RESILIENCE OF ENTERPRISES IN A CONTESTED CYBER-ENVIRONMENT

by

Bahram Yousefi
A Dissertation
Submitted to the
Graduate Faculty
of
George Mason University
in Partial Fulfillment of
The Requirements for the Degree
of
Doctor of Philosophy
Systems Engineering and Operations Research

Committee:

_________________________________________ Dr. Alexander H. Levis
Dissertation Director

_________________________________________ Dr. Lance Sherry, Committee Member

_________________________________________ Dr. Abbas K. Zaidi, Committee Member

_________________________________________ Dr. Paulo Costa, Committee Member

_________________________________________ Dr. Ariela Sofer, Department Chair

_________________________________________ Dr. Kenneth S. Ball, Dean, Volgenau School
of Engineering

Date: ________________________________ Summer Semester 2017
George Mason University
Fairfax, VA
Resilience of Enterprises in a Contested Cyber-Environment

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

by

Bahram Yousefi
Master of Science
Azad University, 2008
Bachelor of Science
Azad University, 2005

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

Summer Semester 2017
George Mason University
Fairfax, VA
DEDICATION

This is dedicated to my loving parents who instilled the passion for knowledge in me, my supportive brother, Babak, and my benevolent sister, Shadi.
ACKNOWLEDGEMENTS

I would like to thank Professor Levis for his mentorship without which this work was not possible. Moreover, I want to express my appreciation to my committee members Dr. Zaidi, Dr. Sherry, and Dr. Costa for their unwavering support and splendid guidance throughout my studies. Furthermore, I extend my gratitude to Professor Sofer, SEOR Department’s Chair, for sponsoring me throughout the years. Finally, I must value the great and meticulous administrative assistance of SEOR department’s staff, Angel Manzo and Josefine Wiecks throughout my work.
# TABLE OF CONTENTS

List of Tables ................................................................................................................... viii
List of Figures .................................................................................................................... ix
List of Abbreviations ......................................................................................................... xi
Abstract ............................................................................................................................. xii

## Chapter 1: Introduction .................................................................................................. 1
1.1 Motivation ............................................................................................................ 3
1.2 Problem Statement ............................................................................................... 3
1.3 Hypothesis ............................................................................................................ 5
1.4 Original Contributions.......................................................................................... 5

## Chapter 2: Related Work ............................................................................................ 6
2.1 Architecture Design.............................................................................................. 6
2.2 Architecture Evaluation........................................................................................ 7
2.3 Colored Petri Nets ................................................................................................ 8
   2.3.1 Discrete Event Model of a Queueing Network ............................................. 8
   2.3.2 Multi-Formalism Technique ......................................................................... 9
   2.3.3 Meta-modeling .............................................................................................. 9
   2.3.4 High Level Architecture ............................................................................. 10
   2.3.5 Hybrid Approach ........................................................................................ 13
2.4 Decision Theory ................................................................................................. 13
2.5 Organization Theory ............................................................................................ 15
2.6 Team Theory ....................................................................................................... 17
2.7 Adaptive Enterprises .......................................................................................... 18
2.8 Resilience ........................................................................................................... 19
2.9 Contested Cyber-environment............................................................................ 20

## Chapter 3: Resilience of an Architecture ................................................................... 22
3.1 Framing Resilience............................................................................................... 22
3.2 Conceptual Resilience Framework
   3.2.1 Capacity
   3.2.2 Tolerance
   3.2.3 Flexibility
   3.2.4 Inter-element Collaboration
3.3 Quantifying the Resilience Framework
   3.3.1 Measures of Capacity
   3.3.2 Measures of Tolerance
   3.3.3 Measures of Flexibility
   3.3.4 Inter-element Collaboration Architectural Description
3.4 Resilience to a Particular Cyber Perturbation

Chapter 4: Enterprise Design
   4.1 Organizational Design
      4.1.1 Five-stage model of Interacting Decision-makers
      4.1.2 Mathematical Representation of an Enterprise
      4.1.3 Lattice Theory
      4.1.4 Task allocation
   4.2 Adaptive Enterprise
      4.2.1 Anomaly Detection
      4.2.2 Execution

Chapter 5: Methodology For Designing Resilient Enterprises
   5.1 Enterprise Architecture
      5.1.1 Verifying the Enterprise Architecture
   5.2 Enterprise of DMs
   5.3 Resilience of the Elements of the Lattice
   5.4 Multi-model construction
      5.4.1 Discrete Event Model of an Enterprise
      5.4.2 Discrete Event Model of the Queuing Network
      5.4.3 Synthesizing the Multi-model
   5.5 Adaptive Enterprise
      5.5.1 Anomaly Detection

Chapter 6: Conclusions and Future Work
6.1 Conclusions ...................................................................................................... 119
6.2 Future Work ..................................................................................................... 121
Appendix: Architecture Design and Evaluation ................................................... 123
  Architecting Process ........................................................................................... 123
  Architecture Design ............................................................................................ 125
  Executable Model Construction ....................................................................... 129
  Architecture Evaluation ...................................................................................... 132
References ............................................................................................................. 134
### LIST OF TABLES

<table>
<thead>
<tr>
<th>Table</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Table 1 Statistical results of generating the state space of the partial FAA's architecture</td>
<td>88</td>
</tr>
<tr>
<td>Table 2 Simple paths of organizational structure B. The top row numbers show the node counts and the three rows below the first row denotes the elements in each simple path. Since the simple path are very long they are broken into shorter lines of 15 elements each</td>
<td>92</td>
</tr>
<tr>
<td>Table 3 Simple paths of organizational structure A. The top row numbers show the node counts and the three rows below the first row denotes the elements in each simple path. Since the simple path are very long they are broken into shorter lines of 15 elements each</td>
<td>93</td>
</tr>
<tr>
<td>Table 4 Summary of graph theoretic metrics for organizational structures A and B</td>
<td>97</td>
</tr>
<tr>
<td>Table 5 Implementations of cyber perturbations</td>
<td>105</td>
</tr>
<tr>
<td>Table 6 IAD Departure and Arrival Real-time data</td>
<td>110</td>
</tr>
<tr>
<td>Table 7 Summary of performance related resilience metric</td>
<td>113</td>
</tr>
<tr>
<td>Table 8 Method of synthesizing CPN model of an Object Oriented Architecture</td>
<td>132</td>
</tr>
</tbody>
</table>
## LIST OF FIGURES

<table>
<thead>
<tr>
<th>Figure</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Figure 1 Different Components of High Level Architecture</td>
<td>11</td>
</tr>
<tr>
<td>Figure 2 Runtime Infrastructure in HLA</td>
<td>11</td>
</tr>
<tr>
<td>Figure 3 High level design of C2 Wind Tunnel</td>
<td>12</td>
</tr>
<tr>
<td>Figure 4 Temporal aspects of resilience</td>
<td>20</td>
</tr>
<tr>
<td>Figure 5 Temporal view of resilience [3]</td>
<td>28</td>
</tr>
<tr>
<td>Figure 6 Capacity attribute of resilience framework (reactive, buffering, and residual capacities)</td>
<td>29</td>
</tr>
<tr>
<td>Figure 7 Two hypothetical organizations: a) organization without interactions between DMs and b) illustrative organization with interaction between DMs</td>
<td>33</td>
</tr>
<tr>
<td>Figure 8 Three simple paths of the illustrative example (organization b)</td>
<td>34</td>
</tr>
<tr>
<td>Figure 9 Five-stage model of intelligent nodes</td>
<td>45</td>
</tr>
<tr>
<td>Figure 10 Interaction between intelligent nodes</td>
<td>47</td>
</tr>
<tr>
<td>Figure 11 Concept map of role</td>
<td>56</td>
</tr>
<tr>
<td>Figure 12 Proposed adaptive enterprise system</td>
<td>58</td>
</tr>
<tr>
<td>Figure 13 Information Fusion stage</td>
<td>63</td>
</tr>
<tr>
<td>Figure 14 Concept map of design and modeling approaches used in this research effort</td>
<td>65</td>
</tr>
<tr>
<td>Figure 15 Partial sequence diagram of air traffic control system</td>
<td>67</td>
</tr>
<tr>
<td>Figure 16 Top-page IDEF0 model of Air Traffic Control System [66]</td>
<td>69</td>
</tr>
<tr>
<td>Figure 17 Partial class diagram of ATC enterprise</td>
<td>71</td>
</tr>
<tr>
<td>Figure 18 Activity model of ATC enterprise (top half)</td>
<td>72</td>
</tr>
<tr>
<td>Figure 19 Activity model of ATC enterprise (bottom half)</td>
<td>73</td>
</tr>
<tr>
<td>Figure 20 Partial process thread of issuing flight plans</td>
<td>74</td>
</tr>
<tr>
<td>Figure 21 Partial class diagram of the reconstructed FAA's Enterprise Architecture</td>
<td>75</td>
</tr>
<tr>
<td>Figure 22 FAA’s TRACON and ATCT classes in standard form</td>
<td>76</td>
</tr>
<tr>
<td>Figure 23 Process for synthesizing the executable model of the OO architecture</td>
<td>77</td>
</tr>
<tr>
<td>Figure 24 Top-page of the hierarchical executable model of FAA's EA</td>
<td>78</td>
</tr>
<tr>
<td>Figure 25 Partial sub-page of a substitution transition</td>
<td>79</td>
</tr>
<tr>
<td>Figure 26 Allocation of tasks to intelligent nodes</td>
<td>80</td>
</tr>
<tr>
<td>Figure 27 Physical viewpoint of the ATC enterprise (Org. A)</td>
<td>81</td>
</tr>
<tr>
<td>Figure 28 Petri Net model of the physical model Figure 27</td>
<td>83</td>
</tr>
<tr>
<td>Figure 30 Constructing the overarching executable model</td>
<td>84</td>
</tr>
<tr>
<td>Figure 29 Physical viewpoint of the ATC enterprise (Org. B)</td>
<td>84</td>
</tr>
<tr>
<td>Figure 31 Top-page of ATC's executable model</td>
<td>85</td>
</tr>
</tbody>
</table>
Figure 32 Partial State Space of the FAA's Architecture. Ill-terminated states are in bold.......................................................................................................................... 87
Figure 33 Subpage of ATCT entity ............................................................................................................................................... 89
Figure 37 Graphical representation of the lattice set of organization B ........................................................................................................... 94
Figure 38 Graphical representation of the lattice set of organization A ................................................................................................. 95
Figure 39 Graphical representation of the truncated lattice set of organization B .................................................................................. 96
Figure 41 The model of constructing the executable model of the enterprise .............................................................................. 101
Figure 42 The lattice model of an Enterprise .......................................................................................................................... 102
Figure 43 Compound module of each terminal in OMNeT++ .............................................................................................................. 103
Figure 44 The top-page and a few sub-pages of the network model ................................................................................................. 104
Figure 41 A temple for cyber perturbation in OMNeT++ ............................................................................................................ 106
Figure 42 Affected network asset by cyber perturbation .................................................................................................................. 107
Figure 43 Obtaining markings of places in Access/ CPN .................................................................................................................. 108
Figure 44 Pseudo code of the run-time infrastructure .................................................................................................................. 108
Figure 45 Colored Petri Nets and OMNeT++ integration .................................................................................................................. 109
Figure 46 Parametric Distribution is fitted to the inter-arrival time of IAD real-time data .................................................................................................................. 111
Figure 47 Average number of flight plans issued by organization A in each hour of a day .................................................................................................................. 112
Figure 48 Average number of flight plans issued by organization B in each hour of a day .................................................................................................................. 113
Figure 49 Role Definition Matrix (RDM) for the to-be organizational design .................................................................................. 115
Figure 50 State space of the structural forms of the adaptive organization (to-be) .................................................................................. 115
Figure 51 Anomaly Detection Triggers Structural Switch .............................................................................................................. 117
Figure 52 Average number of flight plans issued by organization C in each hour of a day .................................................................................................................. 118
Figure 53 Architecture Development Process [7].................................................................................................................. 125
Figure 54 A top-down view of evaluation process .................................................................................................................. 126
Figure 55 Style constraint: Decomposition of classes .................................................................................................................. 130
LIST OF ABBREVIATIONS

Air Route Traffic Control Center ............................................................................. ARTCC
Air Traffic Control System ......................................................................................... ATCS
Air Traffic Control System Command Center ....................................................... ATCSCC
Air Traffic Control Tower .......................................................................................... ATCT
Colored Petri Net ........................................................................................................... CPN
Decision-Maker ............................................................................................................... DM
Discrete Intelligence System........................................................................................... DIS
Discrete Intelligence System........................................................................................... DIE
Enterprise Architecture ............................................................................................... EA
Federal Aviation Administration ................................................................................... FAA
Flight Service Station ...................................................................................................... FSS
Maximally Connected Organization ......................................................................... MAXO
Measure of Effectiveness ............................................................................................... MoE
Measure of Performance ............................................................................................... MoP
Minimally Connected Organization ............................................................................ MINO
Modular Network Test-bed in C++ ........................................................................ OMNeT++
National Airspace System .............................................................................................. NAS
Pre-departure Clearance ............................................................................................... FAA
Role Definition Matrix ................................................................................................. RDM
Run-Time Infrastructure ................................................................................................. RTI
Terminal Radar Approach Control ........................................................................... TRACON
Weather Observation Station ......................................................................................... WOS
ABSTRACT

RESILIENCE OF ENTERPRISES IN A CONTESTED CYBER-ENVIRONMENT
Bahram Yousefi, Ph.D.
George Mason University, 2017
Dissertation Director: Dr. Alexander H. Levis

Assessing the resilience of an enterprise that is supported by a queuing communication network to a particular cyber disruption is an extremely complex problem, but also a very important one due to the rapid increase in both the number and the sophistication of cyber exploits in recent years. There are many questions that must be answered; this research effort investigates the influence of the structural form of the enterprise on its resilience to availability type cyber disruptions. In the availability type disruption, a locality of the supporting cyber environment is rendered slow, in order to make the holistic supporting network less effective. The result of this study is a methodology to identify the structural form(s) that provide resilience.

The solution space is a lattice of the structural forms of the enterprise that is derived through a sequential iterative process from the enterprise architecture. A five-stage model of intelligent nodes is used to construct the executable model of the distributed intelligence enterprise (DIE). The elements of the lattice of the DIE are explored for their resilience and latency properties.
Furthermore, an adaptive organization is constructed to compare its resilience and latency attributes to other organizational structures. The adaptive organization has a default structure with a minimally connected organizational form and secondary structure(s) with maximally connected organizational structure(s) that are adopted when an anomaly is detected. This kind of organizational design provides a balance between the resilience and latency of the DIE. This is the property that non-adaptive organizational structures lack, thus, the added value of adaptive organizations is the balance it provides between resilience and latency.

The Federal Aviation Administration’s Enterprise Architecture of Air Traffic Control System is used to measure some of the attributes of resilience, to illustrate the theoretic findings, and to implement and test the adaptive model of the enterprise.

The main findings of this research effort are summarized as follows. First, adding structure (e.g. a simple path) will improve the resilience from a structural perspective. Second, adding structure introduces additional delay to the system. Consequently, the performance (delay) and the structural attributes of the resilience of an enterprise work against one another. Finally, adaptive enterprise structures provide a balance between the performance (delay) and the structural attributes of resilience.
CHAPTER 1: INTRODUCTION

During the past two decades, we observed a rapid proliferation of cyber exploits. In the meantime, there is a continuous increase in the complexity of these exploits. As the recent threat report published by Symantec Corporation [1] concluded, threats are growing at a much faster rate than the cyber defense technologies. Therefore, we cannot imagine a near-future that assures defense solutions against cyber exploits. As a result, we expect to have to operate in the presence of cyber threats. Fortunately, there exist substantial knowledge and experience in various engineering fields that can be leveraged to face cyber exploits.

Systems that are able to endure and recoup from disruption are called resilient. The notion of resilience is a characteristic of systems and some systems are more resilient to certain kinds of disturbances than others. This attribute of the systems has been under investigation in different engineering disciplines. For example in civil engineering, scholars are concerned with the resilience of buildings to earthquakes. As a result, they have identified particular structural patterns and specifications that are responsible for the holistic resilience of a building in the event of an earthquake. Consequently, they leveraged these patterns and specifications to improve the resilience of the building. This is a widely used analogy for the notion of resilience and in practice it has been a successful approach which could save lives and reduce property damage.
System engineers are interested in identifying the properties that are responsible for resilience of certain kinds of systems to particular types of disturbances and then harness them in order to improve the resilience of the system. In recent years, an interest in designing, developing, and deploying resilient systems developed as the result of increased awareness about the benefits of this salient characteristic of the systems. This phenomenon initiated many research efforts about this issue.

The concept of resilience is both domain-specific and context-specific and therefore it needs a precise definition and formulation. As a result, we must answer some questions as below:

− What are the measures of resilience in the domain?
− What is the disruption of interest in the context?
− What is the model of the disruption in the context?
− What is the model of the disruption in the domain?
− What are the system’s requirements in the domain?

By looking at these questions, we can identify a significant challenge ahead of us for analyzing the resilience of a system. The disruption may happen in the external environment of the system, which may have a different formalism or even an unknown formalism. In addition, we need to model the disruption in the system’s domain. Ultimately, we want to investigate the dynamics of change in the system’s quality in the presence of a disruption. A system’s quality is measured with performance and effectiveness metrics.
In this research effort, we are only interested in distributed intelligence organizations (such as an air traffic control system) that are supported by a queueing communication network (e.g., cloud) for their members’ interactions. This type of organizations is mission critical, i.e., they must accomplish their mission in a high tempo environment and according to a preplanned schedule. Furthermore, these organizations are working with authoritative data and thus integrity of information is of high importance. From the discussion so far, we can envision the devastating impact of a complex cyber exploit on the organization’s missions. We will refer to a cyberspace whose capability can be denied, degraded, disrupted or destroyed as a contested cyber-environment [2]. Therefore, there is a necessity in a contested cyber-environment to design, develop, and deploy resilient decision-making organizations.

1.1 Motivation
This dissertation is inspired by previous works of Pflanz on a resilience framework [3], of Perdu on adaptive enterprises [4], of Remy on distributed intelligent organizations [5], and of Shin on multi-modeling [6]. The penultimate goal of this work is to expand the understanding of cyber resilience of enterprises, and the ultimate goal is to enhance protection against cyber exploits at the enterprise level.

1.2 Problem Statement
The exploitation of the communication network infrastructure of a decision-making organization will impact its performance and effectiveness. This cyber exploitation may degrade the ability of the organization to perform activities and may hinder it from achieving its goals. However, it has been observed that the degradation trajectory of
different organizations may vary in the face of a certain cyber exploit. As a result of this, one organization may be able to perform its activities in a degraded fashion while another organization under similar conditions completely fails to perform. The shape of degradation trajectory defines the resilience characteristic of a system.

We hypothesize that the arrangement of the information flow paths and the assignment of tasks in an organization will affect its resilience in the presence of cyber exploits. The information flow paths are structural features of an organizational architecture and therefore the (architectural) design of the organization will influence its resilience.

Due to the complexity of this problem, we must use a multi-formalism approach to model both the domain, i.e., decision-making organizations, and the environment, i.e., the queuing communication system. The multi-model will enable us to simulate and then analyze the impact of a certain cyber exploit on the performance and effectiveness of an organization that has a particular architecture.

The purpose of this research effort is to articulate a methodology to design more resilient decision-making organizations. This methodology concerns the architectural attributes that are responsible for the resilience of a decision-making organization. In addition, this process enables the investigation of the impact of different architectural structures on the resilience of a particular organization through simulation. The scope of our effort is decision-making organizations and we are interested in their resilience in a contested cyber-environment as defined in [2].
1.3 Hypothesis
We postulate that the interaction style among the decision makers (DMs) of each particular team in the decision-making organization will influence its resilience in a contested cyber-environment. In other words, the pattern of information flow paths between DMs will influence the resilience of the organization in the case of disruption with specific characteristics in a certain locality of the information network. In order to test this hypothesis, we have to evaluate first the impact of a specific cyber exploit on the organization’s performance and effectiveness. Next, we will test different interaction styles (structural styles) to evaluate the resilience of the organization under each arrangement and then identify the architectural features that contribute to resilience.

1.4 Original Contributions
The original contributions of this research effort are as follows:
- Articulating a methodology to design more resilient distributed intelligence organizations
- Developing a novel multi-formalism testbed for analysis
- Constructing a state of the art executable model of the air traffic control system
- Providing an improved understanding of resilience in the context of distributed intelligence organization
CHAPTER 2: RELATED WORK

2.1 Architecture Design
Levis and Wagenhals [7] developed a methodology for architecture design, which is compliant with the C4ISR architectural framework. The C4ISR framework later was renamed as the DoD architectural framework (DoDAF). In addition, Wagenhals et al. [8] provided a structured analysis approach for architecture development. The presented method is a comprehensive and traceable approach that defines the process of generating all the architectural artifacts. Similarly, Bienvenu et al. [9] proposed an Object Oriented (OO) approach for architecture development. The OO paradigm is getting lots of attention because of its various capabilities. Just as Bienvenu et al. devised their OO process using the Unified Modeling Language (UML), we shall use the same technique in our research effort. Moreover, Wagenhals et al. [10] extended their process to convert the architectural artifacts to executable models and also provided some guidance on defining and calculating measures of performance (MOPs) and measures of effectiveness (MOEs).

When it comes to executable models construction, Wagenhals et al. [10] proposed a comprehensive methodology to synthesize the executable models from the OO UML-based architectural artifacts. This method guarantees concordance among all the architectural products such as activity, collaboration, sequence, and class diagrams. Furthermore, static architectural elements can be traced back to their corresponding
executable model’s attributes and vice versa. The traceability between the static and the dynamic models is the foundation of architectural analysis and evaluation.

However, one view not covered by any of these authors is the structural arrangement of an organization. Alberts in [11] discussed the importance of these structural configurations on an organization’s properties and then provided some examples of common structural styles. Similar to Wagenhals et al. [8], AbuSharekh et al. [12] described the process of deriving executable models from DoDAF artifacts. In addition, they explicitly included the notion of time in their executable models, which enhances the analytic capabilities of the dynamic models.

2.2 Architecture Evaluation

There is a large body of literature about the performance and effectiveness evaluation of decision-making organizations. Valraud and Levis [13] presented a methodology for comparing the functionality of a C³ system against its desired requirements using the invariant analysis of Petri Net. Further, Cothier and Levis [14] provided an in-depth definition of measures of performance (MOPs) and measures of effectiveness (MOEs) and then they provided an illustrative example of these concepts. Moreover, Bouthonnier and Levis [15] articulated an approach for analyzing and assessing the effectiveness of the C³ systems. Consequently, a handful of case studies about the architecture evaluation and analysis using these methodologies were conducted [6], [16]–[18].

Using a similar approach, Grevet and Levis introduced [19] the notion of coordination in decision-making organizations. They proposed two MOPs for coordination
in a decision-making organization: degree of information consistency and degree of synchronization. Moreover, they provided two other measures: accuracy and timeliness. These MOPs are required to capture the organization’s performance and thus they are complementary to the coordination measures. We will cover the mathematical models of their work in section 4.2.1.

2.3 Colored Petri Nets
Colored Petri Net (CPN) [20] is a formal language with a rich mathematical notation. It is formal because it has syntax, semantics, well-formed formulas, and theorems. In addition, this paradigm renders a graphical representation of the models under execution. CPN is widely used in modeling business processes, discrete event systems, scheduling systems, etc. mainly because of its powerful graphic features. By using traceable conversion methods, we can transform a model in an informal paradigm like UML into a formal Petri Net design. The CPN development environments, e.g., CPNTools [21], provide various types of analysis such as temporal logic, state space, and invariant analysis.

2.3.1 Discrete Event Model of a Queueing Network
In this work, we will use Object Modular Network Testbed in C++ (OMNeT++) [22] as our preferred simulation environment. Basically, OMNeT++ is a discrete event network simulator in which the models are created in a hierarchical fashion. These models are composed of two types of modules: simple modules and compound modules. A simple module is an atomic entity and two or more atomic modules together form a compound module. These two types of modules are constructed using a stylized C++ code structure, which is very versatile, and thus they may be leveraged to produce some complex behavior
of interest. In addition, we can create customized message structures, which will be passed among related modules. Besides the command-line interface, OMNeT++ provides a multipurpose Graphical User Interface (GUI) that enables editing of the network topologies, coding of the modules, setting execution parameters, and rendering graphic demonstration of the runtime. In order to create a packet level Internet Protocol (IP) network, we use an additional framework called INET Framework [23], which enables the modeling of wired, wireless, and mobile networks.

2.3.2 Multi-Formalism Technique
Multi-formalism is catching a lot of attention due to the increased level of complexity in systems engineering in recent years. These complex systems are developed by engineers from multiple disciplines who use domain-specific paradigms to design, fabricate, and test each single component of the system. Therefore, we can imagine the level of complication involved in validating and evaluating such complex systems as a whole.

Multi-formalism is a well-established theory and thus the authors in [24] discussed the different aspects of it in great detail. For example, Gribaudo and Iacono [25] provided the definitions, classifications, and conceptual problems such as model representation, solution strategies, and model composition.

2.3.3 Meta-modeling
Multi-formalism theory has been challenged for its formality. The salient question is whether or not a multi-model is formal, i.e., if it has a well-defined syntax and semantics. This is an intricate issue that Abu Jbara and Levis studied in [26]. They provided the
solution of a meta-model of the meta-models or in their terminology, workflow language for ensuring the formality of the multi-model.

### 2.3.4 High Level Architecture

High Level Architecture (HLA), which describes the regulations of interoperations between the different environments, is the standard architecture for integrating various simulation environments [27]. In other words, HLA furthers the federation of distributed simulation systems. IEEE manages the HLA standard, and also publishes the necessary documents and datasheets [28].

According to the most recent datasheet published by IEEE [28], HLA has two core components: first, the Object Model Template (OMT), which defines the data structure used by each simulation system, enables the reuse of information structures, and monitors the conversion between different structures; second, the Federate Interface Specification (FIS) defines a standard communication protocol that enables the interoperation between simulation environments. In the context of FIS, a distributed simulation environment is called a federation and each single simulation system is referred to as a federate. As we can see in Figure 1, a federate could be a simulation, a live player, or a passive viewer. One should note that participants are either computerized systems or humans.

Despite the fact that HLA is an architecture, it requires a runtime infrastructure (RTI) to execute the federation (see Figure 2). RTI provides the services and the capabilities, as defined in the FIS, to enable federates’ interoperations and exchange of messages throughout a runtime. Currently, there are multiple commercial, academic, and open source software implementations of RTI available. For example, poRTIco RTI [29]
is an open source product, which supports both Java and C++, originally developed by Calytrix Technologies and is compliant to the IEEE 1516-2000 standard [28]. The poRTIco provides the standard services and methods through the Application Programming Interfaces (API) for both Java and C++.

Although software products, like poRTIco, provide the RTI services, they fall short on integration of federates. Therefore, Ledeczi et al. [30] presented the generic modeling
environment (GME) as a toolset for creation of domain-specific federates and syntheses of federations. A significant feature of GME is that it enables rapid synthesis of different COTS analysis tools. Later, Abu Jbara and Levis [26] investigated the syntactic and semantic correctness of interoperating federates.

All of these advances are used in an HLA implementation called C2 Wind Tunnel (C2WT) [31], [32]. The C2WT is a computational testbed that realizes multi-modeling by integrating multiple modeling paradigms. Therefore, it can provide us with multiple insights about a situation, from different angles. Figure 3 shows high level design of the C2WT.

Figure 3 High level design of C2 Wind Tunnel
2.3.5 Hybrid Approach

Hybrid-modeling is a similar technique to multi-modeling but with one main difference. Hybrid approaches do not explicitly require live execution, while it is mandatory in multi-modeling. Therefore, many hybrid models are working offline, i.e., one model is executed and the resulting data is stored and then this data will be fed to another model while running and so forth. Shin and Levis [6] leveraged the hybrid-modeling in their experiment. They executed in an offline configuration two models: a) A network model developed using the ns-2 [33] design environment and b) an information system modeled by Petri Nets.

2.4 Decision Theory

Decision theory is concerned with goal-oriented decisions in a game of choice [34]. There are two schools of thought in decision theory: 1) descriptive theory explains the rationale behind a human decision maker’s (DM’s) decision, and 2) normative theory finds the optimal decision in a specific situation. However, the normative approach fails to explain many human DM’s decisions because, as many research efforts [34], [35] have suggested, normally human DMs tend to look for a satisficing choice and not necessarily an optimal one. Also, it has been shown that DMs do not attempt to meet all the requirements [34], [35]. They sample among the criteria and only consider those that are perceived to be important in the prevailing scenario. Therefore, human DMs only partially fulfill the criteria due to their bounded rationality.

The descriptive decision theory is constructed based on the decision process, which identifies the stages of decision-making and their interconnection. There exist two categories of decision processes: the sequential, and non-sequential. The sequential model
of decision making has a long history, which dates back to the French philosopher Condorcet in the 1790’s [36]. He introduced a three stage decision model: first, the initial discussion about the principles of the problem to be solved and at this stage the DMs present their personal perspectives. Following the first discussion, there is a second discussion to form a majority view of the problem and also devise a tractable set of solutions. The last step is to find the most appropriate solution to the problem. Condorcet’s model of decision making was later replaced by modern approaches.

Simon conceived a widely discussed sequential model of the decision process [37]. His decision making model contains three stages: intelligence, design, and choice. Simon used the notion of intelligence in its military definition, which means identifying the decision junctures through continuous monitoring. DMs will create a set of possible courses of action (COA) in the design phase. Finally, DMs make a selection among COAs, which are designed in the previous phase, i.e., the choice stage. In addition, Brim et al. [38] proposed their sequential decision making model, which follows the same terminology as Simon’s.

Both the models of Brim et al. and Simon lack the notion of interaction among DMs. As a result Boettcher and Levis [39], [40] introduced the model of interacting DMs with memory. The model consists of four phases, which later was enhanced to five phases [41]. There is a large body of literature about different aspects of their model. For example, Tabak and Levis [42] presented the Petri Net implementation of the decision making model as well as a proof of bounded rationality. Also, Remy and Levis [43] defined the rules that
govern the interactions among DMs. Moreover, many researchers developed applications according to this proposed model [44]–[48].

Mintzberg et al. proposed an eminent non-sequential model of the decision process [49]. They used the same concepts as Simon [10] but with different names: identification, development, and selection. However, in this non-sequential model, stages are enabled in an arbitrary manner unlike the sequential models. As a result, this method cannot be implemented using available technologies today.

2.5 Organization Theory

March and Simon are two foremost exponents of organization theory. They presented the foundation of this theory in their book “Organizations” [50]. Their work is far-reaching and as a result, we will only cover pertinent components of their work. They began the theory by providing the distinctive features of formal organizations, of which the main ones are as follows: 1) members of an organization share a common technical vocabulary, which is usually gained through rigorous past training, 2) members in an organization usually share a special contextual assessment, which enables them to anticipate their colleagues’ trains of thought, 3) each member occupies a defined role in the organization. The notion of role is a pivotal concept in the theory of organization because it has a diverse impact on the whole organization and not only on the member who is assigned to the role. For example, members of the organization are also familiar with the roles of other members and therefore they have certain expectations, i.e., they expect to receive certain pieces of information or things from other members in accordance with their roles, and 4) there is a high level of coordination among the members of an
organization and this is to the point that most of the interactions in an organization are preplanned and pre-coordinated. Furthermore, they elaborated on the two salient schools of thought in organizational theory: 1) Taylor’s method, which mainly involves the analysis of activities in the production line, focuses on improving the efficiency of a production line – which usually involves repetitive activities – through his three precepts: a) use the available attributes such as “skill”, “effort”, “durability”, “fatigue”, and so forth to perform the job in the “one best way”, b) give incentive to the members for doing the job in the best way, and c) consult with the experts in the field to establish an efficient working environment, i.e., methods, machine’s performance, schedules, etc., and 2) the Gulick and Urwick theory of departmentalization, the formal definition of which is the following: considering the mission of the organization, we can decompose the mission into unit tasks. These tasks include some atomic activities. The problem of departmentalization is to create jobs (roles) from a set of atomic activities and then combine a set of jobs in a unit (team) and lastly to aggregate the teams into the organization in a way that the operational cost is minimized.

March and Simon [50] discussed the concept of rationality of the “economic man”. Before going into the definition of rationality, we must explain the notion of “program” that was extensively used in the book. A “program” is the set of responses that an external stimulus evokes and will produce an end result. Based on the notion of “program”, a rational economical decision maker runs the program that will produce the optimal result. However, they have observed that in reality – especially in the complex organizations – programs produce only a satisfactory result and not an optimal one due to the fact that
human decision makers have bounded rationality. As a result of this limit on rationality, organizations leverage simplified models without capturing all the complexities.

Component level theories and concepts of organizational theory are covered in the theory of teams. The theory of teams elaborates on the process of decision-making in a team and also it provides mathematical models of the teams.

2.6 Team Theory

Team theory is a complementary idea to organization theory. Marschack and Radner [51] provided the comprehensive theory in their book “Economic Theory of Teams”. According to them, a team is an organization whose members do not have access to the same data but they have the same interests and beliefs. In addition, economic theory concerns the problem of finding the efficient use of scant resources. Consequently, the “economic theory of teams” deals with finding the arrangement of a team that is both viable and economical. In the context of decision-making teams, multiple persons, with the same interests and beliefs, are jointly performing tasks to achieve a common desired result. The tasks usually are gathering and communicating information and making decisions. Therefore, the team arrangement is the allocation of these tasks among team members in the most desirable form or at least in a viable manner.

Additionally, Marschak and Radner [51] presented mathematical models of teams. The salient components of the quantitative models are information structure, decision functions, and payoff functions. Information structure in an organization can be considered as the strategy that each DM applies to the state of the environment, which will result in a specific signal; all the possible alternative signals are known to DMs in a team. The
decision function interprets the signal to evaluate the state of the environment. Finally, the course of action with a maximum payoff will be selected by the payoff function as the team’s response.

2.7 Adaptive Enterprises

Stanton et al. [52] defined the concept of Systemic Teamwork, which is derived from the theory of departmentalization, in a decision-making organization. Systemic Teamwork requires adherence to rigid patterns and rules of teamwork and thus it suggests a static team structure. Some of these fixed schemas are assignment of roles to DMs, DMs’ network structure, coordination between DMs, communication between DMs, rankings of DMs, shared knowledge, and so forth. Although, Systemic Teamwork is simple to implement and use, it is not adaptive to the changes in the domain. Therefore, this kind of teams may suffer from performance degradation in abnormal conditions.

Perdu [4] introduced the theory of adaptive decision-making teams to address the rigidity of Systemic Teamwork terminology. According to Perdu, the necessity of adapting to the changes in the domain comes from the time constraint and also accuracy of decision making in a very uncertain situation. His method is based on a distributed – in contrast to the centralized approach which is commonly used – task execution. In an adaptive team, a DM may be assigned to one or more secondary tasks that he can perform, in addition to his primary task, as needed. It is usually the case that the DM is an expert in executing his primary task but he may not be as skilled in his secondary assignments. Therefore, a DM may do his secondary tasks at a degraded level of accuracy and with a slower pace. Realization of adaptive teams requires some additional pieces.
When some of the DMs in a team can perform multiple tasks, it can operate in different modes of operations – in the Systemic Teamwork, there is only one. Therefore, one main concept in adaptive team theory is the notion of the modes of operations, which in turn introduces more challenges. One major change of having more than one mode of operation is coordinating the different modes of operation, i.e., when to change the mode and to which mode we should switch. Further, Perdu and Levis [45] introduced the two notions of polymorphic DMs and morphing process. A polymorphic DM can perform many tasks and the process of transforming a DM from one morphism to another one is the morphing process. The morphing process ensures a certain level of performance during a mode change. A very good analogy of a morphing process is a locomotive which needs to be repaired while it is moving at a certain speed toward its destination. As a result, the onboard engineer must conduct the necessary repairs while maintaining a certain velocity and safety of the trains. Similarly, a morphing process transforms the different modes of operations of a decision-making organization while it is conducting its mission in a satisfactory manner.

2.8 Resilience
The International Council on Systems Engineering (INCOSE) established a dedicated working group on resilient systems, Resilient Systems Working Group (RSWG) [53]. This was a part of INCOSE’s effort for advancing the knowledge of resilient systems. RSWG defined the term “Resilience” as: “the capability of a system with specific characteristics before, during and after a disruption to absorb the disruption, recover to an acceptable level of performance, and sustain that level for an acceptable period of time.”
In this definition, we only consider the man-made systems comprised of hardware, humans, algorithms, processes, and so forth. Also, disruption is either a short-term or a long-term digression toward a state with degraded performance. The notion of “resilience” has been the topic of discussion in many articles such as [16], [54], [55]. We are especially interested in the work of Pfanz and Levis in [16], because it is focused on C² systems and therefore adheres to the conceptual elements of this field. Figure 4 Temporal aspects of resilience, illustrates the temporal aspects of a system’s resilience. In [16], they provided an extensive analysis of the survival phase and also defined four primary attributes of it. We will discuss these attributes and their formulation in chapter 3.

2.9 Contested Cyber-environment
A majority of organizations leverage Information and Communication Technology (ICT), together known as “Cyberspace”, to improve their operational efficiency and performance. As we should know, cyberspace or, as we call it in this context, cyber-
environment has manifold vulnerabilities. Therefore, hackers can take advantage of these weaknesses and sabotage its integrity. When the integrity of a cyber-environment or a locality of it is susceptible to breaches using any methods, we call it a contested cyber-environment.

Only by browsing through the corpus of cyber threat descriptions provided by internet security companies, e.g., Symantec Corporation, we can understand the prevalence and the complexity of these threats. For example Symantec Corporation in its recent Internet Security Threat Report (ISTR) [1], reported over 66,400 vulnerabilities over the past two decades. This means that a hacker can exploit these many options in his own arrangement for conducting his malicious operation to breach the integrity of a cyberspace of interest. Therefore, the number of cyber exploits is immense and thus we cannot deal with each one of them. Consequently, according to ISTR [1], the internet security community is trying to fix these known vulnerabilities, but we can imagine that overcoming these many vulnerabilities plus the newly discovered ones will take a long time. Consequently, it seems that we have to live with cyber exploits for now and hence we should design systems that can endure them.
CHAPTER 3: RESILIENCE OF AN ARCHITECTURE

As defined earlier, resilience is the capability of a system to absorb a particular kind of disruptions and recover afterwards. In this definition, disruption is considered to be a short-term deviation from the normal operational conditions; this is in relation to a system’s operational life, which means disruptions are rare events in comparison to the whole operational lifetime of the system. In addition, the characteristics of a system denote the properties of the disruption(s) that impact the system’s performance and effectiveness. At the same time, the characteristics of a system are linked to the resilience-related design elements that could be modified to improve the resilience of the system.

In this chapter, the attributes of resilience are explained and then, by using these attributes, measures of resilience are defined. Due to the fact that our domain in this research effort is enterprise and enterprise design, the resilience framework is tailored according to the notation and terminology in the field of enterprise design and evaluation.

3.1 Framing Resilience

The notion of resilience has many nuances, and that is because a large system such as a big enterprise faces many different disruptions, either natural or man-made, that impacts its performance. Consequently, in formulating the resilience of a system, sources of disruption must be decomposed and each one of them must be analyzed separately. For example, a building is susceptible to both earthquake and flood, but the resilience of the
building to each of these disruptions requires its own formulation and thus has a different solution. The difference in formulation originates from three sources: 1) behavior of a system in the face of disruption, 2) properties of a disruption, especially its magnitude, and 3) the state of a system and its environment when facing the disruption. First, the behavior of various systems is different when facing manifold disruptions. For example, a building may be resilient to earthquake up to magnitude 6.0 on the Richter scale, while another building can withstand a quake with maximum magnitude of 7.0 on the Richter scale. As these two examples show, two systems – here two buildings – behave dissimilarly to one another when facing a 7.0 Richter earthquake. Second, a system may be resilient to a certain kind of disruption only. For example, a building is resilient to tectonic earthquakes up to 6.0 Richter but it is not resilient to volcanic earthquakes. Third, the state of a system is a critical factor on how it can withstand a disruption. For example, the effects may be different if a disruption is occurring during peak operation time of a system, rather than a disruption that occurs during the period of low operational load time of the system. In the building example above, resilience of the building when its weight bearing columns are under intense pressure – for example due to presence of abnormal high weight load – will decrease comparing to the situation when the building structure is bearing normal weight.

3.2 Conceptual Resilience Framework

Resilience could be defined using a conceptual resilience framework [54] that has four attributes: 1) capacity, 2) tolerance, 3) flexibility, and 4) inter-element collaboration. This research effort uses this framework, thus, a brief description of each attribute is
provided in this section. In the next section we will quantify the framework for our analysis of resilience.

3.2.1 Capacity

Capacity of a system is defined as “the ability of the system to absorb or adapt to a disturbance without a total loss of performance or structure” [54]. This attribute denotes the amount of performance degradation a system can experience before it becomes ineffective. To better understand this concept, one should note that a system has multiple capabilities with their own performance specification. This performance level usually exceeds the required level defined in the requirements – as required in marginal engineering [56]. Therefore, there is some gap available for reducing the efficiency of a particular capability. Moreover, if a capability’s efficiency drops below the required level, still, that capability may be available but with a degraded effectiveness. The threshold at which a certain capability is considered lost or failed must be denoted – usually by a domain expert – to clarify the minimum performance level that is required by the capability. In the resilience community the rate at which a capability is degraded but still it is above the capability’s failure threshold is called graceful degradation [55]. Moreover, there is one other condition at which graceful degradation happens, but it applies only to Tolerance attribute of resilience, and thus is presented in the next section.

Capacity is further divided into three types [3], [16]: 1) buffering capacity, 2) reactive capacity, and 3) residual capacity. The buffering capacity is the marginal capacity available between the normal operational performance and the threshold level of performance. The reactive capacity is the marginal capability between the maximum
performance level, which includes any spare capability, and the threshold performance. The residual capacity is the marginal capability that is available at the end of the survival phase and the threshold performance level. Furthermore, capacity is a time varying attribute, and thus it should be evaluated over time to observe variations.

### 3.2.2 Tolerance

A tolerant system “exhibits graceful degradation near the boundary of its performance” [54]. The notion of graceful degradation constitutes two situations. First, a system operates at a reduced level of performance after a disruption, but it is still effective, which means it is above the performance and effectiveness threshold of the particular capability. Second, a system operates at a reduced level of performance that is below the threshold after the disruption, however, the rate of degradation is such that system’s users have enough reaction time. This second situation is especially important because in the real world designing a system that meets a smooth departure slope is much more promising than designing and implementing a system that does not violate a certain threshold point.

Tolerance is further decomposed into three sub-attributes [3], [16]: 1) Rate of Departure, 2) Fault Tolerance, and 3) Point of Failure. Rate of Departure (RoD) is the rate at which a system’s effectiveness – in relation to the threshold level – drops after a certain disruption. RoD is the key tolerance attribute. Fault Tolerance is the proportion of the atomic functionalities that can be unavailable without a loss capability to the total number of atomic capabilities. Point of failure is the impact of local failures on the overall performance level of a capability.
3.2.3 Flexibility

Flexibility is defined as “the ability of a system to construct itself in response to disruptions.” Flexibility is the ability of a system to reconfigure itself – i.e., change resources’ assignments – in the face of a disruption to retain its effectiveness with graceful degradation. Flexibility requires a change in the state of the system. This property is extensively used in this research to provide improved resilience for enterprises.

Flexibility is divided into three sub-attributes [3], [16]: 1) cohesion, 2) common use, and 3) proportion of use. Cohesion denotes the relatedness of the elements within a node that supports an individual capability. Furthermore, Cohesion defines the level at which a system can reorganize itself. For example, a system that is comprised of many modules with shared elements – i.e., highly cohesive – cannot reconfigure itself, because change in one module will require to make changes to other modules as well. Common use is the degree of utilization of each element by the simple functionalities. For example, in a system that all the elements (resources) are highly utilized, there will be a race condition between functionalities over the resources. Proportion of use is the fraction of the elements that is used by any single functionality – i.e., the fraction of elements needed for an atomic functionality.

3.2.4 Inter-element Collaboration

Inter-element collaboration is defined as the interaction and collaboration among the elements. It describes the empirical methods an organization (i.e., a system) can bring about to share the resources and perform the work in various manners. This property accounts for many of the emergent properties of a system or enterprise. In this research
effort, we extend this attribute of the Resilience Framework to influence the actual resilience of an enterprise. On one hand, this attribute cannot be quantified like other attributes [3], but on the other hand it can be considered as a design parameter in the system’s architecture to influence the actual resilience of a system. Inter-element collaboration is used as a design variable in the architecture descriptions used in this endeavor.

3.3 Quantifying the Resilience Framework

The first and the only quantification methodology of the resilience framework is provided by Pflanz [3]. This research effort leveraged many components of Pflanz’s work on resilience [3]. Furthermore, some parts of his research are either tailored or extended to fit our scope and analysis. In this section, the three attributes of the resilience framework are formulated, and lastly the inter-element attribute of the framework is presented as a design variable.

3.3.1 Measures of Capacity

Capacity has a time varying nature, and therefore this section starts with the temporal aspects of the capacity attribute of the resilience. The system starts at time $t_0$ (time origin) and then a disruption occurs at $t_d$. Subsequently, the system reaches its minimum performance at $t_{\min}$, then the system recovers to a new acceptable level at $t_{ret}$.
(see Figure 5). The avoidance phase starts at $t_0$ and ends at $t_d$, the survival phase starts at $t_d$ and continues to $t_{\text{min}}$, and the recovery phase is from $t_{\text{min}}$ to $t_{\text{ret}}$. Consequently, capacity has a measure at each phase; buffering capacity is measured in the avoidance phase and it is the ratio of the distance between the normal performance and the threshold performance to the normal performance (to the distance between the origin and the normal performance). Similar to the buffering capacity, the reactive capacity is the ratio of the distance between maximum capacity and the threshold capacity to the maximum capacity (see Figure 6). However, at the maximum capacity all the spare capacity of the system is available, but this is not the case for the buffering capacity – i.e., spare capacity may be off-line. The residual capacity is defined for the survival phase and in the ratio of the distance between the minimum performance and the threshold performance to either normal performance or maximum performance, whichever is higher. There exists another
reactive capacity formulation called reactive surrogate capacity that is applicable when the additional capacity does not increase the performance above the normal level of performance. Reactive surrogate capacity’s calculation is the same as for the reactive capacity.

### 3.3.2 Measures of Tolerance

The main concept of the tolerance attribute of a system is the ability of the system to withstand a bounded disruption. Tolerance means continued operation with graceful degradation; the system performs with reduced effectiveness but above the required threshold. A salient measure of tolerance is the rate of departure (RoD), i.e., the rate of change in the effectiveness of the system (for a particular capability) over a period of time. In addition to RoD, fault tolerance (FT) describes the ability of the system to perform its functions in the presence of faults. Moreover, the impact of localized failures to total failure is measured using the point of failure (PF) sub-attribute of tolerance. Both FT and PF are
calculated using graph theory concepts. RoD is calculated using capacity (buffering capacity), requirements locus (for calculating measure of effectiveness), and their temporal aspects. In the following subsections, these three measure and their formulations are presented.

Rate of Departure

As denoted previously, rate of departure is the slope of change in a system’s effectiveness over time. Cothier and Levis [14] and Bouthonnier and Levis [15] introduced the effectiveness of a system as the ratio of the intersection of its performance locus and requirements locus to the performance locus (see Equation 1).

\[
\frac{V(L_p \cap L_r)}{V(L_p)}
\]  

(1)

First, performance of a system is evaluated using the relevant measures of performance (MoPs). These MoPs are defined by domain experts and they quantify some salient characteristics of the system. For example, throughput is a commonly used MoP for information processing systems. In the next step, the geometric comparison between the measure of performance and the corresponding requirement of the operation is assessed using the two loci, \(L_p\) and \(L_r\). The geometric comparison method used in this work is the ratio of intersection area of the performance locus (\(L_p\)) and the requirements locus (\(L_r\)) to the total area of the performance locus (\(L_p\)). As Equation 1 shows, in a multidimensional space, measure of effectiveness (MoE) is the ratio of the volume of performance locus that is inside the requirements locus to the whole volume of performance locus. Therefore, MoE is a metric on how well the performance of a system matches the original requirements of
the system. In other words, how well the MoPs of the system fall inside the requirements’ boundaries.

As the result of a disruption, the MoE of a system will change. Therefore, MoE is considered a time varying metric, and subsequently Pflanz introduced the slope of the MoE [3] as shown in Equation 2. The rate of departure calculates the slope between the time that disruption starts ($t_d$) to the time that MoP reaches its minimum ($t_{min}$). To this end, a parameter locus must be defined to capture the performance of the systems under different values of the parameter(s). Selecting the parameter loci should be very meticulous because in many cases (for example in this effort) there is no close form formula to generate the performance locus from the parameter locus. To address this issue, parameters are selected to form the corner posts of the locus, and then by leveraging interpolation the surfaces are generated. An automated software tool, SEAT, has been developed to automate this process [57]. The MoE is calculated using this software package at both the disruption start time ($t_d$) and the minimum MoP time ($t_{min}$); thus RoD is calculated using the MoEs at $t_d$ and $t_{min}$.

\[
\text{RoD} = \frac{\left[ \frac{V(L_p \cap L_r)}{V(L_p)}, t_d \right] - \left[ \frac{V(L_p \cap L_r)}{V(L_p)}, t_{min} \right]}{t_{min} - t_d}
\]

Fault Tolerance

Fault tolerance is the second attribute of resilient systems. This attribute quantifies the number of faults that can occur before total loss of a capability. A capability is lost if its performance goes below a threshold that is defined based on the original requirements.
Fault tolerance as a measure is assessed by the ratio of the number of elements that can go off-line before the supported capability is lost to the total number of elements.

Two concepts must be clarified before the quantitative formulation of fault tolerance. First, capability is a notion which is used extensively in this effort. Valraud and Levis [13], [58] defined a capability as an orchestration of more than one simple functionalities that process input(s) and produces output(s). A simple functionality is further defined as a set of processes that work on an input message by following a particular flow of control. Further, in a distributed intelligence system (DIS) a simple functionality is described as a simple information flow path. In a Petri Net model of DIS, a simple information flow path is a directed path from a source through a set of processes to a sink. Therefore, in a Petri Net model, a capability has multiple simple information paths that have a common sink node.

Petri Net is a formal language that is constructed based on graph theory. As a result, the properties of information flow paths and capability are examined using graph properties. Pflanz [3] defined fault tolerance as the ratio of the number of information flow paths that can be disrupted (without the loss of the supported capability) to the total number of information flow paths for a particular capability. This definition implies that a capability is lost when the sink node of the graph (for that particular capability) is not reachable from the source; therefore tokens (messages) cannot reach the sink node from the source nodes (a single capability may have one or more source nodes).

In graph theory, a node that can be removed and the associated sink remains reachable from the source is named non-cut vertex. In a similar manner, a node that if
removed the path’s sink node will not be reachable from the source node is referred to as a cut vertex. In order to compute fault tolerance of a large-scale Petri Net model, non-cut vertices are used to calculate the numerator of the fault tolerance equation, Equation 3.

\[ FT = \frac{\sum_{i=1}^{r} x_i}{r} \]  

(3)

Where \( r \) is the total number of simple information paths, \( x_i \) is a binary variable that is one when the respective information path contains a non-cut vertex (at least), and it is zero otherwise. In addition, the denominator of Equation 3 is calculated using the place invariant method of finding simple paths [59].

An illustrative example is used to demonstrate the calculation of fault tolerance and other metrics explained in the rest of this chapter (see Figure 7). Organization “a” consists of two DMs, whom individually process input information without any interaction between them. Similarly, in the organizational design “b”, there are two DMs that process input data with interaction. Figure 8 shows the three simple paths of organizational design “b” (paths

![Figure 7](image1.png)

*Figure 7 Two hypothetical organizations: a) organization without interactions between DMs and b) illustrative organization with interaction between DMs*
1, 2, and 3); organization “a” has two simple paths (paths 2 and 3). Martinez and Silva introduced an algorithm to find the simple paths of a Petri Net structure [59]. This algorithm generates all the simple information paths of a Petri Net, thus by intersecting the simple paths, non-cut (and cut) vertices can be found. Their method is useful when dealing with large scale Petri Nets, which require a computer aided tool to calculate their unique simple paths.

The solution for the organization “b” in the illustrative example is as follows.

\[
FT = \frac{x}{r} = \frac{3}{3} = 1
\]

Organizational design “b” has three simple paths (\(r = 3\)) and each one of them could be disconnected without total loss of capability, thus, \(x = 3\). In a similar manner, organization design “a” has two simple paths each of which could be disconnected without the total loss of the capability, thus \(FT\) for organization “a” is also one.

![Three simple paths of the illustrative example (organization b)](image)

Figure 8 Three simple paths of the illustrative example (organization b)
**Point of Failure**

Point of failure is another attribute of tolerance that quantifies the number of individual local failures of elements that can happen without the loss of a capability. This metric assesses the strength of the link between the individual failures to the holistic capability collapse. For example, if a system’s structure is such that a single element failure causes the entire system to fail then we can conclude that the systems is not tolerant. Conversely, if a system can experience multiple element failures before total collapse then the system is considered tolerant (from the PF point of view). Equation 4 shows the formulation of PF [3]. Similar to fault tolerance, point of failure computations are based on graph theory and especially simple information paths terminology. Therefore, as Valraud and Levis noted [58], point of failure is the proportion of elements that are in a single simple path to the total number of elements of a particular capability.

\[ PF = \frac{\sum_{j=1}^{E} q_j}{E} = \frac{q}{E} \]  

where \( q \) is total number of elements with localized impact (i.e., each element is a member of only one single path), and \( E \) is the total number of elements.

In the illustrative example of Figure 7 and Figure 8, the elements with localized impacts are identified. Of the 19 elements, 11 elements are those that only contribute to one simple path (one locality). Consequently, as denoted in Equation 5, point of failure is 0.58.

\[ PF = \frac{q}{E} = \frac{11}{19} = 0.58 \]
As the point of failure measure increases, the number of elements in one and only one simple path will increase which means single faults will remain local. On the contrary, as point of failure measure decreases, the number of elements with local influence will decrease as well. Therefore, it is more likely that a single element failure can cause a broader system-wide stoppage.

3.3.3 Measures of Flexibility

Flexibility is the measure of the ability of a system to reconfigure itself. Normally, a system rearranges itself to adapt to its changing external environment. The ultimate goal for this adaptation is to keep a level of operational effectiveness or, in some cases, improving the effectiveness. A flexible system can deliver a capability through different structural forms. Therefore, flexibility requires some degree of redundancy (to provide more than one rigid form). Consequently, three concepts: redundancy, coupling, and adaptable components are the main pillars of flexibility.

Liles [60] defined coupling as the measure of interdependence among the nodes; in addition, he described cohesion as the relatedness of elements within a node. In Liles’s work, a system is comprised of nodes, and each node has one or more internal elements. Therefore, according to this terminology, coupling is a system wide metric, while cohesion is defined within each node. Nodes with high cohesion cannot be reconfigured easily; the high cost of modifying the form originates from the fact that in a highly cohesive node a minor change requires change to a large portion of the structure (i.e., nodes and arcs). However, the main idea of reconfiguration is that making specific changes in the structure are relatively cheap (i.e., cost beneficial). According to this rationale, a node with low
cohesiveness is cost-beneficial to reconfigure and vice versa. Liles defined the Degree of Reuse as the measure of the ability of a system to execute concurrent threads, each of which is a different capability. Pflanz renamed this metric to common use (CU), but the underlying concept remains the same (i.e., multi-threading). A system with low level of common use among its elements is able to perform multiple capabilities concurrently. The common use has another significance as it denotes the level of utilization of each organizational element. Therefore, low level of common use also suggests low utilization factor. Pflanz defined the proportion of use (PoU) as the normalized metric of common use. The proportion of use denoted the ratio of the number of elements assigned to a single capability to the total number elements. For example, on average a single capability uses 10% of the total elements. In the following subsections, these three concepts are covered in greater detail.

**Cohesion**

Liles modeled a System of Systems (SoS) through the introduction of a SoS instance (SoSI) [60]. Each instance represent a collection of nodes that deliver a certain capability. Further, each node is comprised of elements. Elements within each node interact using information paths (or arcs in the Petri Net). Subsequently, by using the notion of information flow paths and place invariants, Liles defined the cohesion of a node as the ratio of the number of simple paths in the node to the product of the number of inputs and outputs of that node, Equation 6.

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

(6)
where \( k \) is the identifier of the specific instance, \( i \) is the index of the node, \( z_{ki} \) is the number of paths in node \( n_{ki} \). In addition, \( x_{ki} \) is defined as \( x_{ki} = I_{ki} \times Q_{ki} \), where \( I_{ki} \) is the number of the node’s inputs, and \( Q_{ki} \) is the number of the node’s outputs.

Again, using the illustrative example of the enterprises “a” and “b” in Figure 7, cohesion is calculated for DM1 in organization “b”

\[
Coh(n_{DM1}) = \frac{z_{DM1}}{x_{DM1}} = \frac{2}{1 \times 2} = 1
\]

Similarly, for DM2 the cohesion is as below

\[
Coh(n_{DM2}) = \frac{z_{DM2}}{x_{DM2}} = \frac{2}{2 \times 1} = 1
\]

After calculating the cohesion of all nodes, overarching cohesion is derived as

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

where \( m \) is the number nodes in the instance.

In the notional example, overall cohesion is calculated as

\[
Coh(f_k) = \frac{1 + 1}{2} = 1
\]

**Common Use**

Common use quantifies the degree to which a System of Systems instance will support multiple capabilities in a concurrent manner or, in other words, multi-threading degree. Liles [60] observed that common use decreases as the element utilization increases. This situation causes a race condition in which only few elements can enter critical section – gain access to the required resources – and perform their functions. Equation 8 denotes
the Liles formulation of the common use (Degree of Reuse):

\[ CU = \frac{\sum_{j=1}^{E} A_j}{E} = \frac{A}{E} \]  \hspace{1cm} (8)

where \( A_j \) is the number of information flow paths that has element \( j \), \( E \) is the total number of elements. Going back to our illustrative example, common use for organization “a” is \( CU = \frac{A}{E} = \frac{18}{19} = 1 \) and for organization “b” is \( CU = \frac{A}{E} = \frac{27}{19} = 1.42 \).

The interpretation of this measure is that, on average, an element is a member in 1.42 simple information paths (in organization “b”). As the common use increases, the resource conflicts (race condition) will increase as well. That is because a single element can perform multiple functions. A normalized variation of common use is the proportion of use that is presented in the next subsection.

**Proportion of Use**

Proportion of use measures the ratio of the number of elements that are participating in a single functionality to the total number of elements [3]. For example, in an enterprise 5% of elements are participating in a functionality on average while in another enterprise 90% of elements. As mentioned in the common use subsection, utilization and resilience (flexibility attribute) are related to each other. Systems with lower proportion of use are less utilized, and therefore, they can be reconfigured more easily. As a result, they become more resilient. Lower utilization means that elements participate in fewer number of simple functionalities; thus the reconfiguration is more cost-beneficial and vice versa. The proportion of use is denoted in Equation 9.
where \( r \) is the total number of information flow paths, \( B_i \) is the number elements in path \( l_i \), and \( E \) is the total number of elements.

Again, using the notional example, PoU of organization “a” is \( PoU = \frac{18}{18+2} = 0.5 \) and for organization “b” PoU is \( PoU = \frac{27}{19+3} = 0.47 \). These results mean that each simple path contains 0.47 of the elements (in organization “b”). In other words disrupting 47% of the elements may impact only one simple path.

These attributes are used in the transition strategy of the proposed methodology. They will help identify a more resilient structural form after detecting an anomaly. An indexing method (of enterprises resilience) is required in an adaptive organization to find the next resilience locus and its corresponding structural form. This method is discussed in-depth in chapter five.

### 3.3.4 Inter-element Collaboration Architectural Description

As previously mentioned, inter-element collaboration is not a quantified attribute of resilience. However, in this research effort, a system’s architectural descriptions are used to first define the inter-element collaborations among the elements of the system, and then influence the resilience of the system by making changes in the inter-element collaborations’ descriptions. The nature of these modifications is guided by the rules and constraints of enterprise design defined by Remy et al. [5]. These rules guarantee the live execution of the organization’s DES model and they will be elaborated in chapter six.
The critical role of the architecture becomes apparent especially in this part of the design. A system’s architecture is where we can describe inter-element collaboration, and subsequently modify it to influence the overall resilience. In other words, the design decision about the form of the inter-element collaborations has a direct impact on the enterprise’s resilience, i.e., on the four attributes of the resilience framework. Levis introduced the discrete event model of distributed intelligence systems (DIS) [41], which are adopted in this work for modeling the enterprises in general. The original contribution of the Levis’s model is its support of interactions among the enterprise’s elements (or, in his terminology, decision-makers); thus this model is an appropriate paradigm to describe inter-element collaborations. Levis and Perdu extended the idea of modeling DIS by discrete event systems (e.g., Petri Nets) to a design and analysis tool, CAESAR [61]. Consequently, Wangenhals et al. added more features (e.g., cultural factors) to the original CAESAR software package and released CAESAR III [62]. In addition, CAESAR III provides the lattice analysis that is leveraged to find appropriate structural forms in the presence of a disruption. The mathematical foundations of both the organization design used in CAESAR III and corresponding lattice algorithm are presented in chapter four.

3.4 Resilience to a Particular Cyber Perturbation

Pflanz stated that resilience has three aspects: 1) type and magnitude of a disruption, 2) system’s state when facing the disruption, and 3) type and structure of the system, or in this work, enterprise. Therefore, we must precisely define the system, the disruptions of interest, and conditions under which the disruption happens. In this section, the detailed description of these parameters is explained.
The system of interest in this research effort is a distributed intelligence system (DIS) that is comprised of intelligent nodes (also called decision-makers). Intelligent nodes interact with each other to perform tasks and accomplish functions that will contribute to overall goal of the system. In this work, a distributed intelligence system is referred to as an enterprise. An enterprise performs businesses (i.e., engineered processes that produce the desired outcomes) using its resources (i.e., intelligent nodes). These nodes are described in the physical architecture view and they communicate (interact) with each other in order to conduct the enterprise’s business.

The communications are usually facilitated via a queuing communication infrastructure (e.g., a cloud). These types of infrastructure are prone to cyber disruptions, which in this work we refer to as cyber perturbations. This work only considers those perturbations that are generated intentionally by actors with malicious intent. Therefore, we are not working with natural faults and outages that may happen to an infrastructure.

The test setup that is considered for this research is an enterprise performing its business and then faces a cyber disruption. The goal of this test is to find a structural form that makes the organization more resilient under that specific cyber perturbation while performing its business. The test setup only considers a particular business process and tests the resilience of different structural forms of the enterprise to compute the resilience metrics of each configuration. The ultimate goal is to construct an adaptive enterprise that when detects an anomaly (presence of a cyber disruption) it adopts a new form which is more resilient. Therefore, the enterprise can continue its operation in a perturbed cyber environment. While the definition of contested cyber environment is broad and elaborated,
a cyber environment with an active cyber perturbation is considered as a contested cyber environment in this study. In addition, the cyber perturbation type that is investigated in this research effort is the availability type. The more detailed explanation of different cyber perturbations and enterprise structures is presented in chapter five.
CHAPTER 4: ENTERPRISE DESIGN

The novel methodology that is presented in this research is deployed in the discrete event model of the enterprise. In the current effort, an enterprise, which is under investigation, is designed and modeled using the theory of intelligent nodes that was introduced by Levis [41]. The process of generating architectures of intelligent systems (using intelligent nodes) is then provided by Zaidi and Levis [44]. Their method is based on lattice theory that was initially proposed by Remy [43], and later extended by Olmez [47]. In addition, the enterprises that are produced in the final phase of the methodology are adaptive. The foundations of the theory of adaptive organizations is first put forth by Perdu [4]. The current work leverages a modified version of Perdu’s methodology. The only changes in the original theory of Perdu are: a) the anomaly detection mechanism is changed to excessive sojourn time of tokens [46], and b) in the transition strategy a novel performance metric, referred to as capacity score, is introduced to select between the multiple available forms after detecting an anomaly. This chapter provides an in-depth description of all these theories and techniques. In the latter parts of this chapter, the synthesis of all the components of the methodology is presented.

4.1 Organizational Design

The design of an enterprise is created according to decision making theory, Petri Net theory, and lattice theory. Therefore, before elaborating on the research methodology,
these theories are discussed. Petri Net theory is a pivotal method in this work but, because it is a well-known paradigm, it is provided in the appendix.

### 4.1.1 Five-stage model of Interacting Decision-makers

The five-stage model of interacting decision-maker is a sequential model of decision process introduced by Levis [41]. This model is developed for simulating the interacting intelligent nodes and it is comprised of five sequential stages, four of which are interaction stages. Furthermore, an enterprise (which is a distributed intelligent system) is composed of a number of interacting intelligent nodes, which are referred to as decision-makers in decision theory. Therefore, a distributed intelligence enterprise is modeled using the five-stage model of decision-maker. In addition, there exists a discrete event model (Petri Net model) of this model of decision-maker that enables the simulation and analysis of distributed intelligent enterprises.

In the five-stage model of interacting decision-maker, Figure 9, an input signal, $x$, is received from the external environment. The Situation Assessment (SA) stage processes the input signal ($x$) to produce an assessment of the prevailing situation, $z$, which can be sent to other decision-makers. At the Information Fusion (IF) stage, situation assessment...
of other decision-maker(s), \( z' \), is fused with the assessment of the current decision-maker, \( z \), and produces an aggregated situation assessment, \( z'' \).

The fused assessment, \( z'' \), is then processed at the Task Processing (TP) stage to produce the list of applicable responses, \( v \). Subsequently, at the Command Interpretation (CI) stage, the DM receives guidance from the supervisor, \( v' \), and produces the input, \( w \), to the Response Selection (RS) stage which then produces the final response, \( y \).

The five-stage model of interacting decision-makers is an extended version of the two-stage decision making model introduced by March and Simon [50]. The original contribution of the five-stage model of decision-maker is its ability to replicate interactions, and, in fact, this model is the only discrete event model of interacting decision-makers. Therefore, Information Fusion and Command Interpretation stages are the salient stages of the model. This feature makes the five-stage model suitable to model hierarchical (vertical) enterprises that have members with different ranking levels, e.g., subordinates and superiors. Moreover, it must be clarified that vertical enterprises cannot be modeled easily as a multi-agent system (MAS). The reason behind this difficulty is the characteristics of such systems, which are autonomy, decentralization, and limited knowledge of the world. The main conflicting characteristic is the autonomy condition that is required by multi-agent systems; however, vertically structured enterprises may be comprised of members with different rankings, i.e., they are not autonomous. Hierarchical enterprises can take multiple forms.

The interactions between the intelligent nodes (decision-makers) are of six types, when the interactions between the external environment and an intelligent node are
Viable interactions between an intelligent node $i$ and an intelligent node $j$ are as follows (see Figure 10):

- Input $e_i$ from the external environment to the node $i$,
- Output $s_j$ from the node $i$ to external environment,
- Interaction variable $F_{ij}$ denotes sharing situation assessment between node $i$ and node $j$. This interaction is defined between the SA stage of the node $i$ and the IF stage of the node $j$,
- Variable $G_{ij}$ depicts sequential processing from node $i$ to node $j$. This variable is defined between the RS stage of node $i$ and the SA stage of node $j$,
- $H_{ij}$ models the sharing the response selection between node $i$ and node $j$. This interaction links the RS stage of node $i$ to the IF stage of node $j$,
- and $C_{ij}$ denotes the flow of directives from node $i$ to node $j$ and it associates the RS stage of DM$_i$ to the CI stage of DM$_j$.

Figure 10 Interaction between intelligent nodes
One important distinction between the types of interactions is that some of them represent flow of control and others denote information sharing between the intelligent nodes. As a general rule, links to SA (mainly from RS) are defining control flow and connections that end at IF are information sharing variables. Therefore, this model can capture both the flow of control through, $G$ and $C$ and information sharing through $F$ and $H$.

In a hierarchical enterprise, decision-makers may have a sub set of the five stages of decision making; there are four possible structures that a hierarchical enterprise can have [44]: 1) only SA with $y = z$, 2) SA, IF, TP, CI and RS, 3) IF, TP, CI, and RS with $x = z'$, and 4) CI and RS with $x = v'$. Consequently, there are different interactions possible in each structure.

Remy and Levis introduced a set of rules for designing enterprises using distributed intelligence systems terminology [5], [43]. These rules guarantee a deadlock free design and implementation (using Petri Net theory). The four main structural constraints are as follows:

- Rule 1 ($R_1$): a directed path must exist from the source to every node in the enterprise – that is every node is reachable from the source node. The equivalent condition applies to the sink node. Therefore, a sink node is reachable from every node in the enterprise.
- Rule 2 ($R_2$): the structure of an enterprise is acyclic; thus it has no loops.
- Rule 3 ($R_3$): there can be one and only one link of type $G, H, C$ between each pair of intelligent nodes.
• Rule 4 (R₄): SA must have one and only one input.

The set of these four rules is \( R_s = \{R_1, R_2, R_3, R_4\} \). An enterprise design that adheres to set \( R_s \) is a safe net and can be represented by the six-tuple, \( \Sigma \), which is elaborated in the following section.

### 4.1.2 Mathematical Representation of an Enterprise

A distributed intelligence enterprise with \( n \) intelligent nodes can be represented using the six-tuple, \( \Sigma \). A network that is represented by \( \Sigma \) is a Well Defined Net (WDN).

A WDN of dimension \( n \) is represented by the six-tuple

\[
\Sigma = \{e, s, F, G, H, C\}
\]

(10)

where \( e \) and \( s \) are \( n \times 1 \) vectors denoting the interactions between \( n \) intelligent nodes and the external environment

\[
e = [e_i], \quad s = [s_i], \quad i = 1, 2, \ldots, n
\]

(11)

In a similar manner, \( F, G, H, \) and \( C \) are four \( n \times n \) matrices representing the interactions between the intelligent nodes of the enterprise structure

\[
F = [F_{ij}], \quad G = [G_{ij}], \quad H = [H_{ij}]
\]

\[
C = [C_{ij}], \quad i, j = 1, 2, \ldots, n
\]

(12)

Each element of these six matrices is a binary number. A “1” signals the presence of the particular link; while a “0” denotes the absence of a certain link. In addition, all the diagonal elements of the matrices \( F, G, H, \) and \( C \) are all zeros; thus nodes do not interact with themselves.

\[
F_{ii} = G_{ii} = H_{ii} = C_{ii} = 0, \quad i = 1, 2, \ldots, n
\]

(13)
The designer of an enterprise translates the flow of control (as defined in by an activity model) using the interaction variables $G$ and $C$. Subsequently, matrices $e, s, F$ and $H$ are partially completed using potential information exchanges (those that are in addition to flow of control). The notion of user-defined is manifested by assigning the binary variable, $x$, to the matrix elements. For example, a user-defined constraints for an enterprise is given as

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

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

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

(14)

The $x$’s in the matrices represent the optional links. The optional links provide a degree of freedom in the design process that yields multiple candidate solutions that satisfy all the structural constraints ($R_s$). At this point, some definitions pertinent to the candidate solutions are introduced.

Definition 1: Let $\Pi = (e, s, F, G, H, C)$ and $\Pi' = (e', s', F', G', H', C')$ be two candidate structures of an enterprise with dimension $n$. $\Pi$ is referred to as a subnet of $\Pi'$, if and only if:

$$e \leq e', s \leq s', F \leq F', G \leq G', H \leq H', C \leq C'$$

(15)

i.e., all interactions in structure $\Pi$ are also present in structure $\Pi'$. $\Pi'$ can have additional interactions. Furthermore, the set of all candidate structures with subnet relationship is a partially ordered set. One property of the partially ordered set of well-defined structures of an enterprise is that if all the $x$’s are set to zero then the kernel subnet, $\omega^n$, will be obtained.
The kernel structure contains interactions that are present in all structural candidates. The reciprocal structure of the kernel is the structure in which all the \( x \)'s are set to one. This structure is denoted by \( \Omega^n \). There are three complimentary theorems for combining structures.

**Theorem 1:** Let \( \Pi^1 = (e^1, s^1, F^1, G^1, H^1, C^1) \) and \( \Pi^2 = (e^2, s^2, F^2, G^2, H^2, C^2) \) be two candidate structures of an enterprise with dimension \( n \). The join of \( \Pi^1 \) and \( \Pi^2 \) is \( \Pi = \Pi^1 \cup \Pi^2 \) with structure

\[
e = e^1 \cup e^2, s = s^1 \cup s^2, F = F^1 \cup F^2,
\]

\[
G = G^1 \cup G^2, H = H^1 \cup H^2, C = C^1 \cup C^2
\]

(16)

This states that \( \Pi \) contains all interactions in \( \Pi^1 \) and \( \Pi^2 \).

**Theorem 2:** Let \( \Pi^1 = (e^1, s^1, F^1, G^1, H^1, C^1) \) and \( \Pi^2 = (e^2, s^2, F^2, G^2, H^2, C^2) \) be two candidate structures of an enterprise with dimension \( n \). The intersection of \( \Pi^1 \) and \( \Pi^2 \), denoted \( \Pi = \Pi^1 \cap \Pi^2 \) with structure

\[
e = e^1 \cap e^2, s = s^1 \cap s^2, F = F^1 \cap F^2,
\]

\[
G = G^1 \cap G^2, H = H^1 \cap H^2, C = C^1 \cap C^2
\]

(17)

\( \Pi \) contains only the interactions that are both in \( \Pi^1 \) and \( \Pi^2 \).

In addition, these two theorems imply that all combinations of well-defined networks are also well-defined networks.

**Theorem 3:** The set of all possible well-defined structures of an enterprise of dimension \( n \) is a lattice.

Lattice theory plays a key role in selecting the structure of an enterprise; therefore it is discussed in the following subsection.
4.1.3 Lattice Theory

Lattice theory governs the notions that guide the generation of feasible enterprise structures. These notions are user-defined and structural constraints, convexity, universal and kernel nets, and minimally and maximally connected organizations. The user-defined constraints and structural constraints were discussed in section 4.1.1; therefore, only the concepts of feasibility of a structure, universal and kernel nets, and minimally and maximally connected enterprises are explained in the following subsection. These notions follow the work of Remy and Levis [43].

Each six-tuple, $\Pi$, is a Well Defined Net of the enterprise with $n$ intelligent nodes. A Well Defined Net is referred to as a feasible organization (FO) if and only if it satisfies both structural and user-defined constraints. A Well Defined Net that only satisfies the user-defined constraints is called an admissible organization form (AOF). Therefore, an AOF that complies with structural constraints is a FO. The set of all FOs are denoted as $\Phi(R)$.

Minimally and Maximally Connected Organizations (MINOs and MAXOs)

The partially ordered set of all the Well Defined Nets of an enterprise with size $n$ is denoted by $\Psi^n$. It is trivial that the set of all feasible organizations, $\Phi(R)$, is a subset of $\Psi^n$, and that is due to the fact that not all the WDNs satisfy user-defined and structural constraints. Since $\Psi^n$ is a partially ordered set; then $\Phi(R)$ is also a partially ordered set. Consequently, $\Phi(R)$ has one or more lower bound elements, which are called Minimally Connected Organizations, MINO, and similarly, some upper bound elements named
Maximally Connected Organizations, MAXO. In the lattice notation MINO and MAXO are denoted as \( \Phi_{\text{min}}(R) \) and \( \Phi_{\text{max}}(R) \) respectively.

A minimally connected organization (MINO) is a Well Defined Net such that it is not possible to remove a link from it and remain in the set \( \Phi(R) \). In other words, if any of the links in a minimally connected organization is removed then one of the user-defined or structural constraints is violated. Similarly in a maximally connected organization (MAXO), no links could be added without violating at least one user-defined or structural constraint. Equation 18 displays the definitions of MINO and MAXO. In this equation, MINO is denoted as \( \Pi_{\text{min}} \) and MAXO is printed as \( \Pi_{\text{max}} \).

\[
\{ \Pi \in \Psi^n | \exists (\Pi_{\text{min}}, \Pi_{\text{max}}) \in \Phi_{\text{min}}(R) \times \Phi_{\text{max}}(R), \Pi_{\text{min}} \leq \Pi \leq \Pi_{\text{max}} \} = \Phi(R) \tag{18}
\]

One salient property of the set \( \Phi(R) \) is convexity. A set is convex if and only if, for any two sets, \( \Pi^1 \) and \( \Pi^2 \) such that \( \Pi^1, \Pi^2 \in \Phi(R) \), then any \( \Pi \) such that \( \Pi^1 \leq \Pi \leq \Pi^2 \) is also an element of \( \Phi(R) \) (\( \Pi \in \Phi(R) \)). In other words, if an enterprise structure, \( \Pi \), is a subnet of an element in \( \Phi(R) \), and also another element in the set \( \Phi(R) \) is a subnet of \( \Pi \), it can be concluded that \( \Pi \) is also in the set \( \Phi(R) \). As a result, if a MINO and a MAXO of a particular lattice are in the set \( \Phi(R) \), then all of the structures in that lattice are also in the set \( \Phi(R) \). Consequently, if we prove that a MINO and a MAXO of a particular lattice are well-defined and feasible structures then all the elements of that lattice are well-defined and feasible. The condition of convexity is denoted in Equation 19.

\[
\forall (\Pi^1, \Pi^2) \in \Phi(R) | \Pi^1 \leq \Pi \leq \Pi^2 \rightarrow \Pi \in \Phi(R) \tag{19}
\]

**Universal Net and Kernel Net**
The notions of universal net and kernel net denote the most complex element as well as the simplest element of $\Psi^m$. The universal net is the element of the set $\Psi^m$ in which all the binary variables $x$ of the six-tuple, $\Pi = \{e, s, F, G, H, C\}$, are set to one. This assignment makes the universal net the most intricate structure of the set $\Psi^m$; thus all the elements of $\Psi^m$ are subnets of the universal net. Inversely, kernel net is the element of the set $\Psi^m$ in which all the undefined structural variables $x$ are set to zero. Therefore, kernel net is the smallest element of the set $\Psi^m$, and it is the subnet of all the elements of the set $\Psi^m$. Please refer to the Equation 20 for the formulation of both universal and kernel nets.

$$\Phi(R) = \{\Pi \in \Psi^m | \omega(R) \leq \Pi \leq \Omega(R)\} \quad (20)$$

Before moving forward to the lattice algorithm, the concept of simple path should be presented. A simple path is a minimal $S$ invariant of structure $\Pi$ that connects a source place to a sink place (i.e., external places). Or, in brief, a simple path of structure $\Pi$ is a directed elementary path between external places of the structure.

*Lattice algorithm (Remy and Levis [43])*

The Lattice algorithm works on the simple paths to generate MINOs and MAXOs. The generated MINOs and MAXOs must comply with the user-defined and structural constraints. A MINO is generated by adding simple paths to the kernel net of the corresponding enterprise structure until all the user-defined and structural constraints are satisfied. Similarly, a MAXO is generated by removing one or more simple paths from the universal net until all the user-defined and structural constraints are satisfied. In other
words, kernel and universal nets are tailored to the point that become the elements of the set $\Phi(R)$.

The set $\Phi(R)$ is referred to as solution space, and it contains the lattice’s end-to-end structures (i.e., MINOs and MAXOs) as well as the middle structures that satisfy the convexity definition.

4.1.4 Task allocation

The problem of task allocation is investigated by Levis et al. [63], [64]. Their study addressed the decomposition of a mission or functionality into atomic parts that could be carried out by a single intelligent node (or, according to Levis et al., by a single agent). Furthermore, the notion of roles is defined. Roles are pivotal parts of the enterprise design because adaptive enterprises rely on the notion of the role rather than the definition of the task. In summary, according to the activity model (business model) tasks and their flow of control are static notions. Therefore, the dynamism of an adaptive enterprise depends on the roles (as defined in the Role Definition Matrix). In this section task allocation and the notion of role are elaborated.

In an enterprise, human agents and machines are modeled as intelligent nodes. Each node performs several functions either in sequence or concurrently. Therefore, the allocation problem is not only the assignment of decomposed functions to the resources. The real issue is allocating several decomposed functions to resources while the workload of each resource does not surpass its threshold value. This threshold includes both the workload of carrying out the functions and the load generated by the coordinating activities required to perform the functions. These coordination activities are between intelligent
nodes (i.e., the human decision-makers), between computational assets (e.g., computer systems), and between intelligent nodes and the computational assets.

The first step in solving the allocation problem is the functional decomposition. The functional decomposition is accomplished in a hierarchical manner from the mission into functions and functions are further parsed into tasks (see Figure 11). This process can be repeated in a nested manner; therefore, a function in one step is considered a mission for the next step and so forth. Consequently, the decomposition problem is an optimization problem whose objective is to parse the mission into tasks. The notion of the role has been introduced by Levis et al. [63] as a basic concept for modeling the task allocation in the distributed intelligence systems (DIS). This notion modifies the classic association between the intelligent nodes (DMs) and the lowest level tasks in such a way that the concept of the role comes in the middle of the nodes and their tasks (see Figure 11). A role

![Figure 11 Concept map of role](image-url)
facilitates the binding of a task to a node. Therefore, a node has one or more roles, and each role is responsible for executing a certain task. The main logic behind the definition of the role is that it enables the dynamism of an adaptive organization. This is due to the fact that both the flow of control in the activity model (for the holistic functionality) and the intelligent nodes (resources) have static descriptions. Therefore, during a single execution of the business scenario both activity model and physical model will remain unchanged; however, adaptive enterprises (DIS’s) requires dynamic behavior. The concept of role provides this adjustment because roles can be flexible during runtime. The role assignment is mainly expressed in the role definition matrix (RDM) [45].

4.2 Adaptive Enterprise

After providing the formal foundations of enterprise design and analysis, this section presents the overarching enterprise model that is used as the analytical testbed of this research effort. The high-level representation of this novel design is depicted in Figure 12. The figure displays two explicit partitions: 1) anomaly detection and 2) execution/transition. The execution part of the design consists of two modules: 1) execution and 2) transition. Furthermore, components such as “monitor” and “enabler” provide the required connections between the enterprise’s executable model (Petri Net model) and the two partitions. The two partitions assess the state of the enterprise (anomaly detection) and activate appropriate responses (either execution or transition). The anomaly detection is a permanently recurring process that works on every new entry to the event log database of the enterprise. If the new entry suggests normal status then the execution process continues and, if an abnormal state is detected, then the transition process is enabled. The anomaly
detection in this research effort is implemented by an outlier detection method called excessive sojourn time. Once an anomaly is detected, the transition process is initiated through a strategy which will estimate the next structural form (based on the repository of the learned resilience). Furthermore, under normal conditions execution of the business model will continue. The strategy that directs the transition ensures moving toward a more

Figure 12 Proposed adaptive enterprise system
resilient structural form. There exists a problematic dichotomy in selecting the more resilient structural form that will be addressed in the Execution/Transition section. In short, the issue is that not all of the resilience indexes change in the same manner, i.e., improving some of them may degrade others. In order to solve this issue, an optimization solution is introduced in the Execution/Transition section.

Thus, the anomaly detection mechanism discerns any abnormal communicational behavior. However, the actual source of the abnormal behavior as well as its process are not transparent to our method. The reason for this lack of clarity is that from an enterprise point of view the networking process that is executing according to certain protocols is not visible. As a result of this shortfall, only the effects of cyber exploits (i.e., availability and integrity) are considered in our work, and these effects are referred to as cyber perturbation throughout this document. After detecting abnormal communication by the anomaly detection components, a transition command is generated. The transition is from a structural form to another form with enhanced resilience (better effectiveness).

In order to ensure transition to a higher resilience locus, a rigorous strategy is required to select the appropriate structural form. The strategy introduced in the Execution/Transition section is devised such that it guarantees that during the transition none of the resilience indexes will degrade (i.e., indexes will either improve or remain constant). This strategy is neither intuitive nor discernible, and this is due to the complex nature of resilience as explained by the measures defined by the resilience framework. Accordingly, the attributes of resilience do not change in a congruent manner; therefore, improving one
set of attributes may downgrade the other set. In the next two sections, the two components of the methodology are discussed in detail.

4.2.1 Anomaly Detection

Theorists like March and Simon [50] introduced the notion of coordination in organizations but they did not provide any mathematical models for it. Grevet and Levis [19] introduced the mathematical definitions of coordination in decision-making organizations. The two coordination measures are *degree of information consistency* and *degree of synchronization*.

The notion of coordination only applies to the synchronization points in the five-stage model of decision-making, i.e., at the IF or CI stage. There are order relations that guard the firing of transitions in the presence of various tokens. The presentation follows the developments by Grevet [46].

Ψ₁ is a binary relation defined by:

\((x, y, z) \Psi_1 (x', y', z') \iff (x = x') \text{ and } (y \leq y')\)

Ψ₂ is a binary relation defined by:

\((x, y, z) \Psi_2 (x', y', z') \iff (x = x') \text{ and } (z = z')\)

Ψ₃ is a binary relation defined by:

\((x, y, z) \Psi_3 (x', y', z') \iff ((x, y, z) \Psi_1 (x', y', z')) \text{ and } ((x, y, z) \Psi_2 (x', y', z'))\)

Each token in the CPN model has a triplet structure \((T_n, T_d, C)\) where \(T_n\) is the time at which this token entered the organization, \(T_d\) is the time at which this input reached the current stage, and the parameter \(C\) is the content (color) of the input.

The interacting transitions IF (or CI) are synchronized if and only if:
∀i≠k (i, k) ∈ {1, 2, ..., r} × {1, 2, ..., r}, (T_n, T_d, C_i) Ψ_1 (T_n^k, T_d^k, C_k)

where i and k are two interacting DMs. This definition means that the firing of IF (or CI) is synchronized when all the required tokens – in this case triplets i and k – are available. Otherwise, if there is a delay between the two tokens, arriving at the post places of the IF, then it is not synchronized.

The information of the IF (or CI) stage is consistent if and only if:

∀i≠k (i, k) ∈ {1, 2, ..., r} × {1, 2, ..., r}, (T_n, T_d, C_i) Ψ_2 (T_n^k, T_d^k, C_k)

Thus, the information content of all interacting DMs are congruent. As a result, an interaction stage like IF (or CI) is coordinated if and only if:

∀i≠k (i, k) ∈ {1, 2, ..., r} × {1, 2, ..., r}, (T_n, T_d, C_i) Ψ_3 (T_n^k, T_d^k, C_k)

Thus, the IF (or CI) stage is coordinated if it is both synchronized and consistent.

Furthermore, the execution of a task is coordinated if, and only if, it is coordinated for all the interactions that occur during the execution. Moreover, the execution of a Petri Net is coordinated if, and only if, it is coordinated for all the tasks.

The Sojourn time, T_s, of the token m_h, representing input x_i, in the place p_h, measures the amount of time that has been elapsed since m_h entered p_h (T_d^h) and transition t_{int} fires, T_c:

T_s^h (x_i, t_{int}) = T_c - T_d^h

This quantity is zero when t_{int} is fully synchronized. Otherwise, a positive Sojourn time suggests that there is a lag in receiving different pieces of information. Further, the following quantity will be defined:

S_{L}^{hj} (x_i, t_{int}) = T_s^h (x_i, t_{int}) - T_s^j (x_i, t_{int})
This quantity measures the difference between the Sojourn times of the tokens in preplaces \( h \) and \( j \), of \( t_{int} \).

The function \( F(x) \) is defined as:

\[
\forall x \in Q, (x \geq 0) \implies (F(x) = x) \\
(x < 0) \implies (F(x) = 0)
\]

Then the total lag for the transition \( t_{int} \) in processing input \( x_i \), \( S(x_i, t_{int}) \), is as follows:

\[
S(x_i, t_{int}) = \max_{h \in INT(t_{int})} (F[S_{kh}^L(x_i, t_{int})])
\]

The total lag is the maximum of all the lags and therefore the bottleneck of the decision maker’s processing time is the longest wait. The measure of synchronization of decision maker \( DM^k \) is defined as:

\[
S_k = \sum_{x_i} \text{prob}(x_i) \sum_{t_{int}} S(x_i, t_{int}) \quad (21)
\]

which is the expected value of sum of the lags for all the interaction transitions, \( t_{int} \). Finally, the synchronization for the organization, \( S_T \), is defined as:

\[
S_T = \sum_k S_k \quad (22)
\]

The notion of sojourn time in the synchronization measure is used to detect anomalies in this research effort. The sojourn time of each token is calculated and if it is above a certain threshold (e.g., 15 min of the FAA’s EA), then, an anomaly alert is issued and the transition starts.
4.2.2 Execution

The execution mechanism, which governs the executable model of an enterprise, is that of Petri Net’s transition firing rules [20]. The generic firing rule requires the availability of binding markings in all of the pre-places of that particular transition/ event. Consequently, the execution of the transition with the binding markings produces markings in the post-places of that transition. This mechanism is illustrated for the Petri Net in Figure 13 by

\[ t_1: m_i \& m_j \rightarrow m_k \]  

(23)

where \( m_i \) and \( m_j \) are the binding markings of the two pre-places of the transition, \( t_1 \). The firing of the transition produces the marking, \( m_k \), and puts it the post-place of the transition. This generic rules applies to all the transitions and places of the Petri Net. This generic execution rule could be extended to transitions with more than two input places and more than one output places.

![Figure 13 Information Fusion stage](image)
CHAPTER 5: METHODOLOGY FOR DESIGNING RESILIENT ENTERPRISES

The problem of assessing the resilience of an enterprise when facing a certain cyber disruption involves various domains and formalisms of which the resilience framework and enterprise design were provided in chapter 3 and chapter 4, respectively. Due to the intricate nature of this problem, a multi-formalism design approach [24], [65] is used to achieve maximum adherence to the respective real world phenomena, which involves various processes and protocols. Since the multi-formalism technique is used, the corresponding modeling method would be multi-modeling [6], [18]. In this research effort, design is a federation of multiple domains and formalisms such as Enterprise Architecture, Resilience Framework, and queuing communication networks. Subsequently, modeling paradigms such as Colored Petri Nets (CPN), object-oriented modular discrete event network simulation framework (i.e., OMNeT++), High Level Architecture (HLA), and Run-time Infrastructure (RTI) are used to create the federation of multiple federates each of which represents a domain.

To illustrate the state of the art design and modeling approaches that are used in this research effort, a concept map of the formalisms and their respective modeling modules is presented in Figure 14. As displayed in this figure, an enterprise has a design that is represented by its enterprise architecture; in addition, it has a configuration that is defined by a mathematical notation such as the one used by the lattice algorithm. The lattice
algorithm provides a structured method to verify whether a business process is well-defined or not. In addition, it provides means to generate well-defined structural forms for an enterprise through an automated process such as genetic mutation [44]. Furthermore, each enterprise has two main design formalisms: 1) behavioral model (i.e., business model), and 2) supporting communication network models. The enterprise behavioral model is defined by the architecture and its corresponding dynamic models are constructed according to the distributed intelligent systems paradigm. The queuing communication infrastructure is governed by a different set of protocols such as the Internet Protocol (IP). Integrated Design
environments, like OMNeT++, provide a platform for modeling and simulation of queuing networks based on various protocols and mediums (e.g., wired or wireless).

The Resilience framework is used to assess the resilience of the enterprise and the necessary attributes of the framework are measured in the business model (i.e., the Petri Net model) of the multi-model. Therefore, the multi-model runs on a simulation testbed to obtain the parameters needed for calculating different attributes of resilience, this does not have a close form formulation.

The salient experimentation concept behind the concept map (in Figure 14) is to capture the influence of the architectural design on the attributes of resilience when facing a certain disruption in the supporting communication network. In other words, resilience attributes are tuned by varying structural attributes of the enterprise architecture when facing a certain cyber disruption. The current chapter elaborates each of the main concepts presented in Figure 14 through an illustrative example of the federal Aviation Administration’s Enterprise Architecture. Consequently, multiple experiments are performed to capture the impact of modifying the structural form of the architecture on the resilience of the enterprise facing a certain cyber disruption (in a contested cyber environment).

5.1 Enterprise Architecture

Behavioral characteristics, structural attributes, and interactional properties of an enterprise are capture by its architectural design. Although, the architectural artifacts of an enterprise are static, there are methodologies that could be used to convert the architectural artifacts into executable models. These executable models are used for verification and
validation analysis. A comprehensive review of the architecture development, executable model construction, and evaluation is provided in the appendix.

Since, the enterprise architecture of the Federal Aviation Administration (FAA) is available online [66], the architecture design activity was not necessary for the illustrative example. However, the most recent architecture obtained from the National Airspace System (NAS) is constructed using the Structured Analysis and Design Technique (SADT). Therefore, a conversion to the Object-Oriented methodology is necessary to apply the formal method of synthesizing the executable model of the enterprise [10]. Figure 15 shows a partial sequence model of the Air Traffic Control System obtained from the NAS online portal. The enterprise architecture has 18 different sequence models each of which captures an individual operation of the Air Traffic Control (ATC) system. For example,
the operation that is partially depicted in Figure 15 produces a flight plan for departing airplanes that are in a responsibility area of a particular Air Traffic Control Tower (ATCT). The process of issuing flight plan starts by the pilot’s request within the responsibility area of the ATCT. Afterwards, the entities of the ATC system such as ATCT, Terminal RADAR Approach Control (TRACON), Ground Controller, and so forth should interact and perform tasks to generate the output – i.e., the flight plan of the flight. In addition, Figure 16 shows the functionality model of the system using Integrated Computer Aided Manufacturing (ICAM) Definition methods for Function Modeling (IDEF0). This functional model provides a graphical representation of the systems functions as well as their flow of control. For example, “Manage Flight Information” is a function of ATC and it has a control flow to both “Separate Aircraft” and Synchronize Aircraft”. In addition, the mechanisms such as Air Traffic Controller, Surveillance, Automation, etc. are defined. Moreover, controls such as the Federal Aviation Regulations are provided.

As mentioned earlier, these two models are provided by FAA, however, the synthesis method that is used in this research effort [10] requires the architectural artifacts to be expressed in an Object Oriented (OO) language such as UML. Consequently, the original Artifacts, which are developed using the structured analysis approach, are converted to OO models. The conversion method used in this research is introduced by Levis and Wagenhals [7]–[9]. Accordingly, the IDEF0 functional model, the sequence model, Data Flow Diagrams, and so forth are transformed into Unified Modeling Language diagrams such as class diagram, activity diagram, and component diagram. For example, the Operational Node Connectivity Diagram contains the entities of the class model
Figure 16 Top-page IDEF0 model of Air Traffic Control System [66]
such as ATCT and ARTCC. Subsequently, the methods and attributes of each class entity are derived from the Integrated Dictionary’s entries. With the additional support of IDEF0 diagrams and DFDs, the necessary OO architectural artifacts, class diagram and activity diagram are obtained.

The constructed class diagram is displayed in Figure 17. The class model is really extensive; it has 7 main classes that interact with each other according to the defined associations among them. For example, the Aircraft class interacts with Flight Service Station (FSS), Air Traffic Control Tower (ATCT), Air Route Traffic Control Center (ARTCC), and Terminal RADAR Approach Control (TRACON). One of the common attributes in the Aircraft class and the ATCT is Flight Plan request. Consequently, an aircraft can request a flight plan for its destination and upon receiving the request the ATC will process the request accordingly. The process involves interactions between various classes and invokes different operations and as a result consume and/ or produce attributes (i.e., data, material, etc.). These processes are defined in the Activity model of the ATC enterprise (see Figure 18 and Figure 19). The original ATC enterprise architecture presents 16 different threads of activity (i.e., 16 different operational processes) one of which is issuing a Flight Plan. The main process that is implemented is the operation of issuing Flight Plans (as defined in the original enterprise architecture), thus, a separate activity diagram of the particular process thread is provided, as well as more elaborated description of the process and the corresponding dynamic model.

The activity thread of issuing flight plans (see Figure 20) involves three entities of the class
Figure 17 Partial class diagram of ATC enterprise
Figure 18 Activity model of ATC enterprise (top half)
Figure 19 Activity model of ATC enterprise (bottom half)
model ATCT, Aircraft, and TRACON. These three classes interact with each other (as defined in the activity model) and their interactions invoke various methods (operations) and require inputs (attributes). Consequently, each method will produce one or more outputs (attributes). In Figure 20, the process starts with the pilot calling the ground controller for the flight plan (pre-departure clearance request), and then after receiving the request, the ATCT controller will respond to acknowledge receiving the call and the request and proceed with processing the request. The process involves many interactions between ATCT (e.g., ground controller), TRACON controller, and Aircraft pilot and the automation system. This static process is defined using the object oriented architectural
artifacts. In the next step to create the dynamic model of the architecture, the traceable synthesis method of Wagenhals, Haider, and Levis [10] is used to produce the executable model of the architecture.

Before starting the synthesis of the executable model, all the artifacts must put into a specific form. According to the definition of this specific form, each class must either consist of operations or of attributes (but not both of them). In order to illustrate this process a portion of the full architecture that involves the interactions between an aircraft operating in the responsibility area of an ATCT is used. Figure 21 shows the reconstructed segment of the class diagram of the architecture. This partial model consists of two classes: ATCT and Aircraft and the association is bilateral. However, the association class of each direction is different. One simple congruence check is whether all the attributes of the association classes are available in both source and destination classes. Moreover, Figure

![Figure 21 Partial class diagram of the reconstructed FAA's Enterprise Architecture](image-url)
Figure 22 FAA’s TRACON and ATCT classes in standard form

22 shows the class model of Figure 21 in the specific form. Subsequently, the architecture in the specific form is transformed to its corresponding executable model using the synthesis approach introduced by Wangehals et al. [10] a summary of which is provided in Figure 23. The resulting executable model is constructed using the Petri Net discrete event modeling paradigm.

The top-page of the hierarchical executable model of the architecture has seven substitute transitions: 1) Air Traffic Control System Command Center (ATCSCC), 2) Air Route traffic Control Center (ARTCC), 3) Weather Observations Station (WOS), 4) Flight Service Station (FSS), 5) Air Traffic Control Tower (ATCT), 6) Terminal RADAR Approach Control (TRACON), and 7) Aircraft. This high-level view of the executable
model of the FAA’s enterprise architecture is displayed in Figure 24. In addition to the
original class entities, the network federate is also represented using a transition. The
tokens of the pre-place of the network transition are passed to the OMNeT++ federate and
are processed there. Consequently, processed payloads in the network model are placed
into the post-place of the transition. In fact, all the communications between substitute
transitions are facilitated by this network federate. Each substitute transition in the top-
page has its own structure (as defined in the synthesis summary, which is presented in
Figure 23). The fundamental framework of the subpages is constructed based on the five-
stage model of intelligent nodes introduced by Levis [41]. The five-stage model and the
task decomposition and allocation theory [64] are used together in the executable sub-
pages. The atomic tasks in each subpage are assigned to intelligent nodes. In this effort,
each atomic task is assigned to a single intelligent node, modeled using the five-stage
model. Each task in a subpage is assumed to be atomic, which means it cannot be

- Define color sets using the attributes of all the class entities in class
diagram
- Create the hierarchical CPN:
  - Create a substitution transition for each class in the class diagram
  - Create a place for each association and aggregated classes. Assign
    the appropriate color set. If an association is bidirectional, create
    one place for each direction.
  - Create arcs between the substitution transitions and the places
    using the class and activity diagrams.
  - Create sub-page for each substitution transition:
    - Create a transition for each operation
    - Assign the appropriate ports to places
    - Create places for any attributes of the class represented by the
      sub-page
    - Create arcs based on the activity diagram
    - Add arc inscriptions, guard functions, or code segments
      derived from the rules associated with each operation
    - Specify initial markings for each place that represents an aggregate
      class

Figure 23 Process for synthesizing the executable model of the OO architecture

77
decomposed to lower level tasks. In addition, in an adaptive organizational structure, which is elaborated later in this chapter, the switch from one structural form to another structural form changes the task allocation of the nodes. The illustration of this mapping modification is provided in this chapter in the adaptive enterprise section.

Figure 24 Top-page of the hierarchical executable model of FAA's EA
Figure 25 Partial sub-page of a substitution transition
The subpage of a substitution transition of the executable model is displayed in Figure 25. This subpage shows a generic model that applies to all of the subpages. The main parameters in each individual subpage Petri Net are the number of intelligent nodes in the corresponding subpage of the pertinent entity and the tasks allocated to those nodes (this information is obtained from the Role Definition Matrix of the enterprise). In Figure 25, the place named “DMState_1” controls the task assignment of the intelligent nodes of the entity (in the executable model each intelligent node is called a DM). Moreover, in the adaptive organizational structures the various allocations of tasks to DMs are stored in this place and can be controlled based on the anomaly detection system (see Figure 26).

The elements of the Role Definition Matrix (RDM) are the atomic tasks defined in the activity model of the enterprise (figures provided). Each swim lane of the activity diagram captures the activities that the corresponding entity of the class diagram performs as well as their flow of control. The NAS enterprise architecture of the air traffic control
system defines the atomic activities of the enterprise and the overarching static model of the dynamic behavior of the system. However, there are two architectural viewpoint: 1) Functional viewpoint and 2) Physical viewpoint. The architecture designed by NAS provides the functional viewpoint that mainly captures the operational behavior of the enterprise. However, the physical artifacts are missing in the architecture provided by NAS. The physical viewpoint of the architecture is an essential component of this research effort because the impact of different structural forms on the resilience of the enterprise is under investigation. These missing artifacts are produced using expert judgment and added to the publicly available artifacts. In the physical architectural viewpoint, the static arrangement of the intelligent nodes is defined.

The physical viewpoint provides a static view of the physical nodes of the organization. In a distributed intelligence enterprise the physical nodes are modeled as intelligent nodes (using the five-stage model of interacting intelligent nodes). Figure 27 displays one of the physical viewpoints, which is used in this research effort, and referred to as the organizational structural form A (Org. A). This structural form includes 16 intelligent nodes. In addition, these nodes communicate through specific types of links (as

![Figure 27 Physical viewpoint of the ATC enterprise (Org. A)](image-url)
defined in the five-stage model of interacting intelligent nodes). For example, DM1 and DM2 interact using a link of type G, which connects the RS stage of DM1 to the SA stage of DM2. In fact, all of the connections in Org. A are of type G as it is elaborated later in this chapter. This choice of link type impacts the lattice structure of this design. The rationale behind this physical architecture (Figure 27) is that it closely replicates the as-is design of the ATC system, which means that each controller (DM) is assigned to a task and all controllers are using all of the five stages of the intelligent node model (see Figure 28). The latter assumption is normative, and it is not the case all the time [44]. For example, in the organizational form of Org. B (as shown in Figure 29) some of the intelligent nodes only include a few of the five stages (not all of them together), thus, the reduced lattice model changes significantly when compared to the reduced lattice model of Org. A.

The generic process of assigning tasks, which are defined in the activity model (refer to Figure 18 and Figure 19), to the intelligent nodes, which are denoted in the physical model (e.g., Org. A), is shown in Figure 30. According to the Role Definition Matrix (RDM), each role is allocated to at least one intelligent node. For example, Role 1 is assigned to the DM1 only, however, Role 2 is allocated to both DM1 and DM2. In this case, DM2 is the default node responsible for Role 2, but if an anomaly is detected then the DM1 replaces the DM2 and takes Role 2. This replacement is the core mechanism of adaptive enterprises. It enables them to change their operational mode. In addition, the two structural forms that support each operational modes are presented in Figure 30. The default structural form ($\pi_{MNO}^{M}$) considers the diagonal elements of RDM (i.e., Role 1 to
Figure 28 Petri Net model of the physical model Figure 27
DM1 and Role 2 to DM2). The alternative structural form ($\pi^1$) is the physical realization of the first row of the RDM which assigns both Role 1 and Role 2 to DM1.

The flat physical model of a single class (e.g., ATCT) of the ATC system is displayed in Figure 28. This figure shows how big the whole organizational model would be using a flat schematic. Therefore, the model used in this effort is a hierarchical model (please refer to Figure 25 and Figure 26) that is more scalable, and also it is more customizable. Consequently, larger enterprises can be modeled using this hierarchical
design. In addition, various parameters could be adjusted conveniently in this model. For example, various tasks could be added or removed from the model by changing only the tokens in a single place node (see Figure 26). In a similar manner, RDM values (i.e., cell elements) are easily added to this design. Consequently, the top-page of the hierarchical model is presented in Figure 31, which has two entities: 1) aircraft and 2) ATCT. These two entities interact to produce the final outcome (token), which in this effort is the flight plan requested by an airplane. Using the generic process of synthesizing an executable model of the FAA’s enterprise architecture and the physical models of the enterprise (e.g., Org. A and Org. B), the process and results of verifying the FAA’s enterprise architecture are presented.

Figure 31: Top-page of ATC’s executable model
5.1.1 Verifying the Enterprise Architecture

After synthesizing the FAA’s enterprise architecture’s executable model, the soundness of the dynamic model is evaluated. A sound Petri Net always terminates according to the predetermined threads (i.e., end place is reachable from the starting place), thus, the desired results are certainly produced. The soundness property involves both the boundedness of all places and the absence of deadlocks throughout the Petri Net. Consequently, a bounded (safe) and deadlock free Petri Net model shall produce the desired outcomes. A bounded (safe) PN has a limit for the number of tokens in all its places.

In the illustrative example, used in this research effort, the boundedness is already addressed in the lattice algorithm that is incorporated in the design of the structural forms. According to Rule 2 of the structural constraints defined by Remy [5], [43], the structure should have no loops (i.e., structures must be acyclic). Since a Well-Defined Net (WDN) is totally bounded, the evaluation effort is focused on locating deadlocks in the state space of the Petri Net model of FAA’s enterprise architecture.

To identify deadlocks in the executable model of the FAA’s enterprise architecture, the state space of the functional architecture is constructed. The resulting space captures the graph of all possible markings (token configurations) for all possible firing sequences for a given initial condition. The overarching state space is extremely large (with more than 190,000 states) therefore only the set of final states (i.e., deadlocks) is evaluated. A small portion of the state space is shown in Figure 32. In this state space representation, each state is displayed as a node, each of which is representing the place markings of the whole Petri Net model of the architecture. The arc from one state node to another represents the firing of a transition and the resulting change of state. Consequently, a state is reachable
through a particular sequence of firings from a certain state, i.e., the successor state is reachable from the former state. This notion of reachability is pivotal to the behavioral evaluation through model checking. This is because along the reachability path certain behaviors are enabled. Consequently, absence of a particular reachability path means that the related desired behavior is not produced.

A deadlock is represented with a state node that has no output arcs, i.e., a state from which no further transitions are enabled. Therefore, a deadlock is a final state in which all the behavioral processes leading to it are terminated, possibly resulting into a truncated and incomplete process. In the behavioral evaluation, those final states that do not represent the

---

**Figure 32** Partial State Space of the FAA’s Architecture. Ill-terminated states are in bold
successful completion of a desired behavior are identified. The desired end states are denoted in the activity diagram (see Figure 20) by a circle with a black dot in the middle of it. Other than these final states, any other final states are considered as ill-terminated.

Finding the ill-terminated states in the state space of a system is usually a cumbersome task, mainly because the number of nodes is significantly large (Table 1). In order to find the ill-terminated states, model checking was used to explore the entire state space. Temporal logic queries were constructed to search for the final states (i.e., those with no output arcs) and then identify those that are not authorized (i.e., those not defined in the activity diagram). This search is implemented using the Computational Tree Logic (CTL) tool available in the CPNTools package [67].

Consequently, a list of all admissible final states is created first and then the list is used as a look-up table to identify inadmissible end states. Table 1 shows the total number of final states as well as the number of ill-terminated states. The inadmissible or undesired states are traced back to the corresponding conflict place(s) in the Petri Net model. A conflict place suffers from a race condition, i.e., when multiple transitions compete for a particular marking. This is a resource conflict or contention. In addition, an undesired state could be caused by a deadlocked transition that will never be enabled (i.e., required

| Table 1 Statistical results of generating the state space of the partial FAA's architecture |
|---------------------------------|-----------------|-----------------|-----------------|
| SS-graph                      | 190485          | 1261            | 757             |

88
Figure 33 Subpage of ATCT entity
enabling markings are unavailable throughout the runtime). For example, the ill-terminated node with the id number of 10190 in Figure 32 is linked to a conflict at transition “Receive Instructions” which uses markings of places “P2” and “P6” (see Figure 33). Therefore, “Receive Instructions” could fail to execute due to a conflict and that needs to be resolved first at the architecture level and then at the executable model level.

The conflict located in the executable model needs to be traced back to the architectural artifacts. This is facilitated by the traceability property of the synthesis method used. The transition and place(s), which are identified as causing the conflict can be traced to the pertinent operation and attribute(s) in the class diagram as well as activities and control flow in the activity diagram. For example, in the FAA model, the transition “Receive Instruction” and places “P2” and “P6” are traced back to the operation “Receive Instruction” of class ATCT under the swim lane of Supervisor. Similarly, “P2” maps to the Aircraft-Tower association class and “P6” is linked to the control flow between the “Issue Instructions” and “Receive Instruction” activities in the activity diagram.

The final step is to make corrections both in the architecture design and in the executable model. Our procedure for evaluation of the architecture for logical correctness can help in identifying and locating faults in both design artifacts. However, it does not provide any guidance on how to resolve the conflicts. To this end, expert judgment is used in redesigning that part of the architecture. For example in the FAA case, the executable Petri Net model is modified by removing the arc between the place “P2” and transition “Receive Instruction”. The rationale behind this remedy is that the data in place “P2” is already consumed by the transition “Ack. CR” (in Figure 33) and thus the arc from “P2”
to “Receive Instructions” is redundant. Further, in the architectural description, the synchronization point before the transition “Ack. CR” can be moved so the fork happens terminated states are eliminated. Clearly, architecting is essentially an iterative process.

The final Object Oriented description of the Federal Aviation Administration’s Enterprise Architecture (reconstructed from the structured analysis architecture description) is used to synthesize the executable model. Afterwards, this dynamic model is checked for soundness and minor issues are resolved using a structured traceable method [68]. The validated executable model is integrated with the physical executable model (constructed using the five-stage model of decision making) to produce the executable model of the FAA’s ATC enterprise. As a result, the integrated executable model is based on the physical viewpoint, which includes the implementation of the functional viewpoint.

5.2 Enterprise of DMs
The structural model of an enterprise is complementary to its behavioral model. The structural model of the enterprise defines the assignment of tasks to physical nodes (intelligent nodes) that execute those tasks. Since the mapping of functional viewpoint to the physical viewpoint is impactful in the enterprises performance and effectiveness, the structural model of the enterprise is discussed in this section. The structural form is created according to the approach presented in chapter four. The constructed form is mathematically represented by a six-tuple, Π, which is described in chapter four. Each design is analyzed using methods such as the lattice algorithm and state space analysis. The structural model that is depicted in Figure 29, Org. B, is an alternative to the organizational structural form, Org. A (refer to Figure 27) with additional interactions between the
intelligent nodes, while the overall node configuration remains intact. The main idea behind introducing these two structural forms is to assess the influence of adding more interactions between nodes on the resilience of the enterprise when facing a certain cyber disruption. To this end, these two designs (Org. A and Org. B) are analyzed by the lattice set of structures (produced by the lattice algorithm) of the ATC enterprise. Some elements of the lattice set are used to determine the resilience of the organizational structure. The

<table>
<thead>
<tr>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
<th>11</th>
<th>12</th>
<th>13</th>
<th>14</th>
<th>15</th>
</tr>
</thead>
<tbody>
<tr>
<td>P0</td>
<td>T0</td>
<td>P10</td>
<td>T10</td>
<td>P20</td>
<td>T20</td>
<td>P30</td>
<td>T30</td>
<td>P40</td>
<td>T40</td>
<td>P50</td>
<td>T50</td>
<td>P60</td>
<td>T60</td>
<td>P70</td>
</tr>
<tr>
<td>P0</td>
<td>T0</td>
<td>P15</td>
<td>T15</td>
<td>P25</td>
<td>T25</td>
<td>P35</td>
<td>T35</td>
<td>P45</td>
<td>T45</td>
<td>P55</td>
<td>T55</td>
<td>P65</td>
<td>T65</td>
<td>P75</td>
</tr>
<tr>
<td>P0</td>
<td>T0</td>
<td>P15</td>
<td>T15</td>
<td>P25</td>
<td>T25</td>
<td>P35</td>
<td>T35</td>
<td>P45</td>
<td>T45</td>
<td>P55</td>
<td>T55</td>
<td>P65</td>
<td>T65</td>
<td>P75</td>
</tr>
<tr>
<td>P0</td>
<td>T0</td>
<td>P10</td>
<td>T10</td>
<td>P20</td>
<td>T20</td>
<td>P30</td>
<td>T30</td>
<td>P40</td>
<td>T40</td>
<td>P50</td>
<td>T50</td>
<td>P60</td>
<td>T60</td>
<td>P70</td>
</tr>
<tr>
<td>P0</td>
<td>P10</td>
<td>T10</td>
<td>P20</td>
<td>T20</td>
<td>P30</td>
<td>T30</td>
<td>P40</td>
<td>T40</td>
<td>P50</td>
<td>T50</td>
<td>P60</td>
<td>T60</td>
<td>P70</td>
<td>P80</td>
</tr>
<tr>
<td>P0</td>
<td>P15</td>
<td>T15</td>
<td>P25</td>
<td>T25</td>
<td>P35</td>
<td>T35</td>
<td>P45</td>
<td>T45</td>
<td>P55</td>
<td>T55</td>
<td>P65</td>
<td>T65</td>
<td>P75</td>
<td>P85</td>
</tr>
<tr>
<td>P0</td>
<td>P15</td>
<td>T15</td>
<td>P25</td>
<td>T25</td>
<td>P35</td>
<td>T35</td>
<td>P45</td>
<td>T45</td>
<td>P55</td>
<td>T55</td>
<td>P65</td>
<td>T65</td>
<td>P75</td>
<td>P85</td>
</tr>
<tr>
<td>16</td>
<td>17</td>
<td>18</td>
<td>19</td>
<td>20</td>
<td>21</td>
<td>22</td>
<td>23</td>
<td>24</td>
<td>25</td>
<td>26</td>
<td>27</td>
<td>28</td>
<td>29</td>
<td>30</td>
</tr>
<tr>
<td>11</td>
<td>T1</td>
<td>P1</td>
<td>T1</td>
<td>P1</td>
<td>T1</td>
<td>P1</td>
<td>T1</td>
<td>P1</td>
<td>T1</td>
<td>P1</td>
<td>T1</td>
<td>P1</td>
<td>T1</td>
<td>P1</td>
</tr>
<tr>
<td>12</td>
<td>P2</td>
<td>T2</td>
<td>P2</td>
<td>T2</td>
<td>P2</td>
<td>T2</td>
<td>P2</td>
<td>T2</td>
<td>P2</td>
<td>T2</td>
<td>P2</td>
<td>T2</td>
<td>P2</td>
<td>T2</td>
</tr>
<tr>
<td>13</td>
<td>P4</td>
<td>T4</td>
<td>P4</td>
<td>T4</td>
<td>P4</td>
<td>T4</td>
<td>P4</td>
<td>T4</td>
<td>P4</td>
<td>T4</td>
<td>P4</td>
<td>T4</td>
<td>P4</td>
<td>T4</td>
</tr>
<tr>
<td>14</td>
<td>P6</td>
<td>T6</td>
<td>P6</td>
<td>T6</td>
<td>P6</td>
<td>T6</td>
<td>P6</td>
<td>T6</td>
<td>P6</td>
<td>T6</td>
<td>P6</td>
<td>T6</td>
<td>P6</td>
<td>T6</td>
</tr>
<tr>
<td>15</td>
<td>P8</td>
<td>T8</td>
<td>P8</td>
<td>T8</td>
<td>P8</td>
<td>T8</td>
<td>P8</td>
<td>T8</td>
<td>P8</td>
<td>T8</td>
<td>P8</td>
<td>T8</td>
<td>P8</td>
<td>T8</td>
</tr>
</tbody>
</table>

Table 2 Simple paths of organizational structure B. The top row numbers show the node counts and the three rows below the first row denotes the elements in each simple path. Since the simple path are very long they are broken into shorter lines of 15 elements each.

92
The main elements of interest are the lower order elements of the lattice set and the high order elements of the lattice. The analysis is performed on the to-be structural form (or organization B) that is depicted in Table 2. The to-be structure is essentially the as-is form (Org. A) with two additional simple paths. The five available simple paths of organization B are shown in the first column on the upper right hand corner of Table 2. Consequently, the lattice set of organization B covers the elements of the lattice set of organization A. The boundary elements of the to-be design, i.e., MAXOs and MINOs, are calculated using the CAESAR III software package. As shown in Figure 37, the lattice set of organizational structure B has five MINO elements, each of which is a single path of the enterprise model.

There are three layers of intermediate structures (larger elements than MINO but smaller

| Table 3 Simple paths of organizational structure A. The top row numbers show the node counts and the three rows below the first row denotes the elements in each simple path. Since the simple path are very long they are broken into shorter lines of 15 elements each |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
| P0 | T0 | P10 | T10 | P20 | T20 | P30 | T30 | P40 | T40 | P5011 | T11 | P21 | T21 | P31 |
| P0 | T0 | P15 | T15 | P25 | T25 | P35 | T35 | P45 | T45 | P5561 | T61 | P61 | T61 | P61 |
| P0 | T0 | P75 | T75 | P85 | T85 | P95 | T95 | P105 | T105 | P115 | T115 | P125 | T125 | P125 |
| 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 | 30 |
| T31 | P41 | T41 | P5121 | T12 | P22 | T22 | P32 | T32 | P42 | T42 | P5231 | T13 | P33 | T23 |
| T36 | P46 | T46 | P5671 | T17 | P27 | T27 | P37 | T37 | P47 | T47 | P5781 | T18 | P28 | T28 |
| T36 | P46 | T46 | P5671 | T17 | P27 | T27 | P37 | T37 | P47 | T47 | P5781 | T18 | P28 | T28 |
| 31 | 32 | 33 | 34 | 35 | 36 | 37 | 38 | 39 | 40 | 41 | 42 | 43 | 44 | 45 |
| P33 | T33 | P43 | T43 | P54 | T44 | P5 | T5 | P6 | T6 | P7 | T7 | P8 | T8 | P9 |
| P38 | T38 | P48 | T48 | P5891 | T19 | P29 | T29 | P39 | T39 | P49 | T49 | P5911 | T111 | P211 |
| P38 | T38 | P48 | T48 | P58101 | T110 | P210 | T210 | P310 | T310 | P410 | T410 | P5111 | T111 | P211 |
| 46 | 47 | 48 | 49 | 50 | 51 | 52 | 53 | 54 | 55 | 56 | 57 | 58 | 59 | 60 |
| T211 | P311 | T311 | P411 | T411 | P511121 | T112 | P212 | T212 | P312 | T312 | P412 | T412 | P512131 | T113 |
| T211 | P311 | T311 | P411 | T411 | P511121 | T112 | P212 | T212 | P312 | T312 | P412 | T412 | P512131 | T113 |
| 61 | 62 | 63 | 64 | 65 | 66 | 67 | 68 | 69 | 70 | 71 | 72 | 73 | 74 | 75 |
| P213 | T213 | P313 | T313 | P413 | T413 | P513 | T314 | P214 | T214 | P314 | T314 | P414 | T414 | P51451 |
| P213 | T213 | P313 | T313 | P413 | T413 | P513 | T314 | P214 | T214 | P314 | T314 | P414 | T414 | P51451 |
| 76 | 77 | 78 | 79 | 80 | 81 | 82 | 83 | 84 | 85 | 86 | 87 | 88 | 89 | 90 |
| T115 | P215 | T215 | P315 | T315 | P415 | T415 | P515 | T5 | P6 | T6 | P7 | T7 | P8 | T8 |
| T115 | P215 | T215 | P315 | T315 | P415 | T415 | P515 | T5 | P6 | T6 | P7 | T7 | P8 | T8 |
than MAXO) with 25 elements in total. In addition, there is only one MAXO structure that consists of all five simple paths. In addition, Table 3 shows the simple path for organization A. As a comparison between organization A and organization B, the simple paths of the two organizations are compared side by side. The three simple paths of the organization A (see Figure 38) are available in organization B’s simple paths as well. In essence, organization B includes all the simple paths of organization A plus two more additional paths.

In the lattice of the organization B, presented in Figure 37, there are structural elements that do not contain all the 16 intelligent nodes (these elements are in dashed gray). This means that certain structures do not include one or more of the DMs, even in a partial manner (i.e., few stages of the five-stage model are available). Therefore, an additional constraint is applied to the lattice set to filter the valid structural forms. The methodology
presented in this chapter requires that all of the DMs must be present at least in a partial manner. Consequently, the invalid structural elements (the ones in dashed gray) are removed from the original lattice set and the lattice set of remaining elements that is referred to as the truncated lattice set (with only black elements). Similarly for organization A, truncated lattice set is constructed from the original lattice set shown in Figure 38. It is obvious from this figure that the invalid structural forms (in dashed gray) are removed from the original set and the resulting truncated set only has the MAXO element that is the same as the $\Pi^{11}$ structural form in the lattice set of the organization B. In summary, the truncated set of organization A has one element, $\Pi^{11}$, and thus it is like one of the elements of the truncated lattice of organization B, Figure 39.

The salient property of the truncated lattice set is that it satisfies the convexity criterion for lattice sets (please refer to chapter four). The convexity of the truncated lattice
set is observable from the graphical representation of the lattice set (Figure 39), where the MAXO structure is reachable from all of the other elements, e.g., \( \Pi^4 \).

Another observation from the lattice structure of the organization B (shown in Figure 37) is that the organization A’s structural form is an element in the lattice. Therefore, both organizational structures A (in Figure 27) and B (in Figure 29) belong to the same lattice set. Organizational structure A is the structural element, \( \Pi^{11} \), of the truncated lattice and organizational structure B is the \( \Pi^{MAXO} \). This property has happened by design, and it is the result of adding simple paths of type F to the design A to generate structure B. As a result, we can investigate the impact of adding additional simple path on the organizational structure’s resilience. In the following section the resilience of these two structures is evaluated.
5.3 Resilience of the Elements of the Lattice

The resilience metrics are either calculated or estimated for each element of the truncated lattice set. In this research effort, only two elements are evaluated: 1) organization A, element $\Pi^{11}$ and 2) organization B, element $\Pi^{MAXO}$ (Figure 37). All three attributes of resilience capacity, tolerance, and flexibility, which were reviewed in chapter three, are assessed for organizations A and B. Afterwards, the evaluation results are compared in order to understand their relative quantified resilience.

There are two categories of resilience metrics in the resilience framework: 1) graph theoretic metrics and 2) performance related metrics. The graph theoretic metrics are calculated from the bipartite graph properties of the organizational structures. The performance metrics such as timeliness are estimated through simulation that is carried out with the executable multi-model of the enterprise. The reason behind this evaluation method is that there is no close form method available to calculate such metrics. The results

<table>
<thead>
<tr>
<th>Metric</th>
<th>Measures</th>
<th>Question Answered</th>
<th>Org. A</th>
<th>Org. B</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fault Tolerance</td>
<td>The ratio of simple functionalities which may be disrupted without a loss of capability to the total number of simple functionalities.</td>
<td>How many simple functionalities can be disrupted prior to losing the capability. Primarily a tool to draw architects attention to key areas in the design.</td>
<td>0.67</td>
<td>1</td>
</tr>
<tr>
<td>Point of Failure</td>
<td>Relatedness of failures at the element level to an overall loss of capability</td>
<td>Are element level failures relatively localized, or do failures incur broad system-level effects? Primarily a tool to draw the architect's attention to key design areas.</td>
<td>0.44</td>
<td>0.15</td>
</tr>
<tr>
<td>Cohesion</td>
<td>Relatedness of the elements within a node or module which support a given capability</td>
<td>How difficult is it to reorganize the system at the node / module level?</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Common Use</td>
<td>Extent of common use of the elements among the simple functionalities which support the overall capability.</td>
<td>Can a system execute multiple functionalities concurrently, or is it limited by competition for resources?</td>
<td>1.59</td>
<td>1.58</td>
</tr>
<tr>
<td>Proportion of Use</td>
<td>The ratio of the total elements used by any given simple functionality to deliver the overall capability.</td>
<td>Are most of the elements needed for a given functionality, making it more difficult to reorganize?</td>
<td>0.53</td>
<td>0.49</td>
</tr>
</tbody>
</table>
of the graph theoretic measures are shown in Table 4. Organizational structure B has improved resilience metrics in comparison to organization A, with the exception of Point of Failure Tolerance metric. However, since the other tolerance metric, Fault Tolerance, is improved (for organization B) then the tolerance metric has improved in general for organization B. Fault Tolerance is a comprehensive metric because it measures the tolerance of links rather than points. In summary, Fault Tolerance is a more important measure than the Point of Failure Tolerance. A similar pattern and the same explanation is also provide by Pflanz in his dissertation [3]. Therefore, the resilience of organization B is improved from a graph theoretic point of view when compared to the organization A’s structure. In addition, these results show that adding two simple paths to structure A positively influences its resilience when only considering graph theoretic metrics.

All of the Performance metrics do not have close forms, and thus they can only be estimated using modeling and simulation approaches. In this research effort, as mentioned earlier, a multi-modeling testbed is used to capture the impact of a certain scenario of availability cyber disruption on distributed intelligence enterprises (i.e., organizations A and B). Afterwards, performance related resilience metrics can be calculated and compared to infer the resilience properties of different structural forms. In the next section, the process of constructing the multi-model is provided, and then the performance metrics calculations and results are presented.

5.4 Multi-model construction

Models, constructed according to different paradigms, provide their own unique insights about the original phenomenon they represent. Therefore, each model fits a
specific purpose. However, in a very complex problem, it is usually valuable to federate multiple views each of which is obtained from a specific model to generate a holistic view of the problem with enhanced fidelity to the nuances of the original problem [69]. For example, in this work, a complex enterprise that performs multiple business operations is supported by a queuing network for its communication needs. A discrete event model is constructed to model the business processes of the enterprise. This discrete event model is constructed using the five-stage model of intelligent nodes, which was presented in chapter 3. In addition to the discrete event model of business processes, the supporting network infrastructure is modeled also using a discrete event environment according to the internet protocol (IP) framework. The network model will provide maximum fidelity to the true traffic patterns of a real queuing network infrastructure. Consequently, a high fidelity model in OMNeT++ is created to realize the packet level behavior of the supporting communication infrastructure. Subsequently, the supporting infrastructure model (developed in OMNeT++ environment) is federated with the discrete event model of the business processes to enhance the fidelity of modeling and simulation to the original problem.

The two models interoperate via a run-time infrastructure (RTI) that manages the execution of the overarching multi-model. In addition, the infrastructure facilitates the re-use of components, which enhances the synthesis process significantly from a temporal point of view. Furthermore, each model participates in the multi-model through a federate, and thus the holistic multi-model is denoted as a federation.
The multi-model is comprised of three major parts: 1) a discrete event model of business processes of the enterprise, 2) a discrete event model of the queuing network (using OMNeT++), and 3) the Infrastructure Wind Tunnel. Each of the two models has its own federate to interact with the Wind Tunnel test-bed. In addition, the test-bed also hosts an enterprise federate and an infrastructure federate. In the remaining of this chapter, these three components and also the implementation of the cyber perturbations are presented.

5.4.1 Discrete Event Model of an Enterprise
The discrete event model of the enterprise is constructed according to the process that is elaborated in the Appendix. Since this research effort is focused on distributed intelligence organization, the five-stage model of intelligent nodes is used to create the physical executable model of the organization. The process of creating the discrete event model starts with the object oriented architecture of the organization such as the transformed enterprise architecture of the Federal Aviation Administration presented in chapter 3. The next step is to synthesize the executable model of the enterprise according to the methodology of Wagenhals et al. [10]. In parallel to constructing the executable model of the processes, the executable model of the physical nodes is created based on the physical architectural viewpoint of the system. The Petri Net model of the nodes is developed according to the approach introduced by Levis [41], which was later automated by a software package developed by the Systems Architecture Laboratory (SAL). CAESAR III is a tool for the design and analysis of organizations. It enables the design of distributed intelligence organizations and the completed designs can be converted into the corresponding executable Petri net model. Furthermore, it provides various kinds of
analyses such as the application of the lattice algorithm and simple path calculations. The process of generating the complete executable model of the organization is depicted in Figure 41.

The executable model of processes and the physical model of intelligent nodes are mapped together using the additional structural information such as the role definition matrix (RDM) in the adaptive organizations. The non-adaptive enterprise is a special case of the adaptive organization where the role definition matrix is an identity matrix; the allocation of roles to decision-makers is one-to-one.

In an adaptive organization, multiple structural forms are possible for the executable model. These structural forms are derived by the lattice algorithm. Consequently, the decision on how a structural form should be selected among all of the possible options is one of the important questions that is addressed by this research effort. The selection of the right structural form for the prevailing situation is not a trivial decision to make. Figure 42 shows a notional lattice for a hypothetical organization, however, an organization can have multiple MINOs and MAXOs (rather than one as depicted in this figure). The structural form that each node of the lattice represent is shown in Figure 30;
this figure also demonstrates how the functional executable model and the role definition matrix are mapped to each structural form in the lattice.

For each structural form in the truncated lattice set, one or more roles are assigned to each intelligent node (decision maker) according to the role definition matrix. Usually, there is a one-to-one relationship between tasks and roles [41]. As a result, task allocation (i.e., equivalent to role allocation) is a part of the design process presented in Figure 41. Task decomposition is a hierarchical activity and it starts from a high level mission all the way down to atomic tasks that are generated in a nested parsing process [64]. In this dissertation, the real world application of FAA’s enterprise architecture is used, thus, the functional decomposition is already provided by National Airspace System (NAS) and the results are used as-is. Otherwise, task decomposition and allocation would be a critical part of organizational design and the corresponding executable models.
5.4.2 Discrete Event Model of the Queuing Network

The supporting queuing network is modeled using the domain specific environments, OMNeT++, to replicate the real traffic pattern in the communication network of the ATC enterprise. The objective Modular Network Test-bed in C++ [22] captures the underlying components of the network and replicates their packet level protocols and dynamics. Therefore, the highest level of fidelity to the real traffic patterns is achieved. An OMNeT++ model uses two types of modules: 1) simple module and 2) compound module. As its name suggests, a simple module is a single atomic component. Moreover the compound module is a combination of multiple simple modules and/ or other compound modules. Figure 39 shows the simple and compound modules of a compound

Figure 39 Compound module of each terminal in OMNeT++
module, regHost (see upper left corner of Figure 39). The modules are predefined in INET Framework that is used in this work. This framework has built-in compound modules of all the different components of wired and wireless communication networks such as routers, switches, terminals, and so forth. The network infrastructure that is modeled and used in this research effort is adopted from previous work of Levis et al. [32]. Figure 44
shows a high level topology of the supporting network infrastructure developed in the current effort.

Afterwards, the OMNeT++ model is federated with the business processes through the run-time infrastructure (i.e., poRTIco [29]) to generate the overarching representation of the dynamic behavior and physical aspects of the organization. In addition, in Figure 39, individual terminals are linked to their associated intelligent node in the enterprise model. In essence, the communications between decision makers are realized as the communication between their designated computer terminals.

Discrete Event Model of the Cyber Perturbation

There are two main types of cyber disruptions: 1) availability attacks (i.e., denial of service) and 2) integrity attacks. This dissertation is focused on availability disruptions. In general, an availability attack causes delay in some localities of the network. Therefore, the perturbation that is caused by such attacks usually involves slowing down some elements of the network such as routers, links, switches, and so forth. In extreme situations, a network asset may be disabled (i.e., infinite delay). In Table 5, a summary of cyber exploits and the descriptions of their implementations are depicted.

<table>
<thead>
<tr>
<th>Cyber Exploits</th>
<th>Implementation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Disable Node</td>
<td>Completely disable a network node</td>
</tr>
<tr>
<td>Disable Network</td>
<td>Completely disable a sub-network</td>
</tr>
<tr>
<td>Disable Link</td>
<td>Completely disable a network link</td>
</tr>
<tr>
<td>Delay Node</td>
<td>Slow down a network node</td>
</tr>
<tr>
<td>Delay Link</td>
<td>Slow down a network link</td>
</tr>
</tbody>
</table>
The implementation of the cyber perturbations (provided in Table 5) is coded as a compound module, denoted as Scenario Manager, and executed as a predefined scenario. In this work availability perturbations are reproduced by specifying the locality of the disruption, i.e., the asset(s) such as routers, and temporal specific actions such as start and end times. In addition, the changes in the asset’s attributes, e.g., the delay, can be specified (see Figure 41). For example, the node delay that is used in this work, slows down the link connected to the decision maker two (DM2 in ATCT) from simulation time 4:30 to 23:00.

The corresponding network asset (i.e., router) that is affected is displayed in Figure 42.

5.4.3 Synthesizing the Multi-model
The interoperation of the two models is facilitated using a model-based simulation integration platform. A modified version of the C2 Wind Tunnel (C2WT) [70] is used to facilitate the execution of the multi-model. The C2WT was updated by adding Access/CPN [71] in place of the BRITNeY Suit [72], which is not supported anymore. Access/CPN provides a Java API to interact with the CPNTools environment. A sample Java code that
uses Access/CPN is provided in Figure 43. This code extracts the marking of a certain place to be sent to OMNeT++ model. Access/CPN is a framework with a large set of versatile functions such as reading the markings of a specific place and manipulating it. Similarly, it is possible to put tokens in a specific place. In fact, the CPNTools’ interface
with RTI (ambassador) does a similar process; it reads a token from the pertinent place and passes it to the OMNeT++ ambassador through the RTI. After the token is processed by the OMNeT++ model, then it is passed to the CPNTools ambassador through the run-time infrastructure and is put in the particular place that is designated for that. In addition, time is controlled by RTI to make sure the current time is the same in both models, i.e. they are not lagging or leading each other.

The RTI communicates with OMNeT++ using a C++ API. As in CPNTools, the network infrastructure model interacts with the RTI and the CPN model through its ambassador. The pseudo code of the federation (i.e., the multi-model) is presented in Figure 44. In essence, a federate joins a federation, subscribes to an object, and interacts with another object(s) after the creation of the federation and the federate. In the end of the process, the federation is terminated, and all of federates are trashed. Figure 45 shows a

```
marking[n] = simulator.getMarking(cimulator.getAllPlaceInstances().get(n).toString());
```

Figure 43 Obtaining markings of places in Access/ CPN

<table>
<thead>
<tr>
<th>Synopsis of the Run-time infrastructure:</th>
</tr>
</thead>
<tbody>
<tr>
<td>• Declare the ambassador(s)</td>
</tr>
<tr>
<td>• Form the Federation</td>
</tr>
<tr>
<td>• Join the Federation</td>
</tr>
<tr>
<td>• Publish and Subscribe (between federates)</td>
</tr>
<tr>
<td>• Terminate from the Federation</td>
</tr>
<tr>
<td>• Kill the Federation</td>
</tr>
</tbody>
</table>

Figure 44 Pseudo code of the run-time infrastructure
high-level view of the multi-model created with organizational and network federates. All the tokens are being passed between the organizational entities (Petri Net model) are routed through the infrastructure (OMNeT++ model). While traveling in the infrastructure, tokens
experience the availability disruption that is implemented in the network infrastructure model.

Lastly, the inter-arrival time of the incoming calls (requesting flight plans) are obtained from the real-time data of the incoming calls from Dulles International Airport [73] (see Table 6). The assumption is that each departure represents a flight plan request call, thus, each row of in Table 6 denotes a flight plan request call. Subsequently, a Poisson parametric distribution is fitted to the inter-arrival time distribution (of real-time data). Figure 46 shows the fitted Poisson distribution, in which the mean interval time is almost 1.5 min. The input tokens are generated using this Poisson distribution. These tokens are processed by the multi-model of the enterprise and its supporting communication network.

Using the constructed multi-model, the performance related resilience metrics (i.e., capacity metrics) for organizations A and B are estimated. The measure of performance for

<table>
<thead>
<tr>
<th>Ident</th>
<th>Type</th>
<th>Destination</th>
<th>Departure</th>
<th>Estimated Arrival</th>
</tr>
</thead>
<tbody>
<tr>
<td>N68BJ</td>
<td>PAY2</td>
<td>Williamsport Rgnl (K IPT)</td>
<td>Sat 04:40PM EDT</td>
<td>Sat 05:24PM EDT</td>
</tr>
<tr>
<td>UAL542</td>
<td>A320</td>
<td>Denver Intl (K DEN)</td>
<td>Sat 04:39PM EDT</td>
<td>Sat 06:06PM MDT</td>
</tr>
<tr>
<td>JIA5031</td>
<td>CR9</td>
<td>Charlotte/Douglas Intl (K CLT)</td>
<td>Sat 04:30PM EDT</td>
<td>Sat 05:20PM EDT</td>
</tr>
<tr>
<td>UAL807</td>
<td>B772</td>
<td>Beijing Capital Intl (PEK / Z BAA)</td>
<td>Sat 04:28PM EDT</td>
<td>Sun 05:28PM CST</td>
</tr>
<tr>
<td>AVA247</td>
<td>A319</td>
<td>El Dorado Intl (BOG / SK BO)</td>
<td>Sat 04:24PM EDT</td>
<td>Sat 08:01PM COT</td>
</tr>
<tr>
<td>UCA4982</td>
<td>DH8C</td>
<td>Richmond Intl (K RIC)</td>
<td>Sat 04:09PM EDT</td>
<td>Sat 04:50PM EDT</td>
</tr>
<tr>
<td>TMC460</td>
<td>BE40</td>
<td>Francis S Gabreski (K FO K)</td>
<td>Sat 03:53PM EDT</td>
<td>Sat 04:53PM EDT</td>
</tr>
<tr>
<td>UAL1603</td>
<td>A320</td>
<td>Seattle-Tacoma Intl (KSEA)</td>
<td>Sat 03:51PM EDT</td>
<td>Sat 05:50PM PDT</td>
</tr>
<tr>
<td>UAL1574</td>
<td>B739</td>
<td>San Diego Intl (KS AN)</td>
<td>Sat 03:50PM EDT</td>
<td>Sat 06:09PM PDT</td>
</tr>
<tr>
<td>DLH417</td>
<td>B744</td>
<td>Frankfurt Intl (F RA / E DDF)</td>
<td>Sat 03:48PM EDT</td>
<td>Sun 04:43AM CEST</td>
</tr>
<tr>
<td>UAL209</td>
<td>B739</td>
<td>Los Angeles Intl (KL AX)</td>
<td>Sat 03:38PM EDT</td>
<td>Sat 05:52PM PDT</td>
</tr>
<tr>
<td>UCA4949</td>
<td>DH88</td>
<td>Charlottesville-Albemarle (K CHO)</td>
<td>Sat 03:36PM EDT</td>
<td>Sat 03:58PM EDT</td>
</tr>
<tr>
<td>AAL2541</td>
<td>MD83</td>
<td>Dallas/Fort Worth Intl (K DFW)</td>
<td>Sat 03:18PM EDT</td>
<td>Sat 04:58PM CDT</td>
</tr>
<tr>
<td>JBU156</td>
<td>E190</td>
<td>Boston Logan Intl (KBOS)</td>
<td>Sat 03:14PM EDT</td>
<td>Sat 04:28PM EDT</td>
</tr>
<tr>
<td>UAL560</td>
<td>B738</td>
<td>San Francisco Intl (KS FO)</td>
<td>Sat 02:56PM EDT</td>
<td>Sat 05:05PM PDT</td>
</tr>
<tr>
<td>N400MP</td>
<td>GLF4</td>
<td>Boston Logan Intl (KBOS)</td>
<td>Sat 02:54PM EDT</td>
<td>Sat 04:16PM EDT</td>
</tr>
<tr>
<td>WML61</td>
<td>L160</td>
<td>Manassas Rgnl (KHE F)</td>
<td>Sat 02:51PM EDT</td>
<td>Sat 03:01PM EDT</td>
</tr>
<tr>
<td>SWA3366</td>
<td>B737</td>
<td>McCarran Intl (KL AS)</td>
<td>Sat 02:28PM EDT</td>
<td>Sat 04:01PM PDT</td>
</tr>
<tr>
<td>EJA211</td>
<td>CL60</td>
<td>Baltimore/Washington Intl (KB WI)</td>
<td>Sat 02:11PM EDT</td>
<td>Sat 01:18PM EDT</td>
</tr>
</tbody>
</table>
the example of FAA ATC’s EA is the average number of flight plans generated per hour. The availability disruption occurs at hour 4:30 and continues until 23:00; during the disruption the router linked to DM 2 slows down. Figure 47 shows the average number of flight plans issued during one day. The main observation is that there is large response gap due to the disruption, thus, number of flight plans issued drops by 90%. Similarly, Figure 48 shows the average number of flight plans generated in each hour by organization B during a single day. There are two salient observations. First, the response lag during the cyber disruption is significantly smaller when comparing to organization A, and this could be explained by additional simple paths that are available in organization B. Second, the pre-disruption performance of the organization B is lower than organization A; this higher latency is because of the added simple paths that increase the total latency of the enterprise.
By using the measure of performance of the FAA’s ATC enterprise, the capacity related resilience metrics are calculated. Table 7 displays the summary of the estimated metrics. A very important point is that Buffering Capacity of organization B is lower than organization A, which, in other words, means that organization B is underperforming under normal conditions (pre-disruption). Conversely, Residual Capacity of organization B is improved; this higher capacity in this case means that the enterprise is better performing in comparison to organization A in the face of exactly the same cyber disruption.

The Comparison of the both graph theoretic and performance related resilience metrics provides three interesting insights. First, organization A (with $\Pi^{11}$ structure) performs better during the pre-disruption phase in comparison to organization B. Second,
organization B (with MAXO structure) performs better post-disruption. And finally, there is a dichotomy between organizations A and B. Each organizational structure works better in an operational phase, i.e., pre-disruption and post-disruption. Therefore, the MAXO structure performs better in the post-disruption phase and the lower order structural form operates more efficiently during pre-disruption. The salient question here is which

![Flight Plan Generation MoP (Org. B)](image)

Figure 48 Average number of flight plans issued by organization B in each hour of a day

<table>
<thead>
<tr>
<th>Metric</th>
<th>Measures</th>
<th>Question Answered</th>
<th>Org. A</th>
<th>Org. B</th>
</tr>
</thead>
<tbody>
<tr>
<td>Buffering Capacity</td>
<td>Available capability margin between current operating levels and a defined minimum threshold operating level at the time preceding a disruption.</td>
<td>Can a disruption be absorbed with immediately available (on-hand) resources?</td>
<td>0.5</td>
<td>0.3</td>
</tr>
<tr>
<td>Residual Capacity</td>
<td>Available capability margin between operating levels at the end of the survival phase and a defined minimum threshold operating level.</td>
<td>Given survival, how vulnerable is the system to a follow-on disruption that occurs before the system can recover?</td>
<td>0</td>
<td>0.38</td>
</tr>
</tbody>
</table>

Table 7 Summary of performance related resilience metric
organizational structure (A or B) is more suitable for the enterprise. Before answering this question, the adaptive enterprise is presented as another alternative.

5.5 Adaptive Enterprise

An adaptive enterprise has multiple modes of operation or in other words it supports multiple physical structural forms [4]. In the case of FAA’s EA application presented throughout this chapter, organizational structure A and B are among the structures (or enabled modes of operation). Therefore, the adaptive enterprise (referred also as organization C) switches from structure A to structure B as soon as an anomaly is detected in the communications between the intelligent nodes (DMs) of the enterprise. As denoted before, the anomalies are caused by cyber disruptions. The specific cyber disruption that is elaborated previously and used for organizations A and B is also used for the adaptive organization.

One important part of designing an adaptive enterprise is the assignment of roles to decision-makers (or controllers in the ATC enterprise). The fundamental concept behind the concept of role is that it adds a level of transparency between tasks and their corresponding intelligent nodes. Therefore, there is not a direct linkage between tasks and the intelligent nodes anymore. This added level of abstraction enables the virtualization of the intelligent nodes. The notion of virtualization is critical in adaptive enterprises and it is discussed in section 5.5.1 (i.e., Anomaly Detection section) from different perspectives. In essence, in an adaptive enterprise the business processes, i.e., the task layout, remains intact, however, the intelligent nodes (or some of the intelligent nodes based on the role
definition matrix) are virtualizing one another, i.e., performing each other’s roles. Figure 49 shows the role definition matrix (RDM) for the adaptive ATC enterprise. Based on the

<table>
<thead>
<tr>
<th>Role 1</th>
<th>Role 2</th>
<th>Role 3</th>
<th>Role 4</th>
<th>Role 5</th>
<th>Role 6</th>
<th>Role 7</th>
<th>Role 8</th>
<th>Role 9</th>
<th>Role 10</th>
<th>Role 11</th>
<th>Role 12</th>
<th>Role 13</th>
<th>Role 14</th>
<th>Role 15</th>
<th>Role 16</th>
</tr>
</thead>
<tbody>
<tr>
<td>Node 1</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Node 2</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Node 3</td>
<td>2</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Node 4</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Node 5</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Node 6</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Node 7</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Node 8</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Node 9</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Node 10</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Node 11</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Node 12</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Node 13</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Node 14</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Node 15</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Node 16</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Figure 49 Role Definition Matrix (RDM) for the to-be organizational design

RDM nodes 2 and 3 can virtually replicate each other and the same case is true for nodes 10 and 11. For example, role 3 is originally assigned to node 3, however, in abnormal situations node 2 can replicate node 3 and take over role 3. Each role is associated with

![Figure 50 State space of the structural forms of the adaptive organization (to-be)](image-url)
one atomic task or an arrangement of atomic tasks that are not done in parallel. The state space of the modes of operations (created based on Perdu’s notation [4]), which in this effort denotes the mapping between structural forms and RDM’s roles assignments, is depicted in Figure 50. The state space of the adaptive enterprise has four different mappings that is a considerable expansion to the only state of static organizations such as organizations A (state one in Figure 50). Actually, the main influencer of the resilience is the flexibility that is provided by the extended state space in adaptive enterprises. The state space demonstrates the native state, i.e. state 1, and other states with virtualized intelligent nodes, for example, in state 3, node 3 replicate node 2 and takes over role 2. In order to execute the adaptive enterprise, the conditions, at which a transition between different structures trigger must be identified. An anomaly detection mechanism is responsible to initiate the transition from one mode to another according to the state space. The anomaly detection mechanism is explained in the next section, and in addition the performance and resilience results of the adaptive enterprise (organization C) are presented.

5.5.1 Anomaly Detection
The fundamental role of the anomaly detection subsystem is to sense unusual behavior of the supporting network infrastructure, and then initiate a transition to a new more resilient structural form (see Figure 51). It must be reemphasized that the anomaly detection mechanism operates at the enterprise level, thus, the signature of any cyber exploit is not transparent to this subsystem – that is the reason that unusual infrastructure behaviors are not distinguished and all of them are considered as cyber perturbations. There is one unusual behavior of the cyber infrastructure that is of interest in this research effort
The excessive communication delay is a major anomaly that is sensed using some monitors in the Petri Net model of enterprise. These monitors generate live event logs that are used to detect anomalies. Event logs are used to detect excessive communication delay. The metric for communication delay is the sojourn time that was discussed in chapter four. The main concept is that if a token is waiting for another token more than 15 minutes (the threshold of excessive delay) then an anomaly alert is issued and the transition from one model to another starts. The adaptive enterprise (organization C) is tested under the same availability disruption (in the contested cyber environment) that organizations A and B were tested. Figure 52 shows the average number of flight plans issued per hour by the organization C; there are two important observations form this figure. First, the pre-disruption performance of organization C is close to that of organization A and obviously higher than organization B. Second, the post-disruption performance of organization C is close to organization B and much higher than organization A. Therefore, the adaptive organization C provides a balance between the pre-disruption and the post-disruption performance of the ATC enterprise.
Figure 52 Average number of flight plans issued by organization C in each hour of a day
CHAPTER 6: CONCLUSIONS AND FUTURE WORK

6.1 Conclusions
The rapid increase in the number of complex cyber threats caused serious concerns about the damage they can cause the government and private agencies. Meanwhile, the current technologies fall short to provide comprehensive and reliable shield against cyber threats. Therefore, a systematic design that helps systems endure cyber threat looks extremely valuable and promising. This concept is commonly referred to as resilience of a system to a certain cyber threat under particular conditions.

On one hand, the problem of evaluating a distributed intelligence enterprise that is supported by a queuing communication network is modeled using the five-stage model of intelligent nodes. The rules that govern the Well Defined Nets (WDN) are satisfied by the enterprise designs under consideration. In addition, the enterprise model could be linked to a high fidelity model of the supporting queuing network to model the interoperations between the enterprise model and the communication infrastructure model.

On the other hand, assessing the resilience of an information processing enterprise has been quantified using the first (and the only) resilience framework available. This framework enables us to evaluate the resilience of individual designs, and then compare them. The side-by-side comparison enable us to select the design with better resilience attributes. However, what parameters influence the resilience attributes of an enterprise has not been investigated yet.
This dissertation investigated the attributes that may have an impact on the resilience of an enterprise. Moreover, the parameters are formulated by a methodology that guides enterprise architects to improve resilience attribute of the enterprise as needed. This study considered the tradeoff that exists in the design space in order to ensure a balanced design in the end. The attribute of interest in this research effort was configuration of the physical intelligent nodes (i.e., inter-element collaboration), thus, the influence of inter-element collaboration on the resilience attributes was investigated.

The real world application of Federal Aviation Administration’s Enterprise Architecture for the Air Traffic Control System is used to evaluate and demonstrate the methodology of assessing the resilience of the enterprises (e.g. the ATCS) when facing a cyber disruption (in a contested cyber-environment). The impact of different physical structures (with different inter-element collaboration patterns) is demonstrated through the ATCS case.

The study of different elements of the lattice set shows that upper bound elements (e.g. MAXOs) are more resilient from the graph theoretic point of view. Alternatively, the lower bound elements (e.g., MINOs) exhibit lower resilience attribute as well as lower latency. Therefore, there exists a tradeoff between resilience attributes (graph theoretic attributes) and the latency of the bipartite graph of the enterprise model.

Ultimately, the adaptive model of the enterprise is tested and compared with the previous non-adaptive enterprise structures. The adaptive enterprise operates with a MINO structure and when faced with a disruption that is detected as anomaly it switches to a MAXO organizational structure. The results of the adaptive enterprise are promising
because they show the low latency property of a MINO and the high resilience attribute of a MAXO. This characteristic of the adaptive enterprise is its added value when compared to the fixed structure enterprises.

6.2 Future Work

This work was a beginning for quantifying resilience and influencing it through design parameters, thus, this idea could be extended to different engineering fields such as electrical and computer engineering. For example, adding wave guides or data buses may improve the resilience of those systems to certain noise sources.

In the adaptive enterprise more investigation is needed to understand the tradeoff between complexity and the gained added value in resilience: an adaptive enterprise provides added value but certainly it is considerably more complex. Therefore, the question is that how much complexity is the added value can justify.

Again, in adaptive enterprises the anomaly detection subsystem plays an important role. This mechanisms can detect abnormal behavior of the network through its indicators. Therefore, the detection system’s performance metrics such as precision, false positive rate, and recall are critical in the holistic performance of the adaptive enterprise. As a result, the topic of anomaly detection itself is an extremely important one and it has a broad range of questions to answer. For example, what are the suitable indicators? How the indicators should be evaluated and conditioned?

Currently, the enterprise design concepts and theories have been the subject to many research efforts in the past 40 years, thus, they are mature enough. However, implementing tools and algorithms that automate these computations and rule checking
processes will really help to put the enterprise design into practice. For example, a computer aided design environment that gets an enterprise design as its input and returns the lattice set (and the truncated set) could be used to judge the resilience attributes of different structural designs.

The enterprise architecture design and evaluation is another area for future research. Right now, these two views are designed and implemented separately, but in real world they interact heavily (i.e., impact each other), and thus they must be integrated. In addition, available modeling languages such as UML and SysML are weak on physical notations. For example, the modeling of human and computational assets are provided only at a minimal level.

In addition, the process of synthesizing the executable models of the enterprise architecture and the evaluation phase is a promising area of research. The formal verification methods can be applied to evaluate the executable models and, respectively, the corresponding architectural component(s). In this dissertation, the Computational Tree Logic (CTL) was used to search for the deadlocks of the system. However, CTL is an extremely powerful method and could be used in various functions.

From the theoretical point of view, the mathematical relationship of inter-element collaboration parameters and the resilience attributes could be derived. The formulation could be used later to capture the trends of change caused by parameter variations in a rigorous manner. These trends could be later used to find better resilient structures. In addition, the results could be generalized for various applications.
APPENDIX: ARCHITECTURE DESIGN AND EVALUATION

Architectures play a critical role in the design, evaluation, and analysis of complex systems (e.g., System of Systems). Consequently, the first phase of any system’s development is the architecture design. The second phase, after the design of the architecture, is the evaluation and analysis phase. Based on the development design paradigm used, e.g., spiral method, these two phases may repeat for some iterations. However, adherence to the original requirements requires a rigorous methodology for both design and evaluation. The design phase usually relies on modeling languages such as the Unified Modeling Language (UML), the System Modeling Language (SysML), and many others. The evaluation phase is mainly based on executable models based on Petri Nets and their analysis capabilities such as Computational Tree Logic (CTL) for State Space Analysis (SSA). The SSA could be used to investigate the properties of the design such as potential deadlocks, reachability, and so forth. As a result of these analyses, the architecture could be further enhanced.

This appendix presents the architecture design process used in this work. In addition, the evaluation method with which the architecture is evaluated is elaborated.

Architecting Process

The architecture process consists of three stages: 1) Architecture Design, 2) Executable Model Construction, and 3) Evaluation of the Architecture. At the architecture
design stage, the architecture of the system is created and described usually with an architecture description language such as UML or use the IDEF family of languages. In the following stage, executable model construction, the dynamic model of the system is derived from the architectural artifacts. In the third stage, the executable model is used by simulating the dynamic behavior of the system to evaluate the architecture. There are many types of analyses that can be performed such as searching for the Home State, looking for deadlocks, and constructing and then querying the Reachability Graphs to identify all possible states. In addition, there are many types of analyses possible for performance and effectiveness evaluations. However, in this work, the focus is on evaluating the adherence of the architecture to the original requirements and more specifically, liveliness or absence of undesired deadlocks that can cause obstruction in the business process performed by the system.

Maintaining the consistency of a complex design, especially in a system of systems, is an extremely cumbersome and often endless task. The result is the occurrence of situations in which the required inputs (e.g., data, materials, resources, etc.) are not available in a timely manner (or even at all) at a particular step in the process. This error may result in a deadlock that disables the desired behavior of the system either partially or completely. In addition, a conflict condition may exist that disables parts of a system’s reachability graph that in turn may suppress some, but not all, of the desired behavioral aspects of the system.
The traceability feature of the conversion method of constructing the executable model is used to locate the corresponding points of failure in the architectural artifacts for each of the deadlocks. Therefore, the traceability feature of the architecting process provides guidance about the type and location of errors in the design. The decision on how to modify the architecture design must go back to the architecture design experts. The process of eliminating the deadlocks and reachability problems continues until the desired behavior of the system is not hindered by either deadlocks or reachability problems. The architecture development process is articulated by Levis and Wagenhals [7]. Their methodology (see Figure 53) is a sequential iterative model that requires multiple iterations before producing the final architectural artifacts. This process is illustrated in this appendix through the illustrative example of the Federal Aviation Administration’s Enterprise Architecture.

**Architecture Design**

A system is composed of two main units: functional or logical and physical or implementation units. As a result, a system’s architecture is parsed into the Functional
Architecture (FA) and Physical Architecture (PA) viewpoints, as depicted in Figure 54. Both viewpoints are required for performance evaluation, but for evaluating the logical behavior of the architecture the functional architecture viewpoint is sufficient. However, in this work both FA and PA are required, and thus they are elaborated in the remaining of this appendix.

One salient design decision is where the architect should put the boundary of the architecture. There is no well-articulated answer available for this question but, as a general rule, all aspects that are subject to design choices must be included. Entities that may interact with the system whose architecture is being designed, but are not subject to design decisions, must be located outside the boundary.
A second key decision is the method used to decompose the system. The strategy chosen is top-down and breadth-first decomposition. The top-down method starts from the main concept and then parses it into tractable components. The breadth-first approach requires a complete decomposition at each breadth-level before moving to the lower depth-level. The reason for adopting this strategy is that it produces a balanced decomposition, both in depth and breadth. Using this approach, one starts from the top-level concept and at each breadth-level carries out a decomposition. Then, if it is necessary, one can proceed to the lower breadth level and so forth until all the pertinent aspects of the system are exposed. The design methodology that is adopted in this work is the object oriented architecture design process introduced by Wagenhals et al. [10]. In addition, the architecture is expressed using the Unified Modeling Language (UML) formalism. The revised process is provided next.

**Step 1: Develop Use Cases**

Use Cases capture the interactions between the system and external actors. They are derived from the operational concept narrative. It is very critical to capture all the key scenarios that describe the behavior of the proposed system in response to its interactions with the external entities. Extensions of the Use Cases are used to identify the exceptional cases which may cause the system to deviate from the main success scenario. The Cockburn’s template [74] was used for the Use Cases’ narrative and UML for the Use Case diagrams.

The next three steps are carried concurrently. First, a set of potential classes is defined. These classes represent the actors (i.e., the external entities) and the functional components of the proposed system. Then behavioral diagrams are designed: sequence and
activity diagrams. While activity diagrams can be created without class definitions, sequence diagrams cannot. The sequence diagrams and the activity diagrams specify then the operations and the attributes of the classes. This is an iterative process; when the activities are mapped as operations to classes and swimlanes can be added to the activity diagram.

**Step 2: Construct Sequence Diagrams**
Sequence diagrams are one type of UML’s behavioral diagrams. They show the messages passed between the system’s classes or objects over a time line. They are derived from the Use Cases by considering the causality rules. The interactions indicated in the sequence diagrams describe associations among the classes and are entered in the class diagram. In addition, once the activities in the activity diagram (the operations of the classes) are defined, the sequence diagram is augmented with actions/functions to obtain the Augmented Sequence Diagram (ASD).

**Step 3: Create Activity Diagram**
This step enhances the original design methodology introduced by Wagenhals et al. [10]. In this step, data required for creating the activity model are extracted from ASDs. The swim lanes represent the objects (class instances) and the activities each object performs are extracted from each object’s timeline. The sequence in which activities are performed - the flow of control - is also obtained from the sequence of messages being passed in ASDs.

**Step 4: Final Class Diagram**
The class diagram is the pivotal part of the architecture because it has an abstraction level which can handle a phased evolution of the system. It defines the structure of the
functionality of the proposed system. The attributes and operations of each class are extracted from ASDs and the activity diagram. At the end of the final step, the class diagram is fully annotated, and in addition, the behavioral diagrams (e.g. activity diagrams and ASDs) are ready from the previous step. Therefore, all the essential views required to build the executable model of the architecture are available.

**Executable Model Construction**

After meticulous construction of all the essential architectural views, concordance among them is checked. The concordance check includes six rubrics: 1) class name (or object names) must match the swim lanes in the activity diagram. 2) Activity names in the activity diagram must match a corresponding operation in the class diagram. 3) Attributes of the clauses in the rules associated with the execution of activities/operations must appear as attributes in the corresponding class. 4) An association between two classes represents arcs across the respective swim-lanes of the activity model. 5) All the operations in the class diagram should appear in the activity diagram. 6) Attributes of association classes must appear as attributes in both classes connected by that association. After checking the concordance among the structural and behavioral diagrams from UML that have been used to describe the functional architecture design, the executable model is derived.

The synthesis methodology presented in [10] allows for traceability between the attributes of the architectural artifacts and the pertinent elements of the executable model. This traceability property facilitates the iterative enhancement of the design; it will be demonstrated through the illustrative example presented in the next section.
The methodology requires the application of a style constraint to the class diagram of the system. The style constraint requires that every class be decomposed into a superclass containing only operations and one or more subclasses containing only attributes. The UML composition property is used as shown in Figure 55.

The classes with operations exchange messages (have associations between them). Messages are described by the attributes in a class model. In the executable model messages and data are represented by tokens. Consequently, in the style option, a number of attributes could be aggregated using the product color set of CPN. However, it is important to note that this is only a style option to facilitate the construction of the executable model. In practice, usually a single class is used to reduce the model’s size and complexity. When the style constraint has been applied and changes are incorporated, the atomic parts of the architecture are converted to a bipartite directed graph, which could be rendered in a Petri Net environment (e.g., CPN Tools). A bipartite graph has two types of

![Figure 55 Style constraint: Decomposition of classes](image)
nodes, transitions and places, and they are connected to each other using directed arcs. An arc connects a transition to a place and vice versa, but two places or transitions cannot be linked together. This bipartite graph is constructed in a hierarchical manner. At the top-level (top-page), a transition node is created for each non-association parent class. The subclasses (those with attributes only) are converted to place nodes that work as ports to the sub-page(s). The color sets are defined from the attributes and, as mentioned already, product color sets can be created that combine some of the atomic color sets. For each association class in the diagram, a place is created and if an association is bidirectional then two places are needed (one for each direction). Lastly, appropriate color sets are assigned to all of the places. After creating the top-level graph of the architecture based on the class diagram then the sub-pages are constructed.

Each transition node on the top-page that has more than one operation will have a sub-page that denotes the internal structure of that class. As a result, each operation in the associated class diagram is a transition in the sub-page. Places that are created for association classes as well as classes that represent sub-classes (see Figure 55- those with no operations) all reappear in the sub-page again. These places become input or output port places of the sub-graph. At this point, focus is shifted from the class model to the activity model. It provides the information that is needed to define the arcs between the transitions that represent the operations/activities in the CPN. Table 8 provides a summary of the methodology for deriving the executable model.
Architecture Evaluation

The architecting process is an iterative process in which an architecture design is refined through multiple iterations. To achieve rigor, the architecture must be evaluated at each iteration for logical correctness using the constructed executable model. The most primitive form of evaluation is executing the Petri Net model in simulation mode to check that it behaves as desired, e.g., it terminates properly without stalling. Usually, most preliminary designs fail this simple test. The culprit is often the incomplete set of rules that governs the behavior of the Petri Net model. Therefore, a practical evaluation method is needed.

<table>
<thead>
<tr>
<th>Table 8 Method of synthesizing CPN model of an Object Oriented Architecture</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Define color sets using the attributes of all the class entities in class diagram</td>
</tr>
<tr>
<td>2. Create the hierarchical CPN:</td>
</tr>
<tr>
<td>2.1. Create a substitution transition for each class in the class diagram</td>
</tr>
<tr>
<td>2.2. Create a place for each association and part classes. Assign the appropriate color set. If an association is bidirectional, create one place for each direction.</td>
</tr>
<tr>
<td>2.3. Create arcs between the substitution transitions and the places using the class and activity diagrams.</td>
</tr>
<tr>
<td>2.4. Create sub-page for each substitution transition:</td>
</tr>
<tr>
<td>2.4.1. Create a transition for each operation</td>
</tr>
<tr>
<td>2.4.2. Assign the appropriate ports to places</td>
</tr>
<tr>
<td>2.4.3. Create places for any attributes of the class represented by the sub-page</td>
</tr>
<tr>
<td>2.4.4. Create arcs based on the activity diagram</td>
</tr>
<tr>
<td>2.4.5. Add arc inscriptions, guard functions, or code segments derived from the rules associated with each operation</td>
</tr>
<tr>
<td>2.5. Specify initial markings for each place that represents an part class (i.e. linked by a composition relationship)</td>
</tr>
</tbody>
</table>
To this end, model checking methods are used to evaluate properties of the architecture. The method of interest here is Computation Tree Logic (CTL) [67], which is a type of temporal logic. This formal verification method provides the properties of the model’s states. The generic process of querying the design space for certain properties is presented here.

The first step in the analysis is to generate the State Space of the executable model that represents the functional architecture viewpoint [75]. Formal verification tools like Murphi [76], Spin [77], CPNTools [21], and many others are available and most of them will work for this purpose; the development environment used throughout this effort is CPNTools. Calculating the state space of a system is a straightforward procedure. However, there are a few cases that may require tailoring the method of generating the state space to decrease the computation time and/or create a smaller state space [75]. After the calculation is done, enquiries using CTL can be made.

The syntax of CTL is reviewed in [67] and is consistent with the notation that CPNTools uses. A pseudo code template of inquiries by CTL in CPNTools is used. Using the results of the CTL queries, problematic areas of the executable model are identified and then fixed. Moreover, the traceability of the transformation from UML to CPN enables the locating of the cause of failure in the architectural description. Finally, issues in the architecture artifacts are corrected through redesign. This process continues for a number of iterations until the behavior of the architecture meets the requirements set by the operational concept and the Use Cases.
REFERENCES


BIOGRAPHY

Bahram Yousefi holds B.Sc. and M.S. degrees in Electrical and Computer Engineering. He worked as a senior microwave engineer in a major research and development company for more than two years, during which he gained extensive hands on experience in the field of telecommunication. Subsequently, he acquired in-depth knowledge of Systems Engineering during his doctorate education. His areas of interest are Enterprise Architectures’ (EA) design and evaluation, resilience to cyber disruptions, synthesizing the dynamic models of enterprises using multi-modeling approaches, and enterprise-level cyber security.