b]

<table>
<thead>
<tr>
<th>Product</th>
<th>Framework Product Name</th>
<th>General Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>OV-1</td>
<td>High-Level Operational Concept Graphic</td>
<td>High-level graphical/textual description of operational concept</td>
</tr>
<tr>
<td>OV-2</td>
<td>Operational Node Connectivity Description</td>
<td>Operational nodes, connectivity, and information exchange need lines between nodes</td>
</tr>
<tr>
<td>OV-3</td>
<td>Operational Information Exchange Matrix</td>
<td>Information exchanged between nodes and the relevant attributes of that exchange</td>
</tr>
<tr>
<td>OV-4</td>
<td>Organizational Relationships Chart</td>
<td>Organizational, role, or other relationships among organizations</td>
</tr>
<tr>
<td>OV-5</td>
<td>Operational Activity Model</td>
<td>Capabilities, operational activities, relationships among activities, inputs, and outputs; overlays can show cost, performing nodes, or other pertinent information</td>
</tr>
<tr>
<td>OV-6a</td>
<td>Operational Rules Model</td>
<td>One of three products used to describe operational activity—identifies business rules that constrain operation</td>
</tr>
<tr>
<td>OV-6b</td>
<td>Operational State Transition Description</td>
<td>One of three products used to describe operational activity—identifies business process responses to events</td>
</tr>
<tr>
<td>OV-6c</td>
<td>Operational Event-Trace Description</td>
<td>One of three products used to describe operational activity—identifies business process responses to events</td>
</tr>
<tr>
<td>OV-7</td>
<td>Logical Data Model</td>
<td>Documentation of the system data requirements and structural business process rules of the Operational View</td>
</tr>
<tr>
<td>Product</td>
<td>Framework Product Name</td>
<td>General Description</td>
</tr>
<tr>
<td>---------</td>
<td>------------------------</td>
<td>---------------------</td>
</tr>
<tr>
<td>SV-1</td>
<td>Systems Interface Description</td>
<td>Identification of systems nodes, systems, system items, and their interconnections.</td>
</tr>
<tr>
<td>SV-2</td>
<td>Systems Communications Description</td>
<td>Systems nodes, systems, system items, and their related communications.</td>
</tr>
<tr>
<td>SV-3</td>
<td>Systems-Systems Matrix</td>
<td>Relationships among systems in a given architecture.</td>
</tr>
<tr>
<td>SV-4</td>
<td>Systems Functionality Description</td>
<td>Functions performed by systems and the data flows among system functions.</td>
</tr>
<tr>
<td>SV-5</td>
<td>Operational Activity to Systems Function Traceability Matrix</td>
<td>Mapping of system functions to operational activities.</td>
</tr>
<tr>
<td>SV-6</td>
<td>Systems Data Exchange Matrix</td>
<td>Details of system data elements being exchanged between systems and the attributes of that exchange.</td>
</tr>
<tr>
<td>SV-7</td>
<td>Systems Performance Parameters Matrix</td>
<td>Performance characteristics of SV elements for the appropriate time frame.</td>
</tr>
<tr>
<td>SV-8</td>
<td>Systems Evolution Description</td>
<td>Planned incremental steps toward migrating a suite of systems to a more efficient suite, or toward evolving a current system to a future version.</td>
</tr>
<tr>
<td>SV-9</td>
<td>Systems Technology Forecast</td>
<td>Emerging technologies and software/hardware products will affect future development of the architecture.</td>
</tr>
<tr>
<td>SV-10a</td>
<td>Systems Rules Model</td>
<td>Identifies constraints that are imposed on system functionality due to some aspect of system design.</td>
</tr>
<tr>
<td>SV-10b</td>
<td>Systems State Transition Description</td>
<td>Identifies responses of a system to events.</td>
</tr>
<tr>
<td>SV-10c</td>
<td>Systems Event-Trace Description</td>
<td>Identifies system specific refinements of critical sequences of events described in the Operational View.</td>
</tr>
<tr>
<td>SV-11</td>
<td>Physical Schema</td>
<td>Physical implementation of the Logical Data Model entities.</td>
</tr>
</tbody>
</table>
Table 2.6. Technical Standards View Products [DODAF, 2007b]

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

Figure 2.2. Department of Defense Architecture Registry System [DODAF, 2007]

EA = Enterprise Architecture  
COI = Community of Interest  
CADM = Core Architecture Data Model  
DDMS = Defense Discovery Metadata Specification
2.2.2 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 reusable 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 2.3 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 2.3. Role of UML Common Core [OMG, 2007a]

Figure 2.4 illustrates how the MOF is the meta-model used to describe artifacts in the UML. Figure 2.5 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. 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 2.4. UML-MOF Meta-Levels [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.

2.2.3 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 2.6 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.
2.2.4 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 sets 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 though 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.

2.2.5 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.

2.3 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] analyze 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 Chapter 3.

2.4 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 was 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 points out that without a 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 representation must be transformed into the executable form. A CPN is the chosen executable form in 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.
CHAPTER 3
ASSESSMENT MEASURES

This chapter 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.

3.1 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 3.1 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.

![SOSI Taxonomy Diagram]

**Figure 3.1.** SOSI Taxonomy

The next section discusses the properties that differentiate the SOSI from the SOSI.
3.1.1 System of System 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 3.1 is the formal description of the SOS. The SOS \( \Omega \) 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 3.1:
\[
\Omega = \{E, F, M\}
\]
where,
\[
E = \{e_1, e_2, e_3, ..., e_v\}; \text{set of Elements that compose SOS}
\]
\[
F = \{f_1, f_2, f_3, ..., f_k\}; \text{set of SOSI developed from } E
\]
\[
M = \{m_1, m_2, m_3, ..., m_h\}; \text{set of messages used by SOS}
\]

3.1.2 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 3.2 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.
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 3.2.

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 3.2: \[ f_k = \{E_k, N_k, P_k, M_k, C_k\} \]

where,

\[
\begin{align*}
E_k &= \{e_{k1}, e_{k2}, e_{k3}, \ldots, e_{ks}\}; \text{set of Elements in } f_k \\
N_k &= \{n_{k1}, n_{k2}, n_{k3}, \ldots, n_{kw}\}; \text{set of Nodes in } f_k \\
P_k &= \{p_{k1}, p_{k2}, p_{k3}, \ldots, p_{kr}\}; \text{set of SOSIC in } f_k \\
M_k &= \{m_{k1}, m_{k2}, m_{k3}, \ldots, m_{kq}\}; \text{set of Messages in } f_k \\
C_k &= \{c_{k1}, c_{k2}, c_{k3}, \ldots, c_{kb}\}; \text{set of Links in } f_k
\end{align*}
\]

Equations 3.1 and 3.2 describe the relationship of the Node to the SOSI. Equation 3.1 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 3.2 shows that the
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 3.3 shows that Elements may be members of more than SOSIC. Equation 3.4 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 3.3 includes a diagram showing the relationships described in Equations 3.1, 3.2, 3.3, and 3.4. 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.

\[
\bigcap_{j=1}^{m} n_{kj} = \emptyset \tag{3.1}
\]

\[
\bigcup_{j=1}^{m} n_{kj} = E_k \tag{3.2}
\]

where,

\( m = \) the number of Nodes in SOSI \( f_k \)

\[
\bigcap_{j=1}^{m} p_{kj} \neq \emptyset \tag{3.3}
\]

\[
\bigcup_{j=1}^{n} p_{kj} \leq E_k \tag{3.4}
\]

where

\( n = \) the number of SOSICs in SOSI \( f_k \)
3.2 SYSTEM OF SYSTEMS INSTANCE 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.

3.2.1 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 that 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 3.5 shows the
calculation of Node Cohesion. SOSI Cohesion, Equation 3.6, is the average Node Cohesion. The following section is a discussion on computing the number of paths.

\[
\text{Coh}(n_{ki}) = \frac{z_{ki}}{x_{ki}} \tag{3.5}
\]

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

\[
\text{Coh}(f_k) = \frac{\sum_{i=1}^{m} \text{Coh}(n_{ki})}{m} \tag{3.6}
\]

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 3.4 below illustrates the modification of the CPN that represents the Node and the calculation of the paths. Figure 3.4a 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 3.4b 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. Figure 3.4c&d show the paths for this net.

Figure 3.5 shows two CPNs that represent SOSI Nodes. Node\(_1\) is an example of low cohesion. Node\(_2\) is an example of higher cohesion. Both Nodes have three inputs and three outputs. In Node\(_1\) the inputs are associated with only one output, so the cohesion is lower than that of Node\(_2\) where all the inputs affect all the outputs.

Figure 3.6 shows a more complicated node with loops in the CPN representation. In the case of Figure 3.6, Node\(_3\) 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.
3.2.2 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.

![Diagram of export coupling](image)

Figure 3.5. Cohesion Example

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 3.3 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 3.3**

$\text{Scenario}_x = \{ I_x, R_x, f_k \}$

$I_x$ = the initial condition

$R_x$ = parameters

$f_k$ = the particular SOSI

Equation 3.7 shows the formula for Node Coupling. This indicates the degree of dependence of a Node to all the other Nodes.
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 to change due to the complexity of the interfaces 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 3.8.

\[
\text{Coup}_x (f_k) = \frac{\sum_{i=1}^{w} \text{Coup}_x (n_{ki})}{m} 
\]

(3.8)
3.2.3 Adaptability Computation

*Adaptability* is a function of the product of Cohesion and Coupling. Equation 3.9 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 3.9 is the formula for calculating Node Adaptability. Equation 3.9 uses Node Coupling and Node Cohesion and the inputs for the calculation. Equation 3.10 is the SOSI level measure of Adaptability. It uses the SOSI level measure of Cohesion and Coupling from Equations 3.6 and 3.8, respectively.

\[
\text{Adapt}_{x}(n_{ki}) = \frac{1}{D} \quad (3.9)
\]

where,

\[
D = \text{Coh}(n_{ki})^{\alpha} \cdot \text{Coup}(n_{ki})^{\beta}
\]

- $\alpha =$ elasticity constant for Cohesion
- $\beta =$ elasticity constant for Coupling
- $\alpha + \beta = 1$

\[
\text{Adapt}_{x} = \frac{1}{\text{Coh}(f_{k})^{\alpha} \cdot \text{Coup}_{x}(f_{k})^{\beta}} \quad (3.10)
\]

Two assumptions must be made in relation to Equation 3.9 and 3.10. 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 a 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 3.7 shows Cobb-Douglas in its traditional form with different values for $\alpha$ and $\beta$. 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 3.8 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. (There are no empirical data for computing the elasticities. Here they are considered as design parameters). 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, $\alpha$ and $\beta$ define the relationship between changes in Cohesion and Coupling.

Figure 3.9 is a contour plot of Figure 3.8. 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 3.9 shows an example contour
diagram of the Adaptability plot in Figure 3.8. Notice how the score decreases with the increase in cohesion as opposed to an increase in coupling.

Figure 3.10 illustrates how the contour plot is calculated. The contour diagram (bottom) has a contour line indicating the Adaptability value (z-axis of Figure 3.8) 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 3.11 shows three Adaptability plots with different $\alpha$ and $\beta$ values to show how the shape changes with changes in the elasticity constants. The middle plot with $\alpha$ and $\beta$ equal to .8 and .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 $\alpha$ and $\beta$ are reversed. The top right plot shows $\alpha$ and $\beta$ equal to .5.

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
P = BC^\alpha L^\beta

where;
P = Production
C = Capital
L = Labor
B = Scale factor

Figure 3.7. Cobb-Douglas Production Function
Figure 3.8. Adaptability Plot

Figure 3.9. Adaptability Contour
Figure 3.10. Contour Calculation
Figure 3.11. Effects of Elasticity Constant on Adaptability
Figure 3.12 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 3.12. Relationship of Cohesion and Coupling to Adaptability](image)

Figure 3.12. Relationship of Cohesion and Coupling to Adaptability
3.2.4 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 3.11 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 3.12 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.

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

where,
A = a set
\( x \) = a possible element of A
\[
\text{DoR}(p_{ki}) = \frac{\sum_{e \in P_{ki}} \sum_{i=1}^{r} \text{member}_{p_{ki}}(e)}{r}
\]

(3.12)

where,

\(r = \) number of Elements in SOSIC \(p_{ki}\)

Equation 3.13 shows the SOSI computation for Degree of Reuse. It measures the overall Degree of Reuse for the SOSI. Equation 3.14 identifies the Elements that are members of the most SOSICs. This helps identify “highly reused” Elements of the SOSI for the assessment methodology.

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

(3.13)

where,

\(s = \) the number of Elements in SOSI \(f_k\)

\[
E_{\text{max}}(f_k) = \sum_{e \in E_k} \max \left( \sum_{p_{ki} \in P_k} \text{member}_{p_{ki}}(e) \right)
\]

(3.14)

Finally, Exclusiveness is computed as the inverse of degree of reuse. It is calculated for the SOSI as a whole. Equation 3.15 shows the formula to Exclusiveness.

\[
\text{Exclusiveness}(f_k) = \frac{1}{\text{DoR}(f_k)}
\]

(3.15)
3.2.5 Agility

Agility is Adaptability times Exclusiveness. Exclusiveness represents the scale factor defined by Cobb and 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 3.14 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 3.13 shows an Adaptability plot from Figure 3.8 that has been scaled by Exclusiveness = 0.5.

\[
\text{Agility}_k(f_k) = \text{Exclusiveness}(f_k) \cdot \text{Adapt}(f_k)
\]  

(3.14)

Figure 3.13. Agility Plot
3.3 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 3.14 shows the concurrently available SOSICs and the Nodes to which the Elements have been allocated. The set of Nodes, \( N_1 = \{n_{1,1}, n_{1,2}, n_{1,3}\} \) in Figure 3.14 graphically illustrates the allocation of elements. Figure 3.15, Figure 3.16, and Figure 3.17 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.

![SOSI, \( f_1 \)](image)

Figure 3.14. Working Example for Adaptability Calculations with Nodes
3.3.1 Cohesion

The cohesion computation begins with computing the number of paths that connect Node inputs with Node outputs represented in the CPN. Node $n_{1,1}$ from Figure 3.14 will be the initial example. Figure 3.15 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 and 1 output therefore cohesion for Node $n_{1,1}$ is 11/4. Figure 3.16 and Figure 3.17 show the CPN for Node $n_{1,2}$ and Node $n_{1,3}$ from the example. The results of the Cohesion computation appear at the bottom of each diagram. Table 3.1 summarizes the Cohesion results of the example.

3.3.2 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 3.18 shows the results for the coupling calculations for SOSI $f_1$. Node $n_{1,1}$ exported 20 messages to Node $n_{1,3}$ and zero to Node $n_{1,2}$. 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.

With the calculation for Coupling and Cohesion completed, adaptability can be calculated. The Adaptability results for the example are shown in Figure 3.19. 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.

![Diagram of CPN for n_{1,1}](image)

Simple Information Flow Paths

- P1, P12
- P2, P5, P7, P9, P12
- P2, P5, P7, P10, P11, P12
- P2, P5, P6, P8, P9, P12
- P2, P5, P6, P8, P10, P11, P12

P3, P7, P9, P12
P3, P7, P10, P11, P12
P3, P6, P8, P9, P12
P3, P6, P8, P10, P11, P12
P4, P8, P9, P12
P4, P8, P10, P11, P12

I = 4, Q = 1
x = 4(1) = 4
z = 11
Coh(n_{1,1}) = 11/4

Figure 3.15. CPN for n_{1,1}
Figure 3.16. Colored Petri Net (CPN) for $n_{1,2}$

Figure 3.17. Colored Petri Net (CPN) for $n_{1,3}$
Table 3.1. Cohesion Example

<table>
<thead>
<tr>
<th>Node</th>
<th>I (Inputs)</th>
<th>Q (Outputs)</th>
<th>z (poss. paths)</th>
<th>x (total paths)</th>
<th>Coh</th>
</tr>
</thead>
<tbody>
<tr>
<td>n_{1,1}</td>
<td>4</td>
<td>1</td>
<td>4</td>
<td>11</td>
<td>11/4</td>
</tr>
<tr>
<td>n_{1,1}</td>
<td>3</td>
<td>3</td>
<td>9</td>
<td>3</td>
<td>3/9</td>
</tr>
<tr>
<td>n_{1,1}</td>
<td>2</td>
<td>4</td>
<td>8</td>
<td>11</td>
<td>11/8</td>
</tr>
<tr>
<td>SOSI $f_1$</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>3/2</td>
</tr>
</tbody>
</table>

Figure 3.18. Coupling Results for SOSI $f_1$

$$
C = \frac{\sum |M_x (n_{ki}, n_{kj})|}{C}
$$

$$
\text{Coup}_{x}(n_{1,1}) = \frac{20}{3} = 6.67
$$

$$
\text{Coup}_{x}(n_{1,2}) = \frac{20}{3} = 6.67
$$

$$
\text{Coup}_{x}(n_{1,3}) = \frac{20}{3} = 6.67
$$

$$
\text{Coup}_{x}(f_1) = 6.67
$$
3.3.3 Degree of Reuse

The set of interest for these calculations is the set of SOSIC $P_1$ associated with SOSI $f_1$. There are three SOSIC represented. Figure 3.20 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 $e_{1,8}$ supports all three SOSICs.
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 \( p_{1,1} \), \( p_{1,2} \), and \( p_{1,3} \), respectively. This results in degree of reuse values for the SOSIC \( p_{1,1} \), \( p_{1,2} \), and \( p_{1,3} \) of \( 9/5 \), \( 8/4 \) and \( 11/7 \), respectively. The SOSI Degree of reuse is \( 16/11 \). The Overlap calculations are summarized in Table 3.2 and Table 3.3. Each \( x \) in the Table 3.2 indicates the SOSICs that include a particular Element. The total (tot) shows the total number of SOSICs for that Element. The Table 3.3 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 3.2. Degree of Reuse Example Data

<table>
<thead>
<tr>
<th></th>
<th>e\textsubscript{1,1}</th>
<th>e\textsubscript{1,2}</th>
<th>e\textsubscript{1,4}</th>
<th>e\textsubscript{1,5}</th>
<th>e\textsubscript{1,8}</th>
<th>e\textsubscript{1,10}</th>
<th>e\textsubscript{1,14}</th>
<th>e\textsubscript{1,15}</th>
<th>e\textsubscript{1,17}</th>
<th>e\textsubscript{1,19}</th>
<th>e\textsubscript{1,20}</th>
</tr>
</thead>
<tbody>
<tr>
<td>p\textsubscript{1,1}</td>
<td></td>
<td></td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>p\textsubscript{1,2}</td>
<td>x</td>
<td>x</td>
<td></td>
<td>x</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>p\textsubscript{1,3}</td>
<td>x</td>
<td></td>
<td>x</td>
<td></td>
<td>x</td>
<td>x</td>
<td></td>
<td></td>
<td></td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>tot</td>
<td>1</td>
<td>2</td>
<td>2</td>
<td>1</td>
<td>3</td>
<td>1</td>
<td>2</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

Table 3.3. Degree of Reuse Calculation for SOSI \textsubscript{f\textsubscript{1}}

<table>
<thead>
<tr>
<th></th>
<th>Num Elements</th>
<th>Total Reuse</th>
<th>Degree of Reuse</th>
<th>Summary</th>
</tr>
</thead>
<tbody>
<tr>
<td>p\textsubscript{1,1}</td>
<td>5</td>
<td>9</td>
<td>9/5</td>
<td>1.8</td>
</tr>
<tr>
<td>p\textsubscript{1,2}</td>
<td>4</td>
<td>8</td>
<td>8/4</td>
<td>2.0</td>
</tr>
<tr>
<td>p\textsubscript{1,3}</td>
<td>7</td>
<td>11</td>
<td>11/7</td>
<td>1.57</td>
</tr>
<tr>
<td>f\textsubscript{1}</td>
<td>11</td>
<td>16</td>
<td>16/11</td>
<td>1.45</td>
</tr>
</tbody>
</table>

The working example shows that SOSIC p\textsubscript{1,3} has the highest degree of reuse and SOSIC p\textsubscript{1,2} has the lowest; therefore, we can conclude, in relative terms, that p\textsubscript{1,3} has less potential for resource conflict than p\textsubscript{1,2}. The Degree of Reuse also indicates Elements that may be used beyond their capacity. In this simple example, Element e\textsubscript{1,8} 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.
3.3.4 Agility

Agility is Adaptability scaled by Exclusiveness. Figure 3.21 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 3.21. Example Results for Adaptability and Agility

3.4 SUMMARY

This chapter 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.
The 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. Chapter Four presents the methodology used to gather the information required to generate the executable model and conduct the analysis.
CHAPTER 4

METHODOLOGY

4.1 INTRODUCTION

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 it into an executable form for a static and dynamic analysis of the interaction between Elements and Nodes of the SOSI.

Figure 4.1 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, where 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 4.2 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.

4.2 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.

Figure 4.2. Piecewise Constant SOSI

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.

4.3 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 to develop the structural, operational, and behavior information required to construct the combined SOSI architecture.

### 4.4 SYSTEM OF SYSTEMS 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, 2007a] 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 4.3, 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.

**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.

What follows 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 4.4, 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 4.1. 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.

![Operational Node Connectivity Diagrams, OV-2s](image)

Figure 4.4. Operational Node Connectivity Diagrams, OV-2s
Table 4.1. Operational Information Exchange Matrix, OV-3

<table>
<thead>
<tr>
<th>Capability 1</th>
<th>Capability 2</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Sender</strong></td>
<td><strong>Sender</strong></td>
</tr>
<tr>
<td><strong>Msg1</strong></td>
<td><strong>Msg1</strong></td>
</tr>
<tr>
<td><strong>Msg2</strong></td>
<td><strong>Msg3</strong></td>
</tr>
</tbody>
</table>

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 4.2 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.

Table 4.2. Operational Rules Model, OV-6a

<table>
<thead>
<tr>
<th>Capability 1</th>
<th>Capability 2</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Sender</strong></td>
<td><strong>Sender</strong></td>
</tr>
<tr>
<td>if msg2 send msg1</td>
<td>if msg1 = yes send msg2</td>
</tr>
<tr>
<td>if msg1 = no send msg3</td>
<td>if msg3 then send msg1</td>
</tr>
</tbody>
</table>

The Operational Data Models, OV-7s in Figure 4.5 show the messages and roles that appear in the Operational Architecture Views for each capability.

The final diagram, Figure 4.6, 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 4.5. Operational Data Model, OV-7
Figure 4.6. 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 enable 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 that will realize the roles defined in the OV-2. Figure 4.7 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 System3 represent the Receiver and Executor, respectively.

Similar to the OV-3, the Systems Data Exchange Matrices, SV-6s, shown in Table 4.3, 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.
Figure 4.7. System Interface Descriptions, SV-1s

Table 4.3. Systems Data Exchange Matrix, SV-6

<table>
<thead>
<tr>
<th></th>
<th>Sender</th>
<th>Receiver</th>
<th>Type</th>
<th>Length</th>
</tr>
</thead>
<tbody>
<tr>
<td>SA1</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Msg1</td>
<td>System1</td>
<td>System2</td>
<td>Text</td>
<td>10</td>
</tr>
<tr>
<td>Msg2</td>
<td>System2</td>
<td>System1</td>
<td>Text</td>
<td>20</td>
</tr>
<tr>
<td>SA2</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Msg1</td>
<td>System3</td>
<td>System2</td>
<td>Text</td>
<td>10</td>
</tr>
<tr>
<td>Msg3</td>
<td>System2</td>
<td>System3</td>
<td>Text</td>
<td>32</td>
</tr>
</tbody>
</table>

The Physical Schemas, SV-11s, in Figure 4.8 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.
The SV-5s, shown in Table 4.4, 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.

The final representations describe the behavior of the system architecture. The Systems Rules Model, SV-10a, Table 4.5, 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.

Figure 4.8. Physical Schema, SV-11
Table 4.4. Operational Activity to Systems Function Traceability Matrices, SV-5s

<table>
<thead>
<tr>
<th>Role</th>
<th>Op Activity</th>
<th>System 1</th>
<th>System 2</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Sys1Action</td>
<td>Sys2Action</td>
</tr>
<tr>
<td>Sender</td>
<td>ReceiveMsg</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td></td>
<td>ProcessMsg</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td></td>
<td>SendMsg</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>Receiver</td>
<td>ReceiveMsg</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td></td>
<td>ProcessMsg</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td></td>
<td>SendMsg2</td>
<td>X</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Role</th>
<th>Op Activity</th>
<th>System 2</th>
<th>System 3</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Sys2Action2</td>
<td>Sys3Action</td>
</tr>
<tr>
<td>Receiver</td>
<td>ReceiveMsg3</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td></td>
<td>ProcessMsg</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td></td>
<td>SendMsg3</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>Executor</td>
<td>ReceiveMsg3</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td></td>
<td>ProcessMsg</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td></td>
<td>SendMsg</td>
<td>X</td>
<td></td>
</tr>
</tbody>
</table>

Table 4.5. 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 4.9, 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 an 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 a 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 4.10 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 4.9. Systems Functionality Description, SV-4
The SV-6, shown in Table 4.6, 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.

Figure 4.10. SOSI Architecture SV-1

Table 4.6. SOSI Architecture SV-6

<table>
<thead>
<tr>
<th>SA1</th>
<th>Sender</th>
<th>Receiver</th>
<th>Type</th>
<th>Length</th>
</tr>
</thead>
<tbody>
<tr>
<td>Msg1</td>
<td>System1</td>
<td>System2</td>
<td>Text</td>
<td>10</td>
</tr>
<tr>
<td>Msg1</td>
<td>System3</td>
<td>System2</td>
<td>Text</td>
<td>10</td>
</tr>
<tr>
<td>Msg2</td>
<td>System2</td>
<td>System1</td>
<td>Text</td>
<td>20</td>
</tr>
<tr>
<td>Msg3</td>
<td>System2</td>
<td>System3</td>
<td>Text</td>
<td>32</td>
</tr>
</tbody>
</table>
The SV-lls are merged to ensure that all the data in both system architecture views is represented in the SOSI architecture. Figure 4.11 is the merged SV-11. The merged SV-11 shows that System2 has an association with both System1 and System3.

Table 4.7 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.

![SOSI Architecture SV-11](image)

**Figure 4.11.** SOSI Architecture SV-11
With the rule and data represented, the SOSICs are modified versions of the SV-4s taken from the identified system architecture views. Figure 4.12 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 the SOSIC and capabilities modeled. This is not the case in general. Multiple SOSICs can be developed that accomplish particular tasks using the same capability.

**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 Chapter 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 Chapter 5.

**Step 6. Conduct the Analysis for Each Alternative**

Step 6 conducts the analysis for each alternative using the measures described in Chapter 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.
Figure 4.12. SOSI Architecture SV-4
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.

4.5 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 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, Steps four and five, from the UML Activity Diagram to the CPN, are detailed in Chapter 5.
CHAPTER 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 Chapter 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 section provides an overview of the methodology for transforming the Activity Diagram into a CPN. The second section describes the transformation of an Activity Diagram. This transformation into the CPN represents Step 4 of the primary methodology. The last section is Step 5 of the primary methodology and describes the unique modeling artifacts that are required to complete the executable model.
5.1 TRANSFORMATION PROCESS

The steps in the transformation process are similar to the MDA process depicted in Figure 5.1. 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 CPN Tools.

![Figure 5.1: SOS Model Driven Development Process](image)

The transformation process has four steps that follow closely the MDD environment described in Figure 5.1. 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 5.2 shows the process with some the steps described in Chapter 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 CPN Tools 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.

The examples used in this chapter extend the example begun in chapter 4. Figure 5.3 shows the Activity Diagrams for the Element behavior. The Element Activity Diagrams are connected together to create the SOSIC Activity Diagram. Figure 5.4 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.

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 (\textbullet{}), Object Node (\textbullet{}), Call Behavior Action (\textbullet{}), Fork/Join Node (\textbullet{}), Decision Node (\textbullet{}), Initial Node (\textbullet{}), Final Node (\textbullet{}), Stop Node (\textbullet{}), and Activity Parameter Node (\textbullet{}). The components of the CPN are: Transition (\textbullet{}), Place (\textbullet{}), and Substitution Transition (\textbullet{}).

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.
Figure 5.4. Element SOSIC Participation
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 arc 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 5.5 summarizes the above discussion. The Activity diagram components are shown in the top row and first column of Figure 5.5. 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.

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 5.6 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.
5.2 UML TO CPN TRANSFORMATION

Figure 5.7 illustrates the concrete syntax transformation using a fragment of an Activity Diagram and the transformation rules shown in Figure 5.5. 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 5.5. 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 chapter 4.

The first transformation creates the PSM from the SOSIC represented as an Activity Diagram. Figure 5.8 shows an excerpt of the SOSIC Example from Figure 5.3 and Figure 5.4 transformed into an instance of the CPN data model representation, Figure 5.6. 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 5.5. 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.

Step 4.2 is the transformation of the CPN data model to the CPN Tools xml representation. Figure 5.9 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 5.10 are the subpages from the example. Using the connection between System3 and System2 as an example, the place P31 in Figure 5.9 is represented in Figure 5.10 in the CPN pages that represent System2 and System3. Place P31 is an input place for System2 and an output place for System3. The top level CPN page, Figure 5.9, shows this relationship between System2 and System3.

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. CPN Tools 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.

Figure 5.8. Example CPN Data Model
Figure 5.9. Top-level CPN Representation

Figure 5.10. Sub-pages for Example CPN
5.3 SUMMARY

This chapter 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.
CHAPTER 6
CASE STUDY

6.1 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.

6.2 SCENARIO

The fictional mission in this case is as follows [Levis, 2006]. 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 6.1 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 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.

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.
6.3 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 6.2 through Figure 6.4 show the OV-2s for each capability. Figure 6.2 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 6.3 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 6.4 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.

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 6.1 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.
Figure 6.2. Planning and Coordination OV-2

Figure 6.3. Blue Force Tracking OV-2
Figure 6.4. Process and Disseminate Intelligence Information OV-2

Table 6.1. ESG Operational Information Exchange Matrix OV-3

<table>
<thead>
<tr>
<th>Planning and Coordination Capability</th>
<th>Sender</th>
<th>Receiver</th>
<th>Type</th>
<th>Length</th>
</tr>
</thead>
<tbody>
<tr>
<td>Request</td>
<td>Requester</td>
<td>Coordinator</td>
<td>Text</td>
<td>32</td>
</tr>
<tr>
<td>Order</td>
<td>Coordinator</td>
<td>Planner</td>
<td>Text</td>
<td>Variable</td>
</tr>
<tr>
<td>Status</td>
<td>Planner</td>
<td>Coordinator</td>
<td>Text</td>
<td>Variable</td>
</tr>
<tr>
<td>Coordination</td>
<td>Executor</td>
<td>Requestor</td>
<td>Text</td>
<td>32</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Blue Force Tracking Capability</th>
<th>Sender</th>
<th>Receiver</th>
<th>Type</th>
<th>Length</th>
</tr>
</thead>
<tbody>
<tr>
<td>BluePLI</td>
<td>Reporter</td>
<td>Distributor</td>
<td>Text</td>
<td>32</td>
</tr>
<tr>
<td>BluePLI</td>
<td>Distributor</td>
<td>Receiver</td>
<td>Text</td>
<td>32</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Process and Disseminate Intelligence Information</th>
<th>Sender</th>
<th>Receiver</th>
<th>Type</th>
<th>Length</th>
</tr>
</thead>
<tbody>
<tr>
<td>Input</td>
<td>Sensor</td>
<td>Controller</td>
<td>Text</td>
<td>48</td>
</tr>
<tr>
<td>SpotReport</td>
<td>Controller</td>
<td>Distributor</td>
<td>Text</td>
<td>48</td>
</tr>
<tr>
<td>SpotReport</td>
<td>Distributor</td>
<td>Analyzer</td>
<td>Text</td>
<td>48</td>
</tr>
<tr>
<td>RedPLI</td>
<td>Analyzer</td>
<td>Distributor</td>
<td>Text</td>
<td>48</td>
</tr>
<tr>
<td>RedPLI</td>
<td>Distributor</td>
<td>Receiver</td>
<td>Text</td>
<td>48</td>
</tr>
</tbody>
</table>
Operational activities describe the actions that each role executes to create the capability. The Operational Rules Models, OV-6a, Table 6.2, documents 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.

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 Figure 6.5, Figure 6.6, and Figure 6.7. 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.

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 6.8 is the Activity Diagram for Planning and Coordination followed by Figure 6.9 and Figure 6.10 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.
Table 6.2. ESG Operational Rules Model OV-6a

<table>
<thead>
<tr>
<th>Planning and Coordination Capability</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Requester:</strong> SendRequest</td>
</tr>
<tr>
<td><strong>Requester:</strong> ReceiveCoordination</td>
</tr>
<tr>
<td><strong>Coordinator:</strong> ReceiveRequest</td>
</tr>
<tr>
<td><strong>Coordinator:</strong> ProcessRequest</td>
</tr>
<tr>
<td><strong>Coordinator:</strong> ProcessRequest</td>
</tr>
<tr>
<td><strong>Coordinator:</strong> SendOrder</td>
</tr>
<tr>
<td><strong>Coordinator:</strong> DistributeOrder</td>
</tr>
<tr>
<td><strong>Coordinator:</strong> ReceiveStatus</td>
</tr>
<tr>
<td><strong>Planner:</strong> ReceiveOrder</td>
</tr>
<tr>
<td><strong>Planner:</strong> ProcessOrder</td>
</tr>
<tr>
<td><strong>Planner:</strong> ProcessOrder</td>
</tr>
<tr>
<td><strong>Planner:</strong> SendOrder</td>
</tr>
<tr>
<td><strong>Executor:</strong> ReceiveOrder</td>
</tr>
<tr>
<td><strong>Executor:</strong> ComputeStatus</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Blue Force Tracking Capability</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Reporter:</strong> ComputeBpli</td>
</tr>
<tr>
<td><strong>Reporter:</strong> SendBpli</td>
</tr>
<tr>
<td><strong>Distributor:</strong> ReceiveBpli</td>
</tr>
<tr>
<td><strong>Distributor:</strong> DistributeBpli</td>
</tr>
<tr>
<td><strong>Receiver:</strong> ReceiveBpli</td>
</tr>
<tr>
<td><strong>Receiver:</strong> ReceiveBpli</td>
</tr>
</tbody>
</table>

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:** ReceiveRpli               | Receive and Store all RedPLI        |
Figure 6.5. Planning and Coordination Capability OV-7

Figure 6.6. Blue Force Tracking Capability OV-7

Figure 6.7. Process and Disseminate Intelligence Information Capability OV-7
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 (P2P), centralized-server (CS) and service oriented.
architecture (SOA). Figure 6.11 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.

![Diagram](image)

**Figure 6.9. Blue Force Tracking OV-5**

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.
Figure 6.10. Process and Disseminate Intelligence Information OV-5

Figure 6.11. Relationship Among Architecture Views
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 Figure 6.12, Figure 6.13, and Figure 6.14. 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.

![Diagram](image)

**Figure 6.12. Planning and Coordination SV-1**
Figure 6.13. Blue Force Tracking SV-1

Figure 6.14. Process and Disseminate Intelligence Information 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 are shown in Table 6.3, Table 6.4, and Table 6.5. 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.

Table 6.3. Planning and Coordination SV-6

<table>
<thead>
<tr>
<th></th>
<th>Sender</th>
<th>Receiver</th>
<th>Type</th>
<th>Len</th>
</tr>
</thead>
<tbody>
<tr>
<td>Request</td>
<td>TLC2S</td>
<td>OLC2S</td>
<td>Text</td>
<td>32</td>
</tr>
<tr>
<td>Request</td>
<td>OLC2S</td>
<td>OLC2S</td>
<td>Text</td>
<td>32</td>
</tr>
<tr>
<td>Order</td>
<td>TLC2S</td>
<td>TLC2S</td>
<td>Text</td>
<td>*</td>
</tr>
<tr>
<td>Order</td>
<td>OLC2S</td>
<td>OLC2S</td>
<td>Text</td>
<td>*</td>
</tr>
<tr>
<td>Order</td>
<td>OLC2S</td>
<td>TLC2S</td>
<td>Text</td>
<td>*</td>
</tr>
<tr>
<td>Coordination</td>
<td>TLC2S</td>
<td>TLC2S</td>
<td>Text</td>
<td>32</td>
</tr>
<tr>
<td>Status</td>
<td>TLC2S</td>
<td>TLC2S</td>
<td>Text</td>
<td>56</td>
</tr>
<tr>
<td>Status</td>
<td>OLC2S</td>
<td>OLC2S</td>
<td>Text</td>
<td>56</td>
</tr>
<tr>
<td>Status</td>
<td>OLC2S</td>
<td>TLC2S</td>
<td>Text</td>
<td>56</td>
</tr>
</tbody>
</table>

Table 6.4. Blue Force Tracking SV-6

<table>
<thead>
<tr>
<th></th>
<th>Sender</th>
<th>Receiver</th>
<th>Type</th>
<th>Len</th>
</tr>
</thead>
<tbody>
<tr>
<td>BluePLI</td>
<td>BFT</td>
<td>OLC2S</td>
<td>Text</td>
<td>32</td>
</tr>
<tr>
<td>BluePLI</td>
<td>BFT</td>
<td>TLC2S</td>
<td>Text</td>
<td>32</td>
</tr>
<tr>
<td>BluePLI</td>
<td>BFT</td>
<td>ICS</td>
<td>Text</td>
<td>32</td>
</tr>
<tr>
<td>BluePLI</td>
<td>BFT</td>
<td>BFT</td>
<td>Text</td>
<td>32</td>
</tr>
</tbody>
</table>
Table 6.5. Process and Disseminate Intelligence Information SV-6

<table>
<thead>
<tr>
<th>Sender</th>
<th>Receiver</th>
<th>Type</th>
<th>Len</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sighting</td>
<td>HumintSensor</td>
<td>ICS</td>
<td>Text</td>
</tr>
<tr>
<td>Sighting</td>
<td>SigintSensor</td>
<td>ICS</td>
<td>Text</td>
</tr>
<tr>
<td>Signal</td>
<td>SurfaceRadar</td>
<td>ICS</td>
<td>Text</td>
</tr>
<tr>
<td>RedPLI</td>
<td>ICS</td>
<td>OLC2S</td>
<td>Text</td>
</tr>
<tr>
<td>RedPLI</td>
<td>ICS</td>
<td>TLC2S</td>
<td>Text</td>
</tr>
</tbody>
</table>

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. Table 6.6, Table 6.7, and Table 6.8 are the rule models for the system architectures. Table 6.6 shows that if a Request message is received send a Request message.

Table 6.6. Planning and Coordination SV-10a

<table>
<thead>
<tr>
<th>TLC2S</th>
</tr>
</thead>
<tbody>
<tr>
<td>if Request then Request</td>
</tr>
<tr>
<td>if Order &lt;&gt; ord_warning then SendOrder</td>
</tr>
<tr>
<td>if Status then SendStatus</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>OLC2S</th>
</tr>
</thead>
<tbody>
<tr>
<td>if Request then Request</td>
</tr>
<tr>
<td>if Order &lt;&gt; ord_warning then SendOrder</td>
</tr>
<tr>
<td>if Status then SendStatus</td>
</tr>
</tbody>
</table>
Table 6.7. Blue force Tracking SV-10a

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>TLC2S</td>
<td>if BluPLI then PassBpli</td>
</tr>
<tr>
<td>OLC2S</td>
<td>if BluPLI then PassBpli</td>
</tr>
<tr>
<td>BFT</td>
<td>if BluePLI &lt;&gt; rpli_pass the rpli</td>
</tr>
</tbody>
</table>

Table 6.8. Process and Disseminate Intelligence Information SV-10a

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>TLC2S</td>
<td>if RedPLI then SendRpli</td>
</tr>
<tr>
<td>OLC2S</td>
<td>if RedPLI then SendRpli</td>
</tr>
<tr>
<td>ICS</td>
<td></td>
</tr>
<tr>
<td></td>
<td>if SpotReport = sr_pers then rpli_pers</td>
</tr>
<tr>
<td></td>
<td>if SpotReport = sr_equip then rpli_equip</td>
</tr>
<tr>
<td></td>
<td>if RedPLI = rpli_pass then empty</td>
</tr>
<tr>
<td></td>
<td>if Blip = blip_pers then sr_pers</td>
</tr>
<tr>
<td></td>
<td>if Blip = blip_equip then sr_equip</td>
</tr>
<tr>
<td></td>
<td>if Signal = signal_pers then sr_pers</td>
</tr>
<tr>
<td></td>
<td>if Signal = signal_equip then sr_equip</td>
</tr>
<tr>
<td></td>
<td>if Sighting = sighting_pers then sr_pers</td>
</tr>
<tr>
<td></td>
<td>if Sighting = sighting_equip then sr_equip</td>
</tr>
</tbody>
</table>

The SV-11 shows the physical schema for each system architecture. Figure 6.15, Figure 6.16, and Figure 6.17 show the physical schemas for the system architectures.

Each one describes the Messages and Systems that realize the capability.
The SV-4 is represented in this methodology as a 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 means there is a lower level activity diagram the further represents the behavior of the system. Figure 6.18, Figure 6.19, and Figure 6.20 represent the SV-4s for the system architectures.

![Diagram](image)

Figure 6.17. Process and Disseminate Intelligence information SV-11

Figure 6.21 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.
Figure 6.18. Planning and Coordination SV-4
Figure 6.19. Blue Force Tracking SV-4
Figure 6.20. Process and Disseminate Intelligence Information SV-4
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 6.9 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 Table 6.9, Table 6.10, and Table 6.11.
Table 6.9. Planning and Coordination SV-5

<table>
<thead>
<tr>
<th>Role</th>
<th>Activity/Function</th>
<th>TLC2S</th>
<th>OLC2S</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Send</td>
<td>Rec</td>
</tr>
<tr>
<td>Requestor</td>
<td>SendRequest</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td></td>
<td>Receive Coordination</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Coordinator</td>
<td>Receive Request</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td></td>
<td>ProcessRequest</td>
<td></td>
<td>X</td>
</tr>
<tr>
<td></td>
<td>SendOrder</td>
<td></td>
<td>X</td>
</tr>
<tr>
<td></td>
<td>Distribute Order</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>ReceiveStatus</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Planner</td>
<td>RecieveOrder</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td></td>
<td>ProcessOrder</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td></td>
<td>SendOrder</td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>Executor</td>
<td>RecieveOrder</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td></td>
<td>ProcessOrder</td>
<td></td>
<td>X</td>
</tr>
<tr>
<td></td>
<td>Send Coordination</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Compute Status</td>
<td></td>
<td>X</td>
</tr>
</tbody>
</table>

Table 6.10. Blue Force Tracking SV-5

<table>
<thead>
<tr>
<th>Role</th>
<th>Activity/Function</th>
<th>TLC2S</th>
<th>OLC2S</th>
<th>BFT</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>ProcSA</td>
<td>ProcSA</td>
<td>SendBpli</td>
</tr>
<tr>
<td>Reporter</td>
<td>ComputeBpli</td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td></td>
<td>SendBpli</td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>Distributor</td>
<td>ReceiveBpli</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td></td>
<td>DistributeBpli</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Receiver</td>
<td>RecieveBpli</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
</tbody>
</table>
### Table 6.11. Process and Disseminate Intelligence Information SV-5

<table>
<thead>
<tr>
<th>Role</th>
<th>Activity/Function</th>
<th>System</th>
<th>OLC2S</th>
<th>ICS</th>
<th>Sensor</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sensor</td>
<td>Sense</td>
<td>Proc</td>
<td>Proc</td>
<td>Proc</td>
<td>Proc</td>
</tr>
<tr>
<td></td>
<td></td>
<td>SA</td>
<td>SA</td>
<td>SR</td>
<td>Sight</td>
</tr>
<tr>
<td>Controller</td>
<td>ProcessInput</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td></td>
<td>SendSpotReport</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Analyzer</td>
<td>ProcessSpotReport</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>SendRedPLI</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Distributor</td>
<td>DistributeRedPLI</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>Receiver</td>
<td>ReceiveRpli</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>StoreRpli</td>
<td></td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
</tr>
</tbody>
</table>

#### 6.5 SYSTEM OF SYSTEMS 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 6.22 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.

![Peer-to-Peer Architecture Diagram]

Figure 6.22. 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; there are up to three servers in the SOSI.

Figure 6.23 shows the architectures for the one server SOSI.

![Client Server SOSI with One Server](image)

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 the SOA architecture configuration. They are
differentiated by the number of instances of each service. SOA1 SOSI, Figure 6.24,
shows a single instance of each service.
Figure 6.24. Service Oriented Architecture with One Instance of Each Service

The P2P SOSI Architecture will be used as the example for the remaining steps of the methodology. Figure 6.25 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).
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. Figure 6.26, Figure 6.27, and Figure 6.28 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 6.26, 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 Efracia. 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 6.27, 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 6.28, 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 6.26. P2P_1 SV-1 Six Nodes
The SOSI Architecture SV-6, Table 6.12, 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.
The SV-10a, Table 6.13, 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.
Table 6.12. P2P Systems Data Exchange Matrix, SV-6

<table>
<thead>
<tr>
<th>Sender</th>
<th>Receiver</th>
<th>Type</th>
<th>Len</th>
</tr>
</thead>
<tbody>
<tr>
<td>Request</td>
<td>TLC2S</td>
<td>OLC2S</td>
<td>Text 32</td>
</tr>
<tr>
<td>Request</td>
<td>OLC2S</td>
<td>OLC2S</td>
<td>Text 32</td>
</tr>
<tr>
<td>Order</td>
<td>TLC2S</td>
<td>TLC2S</td>
<td>Text *</td>
</tr>
<tr>
<td>Order</td>
<td>OLC2S</td>
<td>OLC2S</td>
<td>Text *</td>
</tr>
<tr>
<td>Order</td>
<td>OLC2S</td>
<td>TLC2S</td>
<td>Text *</td>
</tr>
<tr>
<td>Coordination</td>
<td>TLC2S</td>
<td>TLC2S</td>
<td>Text 32</td>
</tr>
<tr>
<td>Status</td>
<td>TLC2S</td>
<td>TLC2S</td>
<td>Text 56</td>
</tr>
<tr>
<td>Status</td>
<td>OLC2S</td>
<td>OLC2S</td>
<td>Text 56</td>
</tr>
<tr>
<td>Status</td>
<td>OLC2S</td>
<td>TLC2S</td>
<td>Text 56</td>
</tr>
<tr>
<td>BluePLI</td>
<td>BFT</td>
<td>OLC2S</td>
<td>Text 32</td>
</tr>
<tr>
<td>BluePLI</td>
<td>BFT</td>
<td>TLC2S</td>
<td>Text 32</td>
</tr>
<tr>
<td>BluePLI</td>
<td>BFT</td>
<td>ICS</td>
<td>Text 32</td>
</tr>
<tr>
<td>BluePLI</td>
<td>BFT</td>
<td>BFT</td>
<td>Text 32</td>
</tr>
<tr>
<td>Sighting</td>
<td>HumintSensor</td>
<td>ICS</td>
<td>Text 48</td>
</tr>
<tr>
<td>Sighting</td>
<td>SigintSensor</td>
<td>ICS</td>
<td>Text 48</td>
</tr>
<tr>
<td>Signal</td>
<td>SurfaceRadar</td>
<td>ICS</td>
<td>Text 48</td>
</tr>
<tr>
<td>RedPLI</td>
<td>ICS</td>
<td>OLC2S</td>
<td>Text 48</td>
</tr>
<tr>
<td>RedPLI</td>
<td>ICS</td>
<td>TLC2S</td>
<td>Text 48</td>
</tr>
</tbody>
</table>

The SV-11, Figure 6.29, is the merged data model from the system architecture views. It completes the SV-11 from Figure 6.25. 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.

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
Table 6.13. P2P Systems Rule Model, SV-10a

<table>
<thead>
<tr>
<th>Rule Model</th>
<th>Conditions</th>
</tr>
</thead>
<tbody>
<tr>
<td>TLC2S</td>
<td>if Request then Request</td>
</tr>
<tr>
<td></td>
<td>if Order &lt;&gt; ord_warning then SendOrder</td>
</tr>
<tr>
<td></td>
<td>if Status then SendStatus</td>
</tr>
<tr>
<td></td>
<td>if SpotReport then PassSpotReport</td>
</tr>
<tr>
<td></td>
<td>if BluPLI then PassBpli</td>
</tr>
<tr>
<td></td>
<td>if RedPLI then SendRpli</td>
</tr>
<tr>
<td>OLC2S</td>
<td>if Request then Request</td>
</tr>
<tr>
<td></td>
<td>if Order &lt;&gt; ord_warning then SendOrder</td>
</tr>
<tr>
<td></td>
<td>if Status then SendStatus</td>
</tr>
<tr>
<td></td>
<td>if SpotReport then PassSpotReport</td>
</tr>
<tr>
<td></td>
<td>if BluPLI then PassBpli</td>
</tr>
<tr>
<td></td>
<td>if RedPLI then SendRpli</td>
</tr>
<tr>
<td>BFT</td>
<td>if BluePLI &lt;&gt; rpli_pass then rpli</td>
</tr>
<tr>
<td>ICS</td>
<td>if SpotReport = sr_pers then rpli_pers</td>
</tr>
<tr>
<td></td>
<td>if SpotReport = sr_equip then rpli_equip</td>
</tr>
<tr>
<td></td>
<td>if RedPLI = rpli_pass then empty</td>
</tr>
<tr>
<td></td>
<td>if Blip = blip_pers then sr_pers</td>
</tr>
<tr>
<td></td>
<td>if Blip = blip_equip then sr_equip</td>
</tr>
<tr>
<td></td>
<td>if Signal = signal_pers then sr_pers</td>
</tr>
<tr>
<td></td>
<td>if Signal = signal_equip then sr_equip</td>
</tr>
<tr>
<td></td>
<td>if Sighting = sighting_pers then sr_pers</td>
</tr>
<tr>
<td></td>
<td>if Sighting = sighting_equip then sr_equip</td>
</tr>
</tbody>
</table>

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 6.30, Figure 6.31, Figure 6.32, and Figure 6.33 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 6.14 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 6.30. P2P SOSI Architecture Process and Disseminate Intelligence Information SOSIC
Figure 6.31. Conduct Support Operations SOSIC
Figure 6.32. Blue Force Tracking SOSIC
Figure 6.33. Conduct Security Operations SOSIC
<table>
<thead>
<tr>
<th></th>
<th>TLC2S</th>
<th></th>
<th>OLC2S</th>
<th></th>
<th>ICS</th>
<th></th>
<th>BFT</th>
<th></th>
<th>Sensor</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Requestor</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>SendRequest</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>ReceiveCoord</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Coordinator</td>
<td>ReceiveRequest</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>ProcessRequest</td>
<td></td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>SendOrder</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>DistributeOrder</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>ReceiveStatus</td>
<td></td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Planner</td>
<td>ReceiveOrder</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>ProcessOrder</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>SendOrder</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Executor</td>
<td>ReceiveOrder</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>ProcessOrder</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>SendCoord</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>ComputeStatus</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>SendStatus</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Reporter</td>
<td>ComputeBpli</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>SendBpli</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Distributor</td>
<td>ReceiveBpli</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>DistributeBpli</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Receiver</td>
<td>ReceiveBpli</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Sensor</td>
<td>Sense</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>SendInput</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Controller</td>
<td>ProcessInput</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>SendSpotReport</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Analyzer</td>
<td>ProcSpotReport</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>SendRedPLI</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Distributor</td>
<td>DistriRedPLI</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Receiver</td>
<td>ReceiveRpli</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>StoreRpli</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
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 6.34 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.

6.6 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: Figure 6.26, Figure 6.27, and Figure 6.28. This example distributes the Elements 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.

6.6.1 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 6.15. 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.

Figure 6.34. P2P_1 SOSI Alternative Top Level CPN Representation
Figure 6.35. P2P_1 Port Node CPN

Figure 6.36. P2P_1 SOSI Ground Station Node CPN

Figure 6.37. P2P_1 SOSI Tarawa Node Colored Petri Net
Figure 6.38. P2P_1 SOSI Harper's Ferry Node CPN

Figure 6.39. P2P_1 SOSI Beach Node CPN
Table 6.15. Cohesion SOSI P2P_1

<table>
<thead>
<tr>
<th>Figure</th>
<th>Node</th>
<th>Name</th>
<th>I</th>
<th>Q</th>
<th>x</th>
<th>z</th>
<th>Coh</th>
</tr>
</thead>
<tbody>
<tr>
<td>Figure 6.35</td>
<td>1,1</td>
<td>Port</td>
<td>3</td>
<td>1</td>
<td>3</td>
<td>3</td>
<td>1.0</td>
</tr>
<tr>
<td>Figure 6.36</td>
<td>1,2</td>
<td>Ground Station</td>
<td>3</td>
<td>1</td>
<td>3</td>
<td>3</td>
<td>1.0</td>
</tr>
<tr>
<td>Figure 6.37</td>
<td>1,3</td>
<td>Tarawa</td>
<td>12</td>
<td>2</td>
<td>24</td>
<td>66</td>
<td>2.75</td>
</tr>
<tr>
<td>Figure 6.38</td>
<td>1,4</td>
<td>Harper’s Ferry</td>
<td>8</td>
<td>5</td>
<td>40</td>
<td>35</td>
<td>.88</td>
</tr>
<tr>
<td>Figure 6.39</td>
<td>1,5</td>
<td>Beach</td>
<td>10</td>
<td>4</td>
<td>40</td>
<td>54</td>
<td>1.35</td>
</tr>
<tr>
<td>Figure 6.40</td>
<td>1,6</td>
<td>Satellite</td>
<td>3</td>
<td>14</td>
<td>42</td>
<td>14</td>
<td>.33</td>
</tr>
<tr>
<td></td>
<td>SOSI</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1.22</td>
</tr>
</tbody>
</table>
### 6.6.2 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 6.16. 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 6.16. SOSI P2P_1 Coupling Results**

<table>
<thead>
<tr>
<th>Node</th>
<th>Port</th>
<th>Ground Station</th>
<th>Tarawa</th>
<th>Harper’s Ferry</th>
<th>Beach</th>
<th>Satellite</th>
<th>Coupling</th>
</tr>
</thead>
<tbody>
<tr>
<td>Port</td>
<td>x</td>
<td>15</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Ground Station</td>
<td>20</td>
<td>x</td>
<td>50</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Tarawa</td>
<td>30</td>
<td>x</td>
<td></td>
<td></td>
<td>30</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Harper’s Ferry</td>
<td>40</td>
<td>30</td>
<td>50</td>
<td>x</td>
<td>10</td>
<td>10</td>
<td>140/15</td>
</tr>
<tr>
<td>Beach</td>
<td>2</td>
<td></td>
<td></td>
<td></td>
<td>x</td>
<td></td>
<td>2/15</td>
</tr>
<tr>
<td>Satellite</td>
<td>2</td>
<td></td>
<td></td>
<td></td>
<td>x</td>
<td></td>
<td>2/15</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>S0SI</td>
</tr>
</tbody>
</table>
6.6.3 Degree of Reuse and Exclusiveness

The Degree of Reuse calculations for the P2P SOSI group appear below. Table 6.17 and Table 6.18 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 6.17. 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 | x |
| ISR | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x |
| GRN | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x |
| NGO | x | x | x | x | x | x | x | x | x | x | x | 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 6.18. P2P Degree of Reuse and Exclusiveness Results

<table>
<thead>
<tr>
<th>Degree of Reuse</th>
<th>Avg Element</th>
<th>High Reuse</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>2.08</td>
<td>4.00</td>
</tr>
<tr>
<td>Exclusiveness</td>
<td>0.48</td>
<td></td>
</tr>
</tbody>
</table>
6.7 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.

6.8 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.

Figure 6.41 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 Figure 6.40. 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 that the Nodes of the other P2P alternatives. This reinforces the claim that Degree of Reuse affects the overall Agility of the SOSI.

Figure 6.42 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 6.41. P2P SOSI Architecture Results
6.9 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.

Figure 6.43, Figure 6.44, and Figure 6.45 show a sample 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, Figure 6.46 and Figure 6.47 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 a Node with server Elements in the CS alternatives. Notice there is little communication directly among 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 6.48 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 reduced 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 6.43. CS1 SOSI Group Adaptability

Figure 6.44. CS2 SOSI Group Adaptability

Figure 6.45. CS3 SOSI Group Adaptability
Figure 6.46. High Cohesion Node Example CS Architecture
Figure 6.47. CS Low Cohesion Node

Figure 6.48. CS SOSI Architecture Adaptability
6.10 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 Figure 6.49, Figure 6.50, and Figure 6.51 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.

6.11 OVERALL RESULTS

This section compares the assessed Adaptability and Agility of the SOSI groups. Figure 6.54 shows Adaptability for all the SOSI alternatives. The results for CS are clearly lower than the P2P and SOA alternatives.

Figure 6.55 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. 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, Figure 6.56, 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 6.53 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 Coupling between the Nodes; the end result being higher Adaptability.
Figure 6.49. SOA1_1 SOSI Adaptability

Figure 6.50. SOA2_1 SOSI Adaptability
Figure 6.51. SOA3_1 SOSI Adaptability

Figure 6.52. SOA SOSI Architecture Adaptability
Figure 6.53. Impact of Highly Reused Elements
Figure 6.54. Case Study Adaptability Results

Figure 6.55. Case Study Exclusiveness Results
6.12 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 unpredictable 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. Figure 6.57 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 6.57. 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.
CHAPTER 7
CONCLUSION

7.1 SUMMARY

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). The 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 reveals
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.

7.2 CONTRIBUTIONS

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.

7.3 **FUTURE WORK RECOMMENDATIONS**

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 reduce the ambiguity currently in the UML specification.
REFERENCES
REFERENCES


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.


JCS(2007), Joint Chiefs of Staff, Dictionary of Military and Associated Terms, Department of Defense, Editor. 2007, Joint Chiefs of Staff.


CURRICULUM VITAE

Stewart W. Liles graduated from Woodward High School, Woodward, Oklahoma in 1984. He received his Bachelor of Science in Management Sciences and Computer Systems from Oklahoma State University, Stillwater, Oklahoma, in 1988. He then completed 10 years of service in the U.S. Army. He received a Master of Science in Modeling, Virtual Environments and Simulation from the Naval Post-Graduate School, Monterey, California, in 1998.