RAPID ARCHITECTURE ALTERNATIVE MODELING (RAAM): A FRAMEWORK FOR CAPABILITY-BASED ANALYSIS OF SYSTEM OF SYSTEMS ARCHITECTURES

A Thesis
Presented to
The Academic Faculty

by

Joseph V. Iacobucci

In Partial Fulfillment
of the Requirements for the Degree
Doctor of Philosophy in the
School of Aerospace Engineering

Georgia Institute of Technology
May 2012

RAPID ARCHITECTURE ALTERNATIVE MODELING (RAAM): A FRAMEWORK FOR CAPABILITY-BASED ANALYSIS OF SYSTEM OF SYSTEMS ARCHITECTURES

Approved by:

Professor Dimitri Mavris, Advisor
School of Aerospace Engineering
Georgia Institute of Technology

Professor Daniel P. Schrage
School of Aerospace Engineering
Georgia Institute of Technology

Professor Brian J. German
School of Aerospace Engineering
Georgia Institute of Technology

Mrs. Kelly Cooper
Office of Naval Research
US Navy

Dr. Russell Peak
Manufacturing Research Center
Georgia Institute of Technology

Date Approved: April 2^{nd} 2012

To my loving wife,

Sara DaVia Iacobucci,

the one who encouraged me throughout this process.

ACKNOWLEDGEMENTS

No campaign plan survives first contact with the enemy

[Simplified English translation]
Generalfeldmarschall Helmuth Graf von Moltke

First I would like to thank my advisor, Dr. Dimitri Mavris. He has provided invaluable support in preparing this dissertation. At first, as a newbie, I did not understand his methods, but over the course of this journey I have grown to see the wisdom in them. I would like to thank my distinguished committee members Dr. Schrage, Dr. German, Dr. Peak, and Mrs. Cooper for their time and patience. The valuable comments and critique of my work were used to make it stronger. This manuscript would not have been possible with out financial assistance from ONR.

Within the Aerospace Systems Design Laboratory, I have many people that I would like to thank. First, I would like to thank the Architect team: Kelly Griendling, Charles Domerçant, Burak Bağdatlı, Annie Jones, and David Warshawsky. A special thanks to Kelly Griendling for leading the team. Charles Domerçant has taught me more than I think he knows. We formed a close knit team that brought out the strengths in each of the members. We were able to be completely honest about the merits of each others work, without taking criticisms personally. I feel that the level of camaraderie within the team is not something that happens many times in someone's life. I am glad that I was able to be part of it. I would also like to take a variety of other people in the lab. John Salmon was always available when I needed to talk about just about anything. Daniel Cooksey kept everyone dreaming about bitcoins. Curtis Iwata would never fail to rearrange my desk if I was not there when he visited. Damon Rousis, Reza Rezvani, and Woongje Sung were all great office mates. I have hope for the future (Jon Sands and Metin Ozcan) and hope for the past (SPAM aka

William Engler) in their quests to graduate. Everyone has helped me develop my ideas further.

I'd like to thank Andrew Kerr, Andrew Howard, Adam Knez, and Todd Hoffmann. Over the years they have made sure that my weekends were always fun. Jeremy and Lisa Bain have been great friends to Sara and I.

I would like to thank my family. My mom and dad were always there for me to talk to when I needed it. Denisse and Steph kept the whole PhD thing in perspective for me. Sorry about the missed birthdays, but I shouldn't have to miss any for a while.

I would like to thank billy and all of the woodland creatures. John Moses Browning's prolific inventing mind was an inspiration. This thesis would not have been possible without John McCarthy and his discovery of Lisp.

I need to thank my pets, Koda (the cat) and Luna (the dog). Koda was always there for me and kept me going for the majority of the time I was working on my PhD. He knows that 4am is a great time to take a break. The new addition, Luna, has been my thesis writing buddy. She can't understand why I am so boring, but sticks with me anyway.

Finally, I would like to thank my wife, Sara. I hope the sacrifice of putting your life on hold while I worked on this will pay off. Thank you for taking a gamble on me. When this seemed impossible, you were always there to provide inspiration.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

Listings

SUMMARY

Essentially, all models are wrong, but some are useful.

George E. P. Box[14]

The current national security environment and fiscal tightening make it necessary for the Department of Defense to transition away from a threat based acquisition mindset towards a capability based approach to acquire portfolios of systems. This requires that groups of interdependent systems must regularly interact and work together as systems of systems to deliver desired capabilities. Technological advances, especially in the areas of electronics, computing, and communications also means that these systems of systems are tightly integrated and more complex to acquire, operate, and manage. In response to this, the Department of Defense has turned to system architecting principles along with capability based analysis. However, because of the diversity of the systems, technologies, and organizations involved in creating a system of systems, the design space of architecture alternatives is discrete and highly non-linear. The design space is also very large due to the hundreds of systems that can be used, the numerous variations in the way systems can be employed and operated, and also the thousands of tasks that are often required to fulfill a capability. This makes it very difficult to fully explore the design space. As a result, capability based analysis of system of systems architectures often only considers a small number of alternatives. This places a severe limitation on the development of capabilities that are necessary to address the needs of the war fighter.

The research objective for this manuscript is to develop a Rapid Architecture Alternative Modeling (RAAM) methodology to enable traceable Pre-Milestone A decision making during the conceptual phase of design of a system of systems. Rather than following current trends that place an emphasis on adding more analysis which tends to increase the complexity of the decision making problem, RAAM improves on

current methods by reducing both runtime and model creation complexity. RAAM draws upon principles from computer science, system architecting, and domain specific languages to enable the automatic generation and evaluation of architecture alternatives. For example, both mission dependent and mission independent metrics are considered. Mission dependent metrics are determined by the performance of systems accomplishing a task, such as Probability of Success. In contrast, mission independent metrics, such as acquisition cost, are solely determined and influenced by the other systems in the portfolio. RAAM also leverages advances in parallel computing to significantly reduce runtime by defining executable models that are readily amendable to parallelization. This allows the use of cloud computing infrastructures such as Amazon's Elastic Compute Cloud and the PASTEC cluster operated by the Georgia Institute of Technology Research Institute (GTRI). Also, the amount of data that can be generated when fully exploring the design space can quickly exceed the typical capacity of computational resources at the analyst's disposal. To counter this, specific algorithms and techniques are employed. Streaming algorithms and recursive architecture alternative evaluation algorithms are used that reduce computer memory requirements. Lastly, a domain specific language is created to provide a reduction in the computational time of executing the system of systems models. A domain specific language is a small, usually declarative language that offers expressive power focused on a particular problem domain by establishing an effective means to communicate the semantics from the RAAM framework. These techniques make it possible to include diverse multi-metric models within the RAAM framework in addition to system and operational level trades.

A canonical example was used to explore the uses of the methodology. The canonical example contains all of the features of a full system of systems architecture analysis study but uses fewer tasks and systems. Using RAAM with the canonical example it was possible to consider both system and operational level trades in the same analysis.

Once the methodology had been tested with the canonical example, a Suppression of Enemy Air Defenses (SEAD) capability model was developed. Due to the sensitive nature of analyses on that subject, notional data was developed. The notional data has similar trends and properties to realistic Suppression of Enemy Air Defenses data. RAAM was shown to be traceable and provided a mechanism for a unified treatment of a variety of metrics. The SEAD capability model demonstrated lower computer runtimes and reduced model creation complexity as compared to methods currently in use. To determine the usefulness of the implementation of the methodology on current computing hardware, RAAM was tested with system of system architecture studies of different sizes. This was necessary since system of systems may be called upon to accomplish thousands of tasks. It has been clearly demonstrated that RAAM is able to enumerate and evaluate the types of large, complex design spaces usually encountered in capability based design, oftentimes providing the ability to efficiently search the entire decision space. The core algorithms for generation and evaluation of alternatives scale linearly with expected problem sizes. The SEAD capability model outputs prompted the discovery a new issue, the data storage and manipulation requirements for an analysis. Two strategies were developed to counter large data sizes, the use of portfolio views and top 'n' analysis. This proved the usefulness of the RAAM framework and methodology during Pre-Milestone A capability based analysis.

CHAPTER I

INTRODUCTION

> Any intelligent fool can make things bigger and more complex ... It takes a touch of genius – and a lot of courage – to move in the opposite direction.
>
> Albert Einstein[1]

Andrew P. Sage discusses in 1982 [125] how large models and optimization efforts are expensive and difficult to understand and interpret. The issues that he describes not only still exist thirty years later, but have become more prominent with technological developments and design needs. The research detailed with this manuscript focuses on the conceptual phase of design. A well known motivation for improving the conceptual phase of design is that the early decisions have a disproportionately large impact on program cost and schedule. Military system procurement has witnessed an increase in complexity over the past years which also has increased program cost and stretched program schedules. To understand the required trade-offs the system must be considered holistically with the different systems that it interacts with. The collection of interacting systems are known as systems of systems.

The motivations and background come from two main sources, architecture related and Department of Defense related. The knowledge about architectures and their use in the DoD provides context for RAAM. Unique characteristics and challenges in designing military system of systems provide the backdrop of the RAAM research effort.

[1] Commonly attributed to Albert Einstein

1.1 Department of Defense Related Motivations

The Quadrennial Defense Review promotes reforming how the Department of Defense (DoD) does business. Related to this research it recommends reforming how we buy systems. The conventional acquisition process is noted as too long and cumbersome to fit the needs of the Department of Defense. Maintaining disciplines such as system engineering approaches is mentioned as a potential area for improvement. There will be hard choices in the future of our capability needs that will require practical and efficient procurement processes. The Quadrennial Defense Review 2010 notes that, "we must demand cost, schedule, and performance realism in our acquisition process." It also mentions that a comprehensive design review will be required to reduce technical risk. [51]

Major acquisition programs tend to take too long to deliver a product to the warfighter. An example of the problems affecting defense acquisition, during World War I and the early Cold War major systems were delivered in fewer than six years. The major systems included the Manhattan Project, the Defense Support Program, the intercontinental ballistic missile, and the U-2. In the current acquisition environment, one will see major programs requiring an average of ten to twenty years. By improving the analysis of system of systems, it is hoped that the defense industry can reduce the number of years required to deliver new capabilities to the American warfighter.

In an effort to rapidly acquire or modify special purpose weapon system, the Department of Defense recently has used accelerated acquisition models like Big Safari. Big Safari is an Air Force program responsible for the recent MC-12W Liberty Project aircraft for Intelligence, Surveillance, and Reconnaissance (ISR) in Iraq and Afghanistan. Lt. General David A. Deptula, the U.S. Air Force Deputy Chief of Staff for Intelligence, Surveillance, and Reconnaissance, has said during an Aviation Week and Space Technology interview [70],

> Major acquisition reform will be required. We are going to have to shed layers and layers of process and eliminate excessive legislative and bureaucratic oversight, replacing them with judgment and accountability if we are going to achieve real reform.

The fact that a program such as Big Safari has to exist points to a problem. Acquisition has become so unwieldy and cumbersome that the very organization that imposes the rules has created a way to get around the rules. Big Safari is not the only group created to deal with novel systems. A non-exhaustive list includes the Joint Rapid Acquisition Cell, Army Rapid Equipping Force, Air Force Quick Reaction Cell, Rapid Reaction Technology Office, DARPA, and Air Force Battle Labs (closed). Problems during the system development process are often from poor organization and communication of information. Management of project complexity can have a bigger impact than technological concerns of subsystems [91].

Charette begins his article, *What's Wrong With Weapons Acquisitions?* [20], with the following quote:

> Escalating complexity, a shortage of trained workers, and crass politicization mean that most programs to develop new military systems fail to meet expectations.

The research hopes to make a small contribution toward being able to deal with the escalating complexity. Training workers and politics will be outside of the scope. The complexity arises from the connections between the systems. An example from the article is that the Future Combat Systems (FCS) program had a 28 percent chance of success when it was approved. The Department of Defense needs better tools to assess the ability of programs to meet performance, schedule, and cost.

1.1.1 Weapons System Acquisition Reform Act of 2009

The Weapons System Acquisition Reform Act of 2009 was signed into law on May 22^{nd} 2009 as Public Law 111-23. [1] The act is designed to reform how defense acquisition occurs. The act changes how acquisition is organized and a variety of acquisition policies. [95]

The organizational changes include changes to system engineering capabilities, developmental testing, technological maturity assessments, independent cost assessment, and the role of combatant commanders. System engineering capabilities will be improved by requesting that the DoD assess the extent of system engineering capabilities and establishing organizations and people to fix any deficiencies in the system engineering capabilities. Developmental testing has been allowed to atrophy and the bill requests the DoD to remedy any deficiencies in developmental testing and to establish the position of Director of Developmental Test and Evaluation. The bill does not explain how to fix the deficiencies in developmental testing which is an active area of research. Technological maturity assessments are now the responsibility of the Director of Defense Research and Engineering. Independent cost assessment will see more use with the establishment of the position of Director of Independent Cost Assessment. The role of combatant commanders will be expanded to create more influence into the acquisition process to ensure that long term needs are met.[95]

The acquisition policy changes include changes to trade-offs of cost, schedule and performance; the Preliminary Design Review (PDR); Lifecycle Competition; Nunn-McCurdy breaches; organizational conflicts of interest; and acquisition excellence. Trade-offs of cost, schedule, and performance will now include more analysis to determine the impact of requirements change on cost, schedule, and performance. The barriers between the budget, requirements, and acquisition stovepipes should be removed. The PDR will be required and a post-PDR assessment will be done before

Milestone B approval. Lifecycle competition should improve the incentives for keeping costs low. The new measures should encourage competition throughout the life of the program. Cost growth has become almost expected and acceptable in major weapon systems. Any Nunn-McCurdy breaches will require Secretary certification and new Milestone approval using independent cost estimates. Organizational conflicts of interest should be avoided. System engineering contractors will be prohibited from participating in the development or construction of a major weapon system in which they are system engineering contractors. The new policies will require highly skilled and capable acquisition specialists. The act establishes an awards program to reward exemplary employees. [95]

Along with the increased complexity of systems of systems, continued financial pressure has forced the political and military leadership in the United States to be interested in acquisition reform. The Weapon Systems Acquisition Reform Act of 2009 includes organizational and acquisition policy changes including improvements to system engineering capabilities and the process for cost, schedule, and performance trade-offs. Further progress is needed, however, to continue improving the DoD acquisition process. The DoD breaks the acquisition process up into three main phases: pre-systems acquisition, systems acquisition, and sustainment. This dissertation is concerned with Pre-Milestone A activities in the pre-system acquisition phase. [50] These are activities that occur in the conceptual phase of design before technology development occurs. Efforts have been focused on this phase because decisions made at the beginning of the process have been shown to have the greatest influence on performance, cost, and schedule. [28]

1.1.2 Analysis of Alternatives

DoD instruction for the operation of the defense acquisition system explains the important aspects of an analysis of alternatives. The Analysis of Alternatives (AoA)

should, "focus on identification and analysis of alternatives, measures of effectiveness, cost, schedule, concepts of operations, and overall risk." The DoD Instruction recommends that emphasis is placed on innovation and competition during the analysis of alternatives. An AoA is evaluated by the DPA&E (Director, Program Analysis & Evaluation) with the OSD (Office of the Secretary of Defense) and Joint Staff. It is evaluated with a focus on whether or not the AoA illuminated capability advantages and disadvantages, considered joint operational plans, examined sufficient feasible alternatives, discussed key assumptions and variables and sensitivity to changes, calculated costs, assessed technology risk and maturity, assessed alternative way to improve the energy efficiency, and assessed the appropriate system training. [40]

In their Pre-Milestone A/B checklist, the Committee on Pre-Milestone A Systems Engineering mentions the importance of alternative concepts in the decision process. [28] By comparing alternative concepts, the decision maker is made fully aware of the strengths and weaknesses of a course of action. Without analyzing alternatives, the decision process is predetermined and of little value.

1.1.3 Architecture Framework Deficiencies

Recently the new version of DoDAF (version 2.0 [48]) was released. DoDAF is the Department of Defense's Architecture Framework which is used to document and represent military architectures. The Architecture Frameworks Working Group from the National Defense Industrial Association recommended changes and additions. They identified eight different systems engineering needs, a standard architecture modeling methodology, greater definition and standardization of architecture elements, executable/simulatable architecture models, composable/decomposable models, standard architecture alternatives analysis method, standard architecture modeling notion and symbology, and the auto-generation of systems engineering artifacts. They conclude that "DoDAF v2 improves on satisfaction of SE [Systems Engineering] needs,

but systems engineers need greater definition and standardization of semantics and methods that are important to them."[129] The solutions that they propose are based on the current UML, SysML, and UPDM stack for modeling military architectures. For more information on these architecture frameworks see Chapter 4. They are concerned with the creation of detailed architecture artifacts and models that can be shared at multiple levels of scope. It is the authors assertion that the conceptual development of system of system architectures can use many of the same recommendations. The recommendations enable a traceable analysis of alternatives that provides an architecture alternative that can be developed further in the next steps of design.

1.1.4 Need for better ways to do early SE

In 2008 the National Research Council formed a committee on Pre-Milestone A systems engineering to look at the past and future benefits of system engineering for Air Force acquisition. [28] The potential role of systems engineering in the defense acquisition life cycle to address causes of program failure in the early phases of the program were analyzed in the report. As noted in the report, "Recent years have seen a serious erosion in the ability of U.S. forces to field new weapon systems quickly in response to changing threats, as well as a large increase in the cost of these weapon systems." The report finds it puzzling that as we have increased in technology and experience that we are worse at developing systems than thirty years ago by two to three times. Figure 1 shows the contrast between historical and existing systems.

In the 1990s the development planning function within the Air Force Systems Command was removed. The 2008 report also states, "Currently, few formal SE [System Engineering] processes are applied to Air Force development programs before the Milestone A review." [28] A main finding that motivated the creation of RAAM was that:

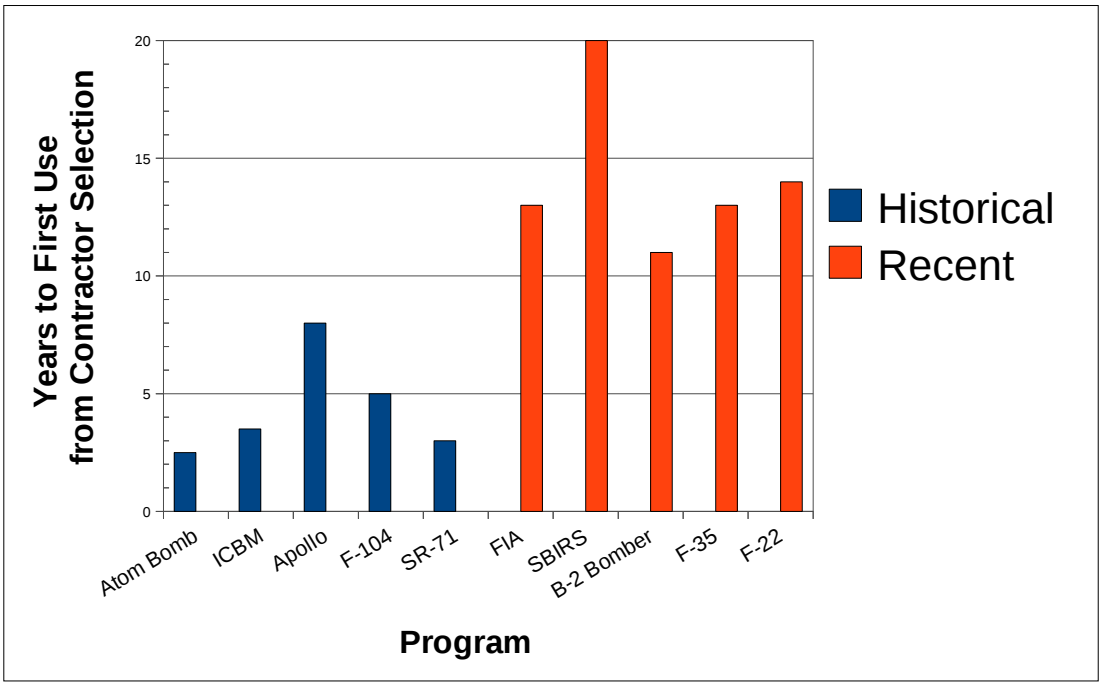


Figure 1: Program Development Times for Major Programs Adapted from tabular data in [28]

> Attention to a few critical systems engineering processes and functions particularly during preparation for Milestones A and B is essential to ensuring that Air Force acquisition programs deliver products on time and on budget.

The long development cycles as shown in Figure 1 can place the U.S. warfighter at a distinct disadvantage. The rapid advancement of adversary technology needs to be considered in contrast to the long cycle times of system development. The increase in cycle time, however, is not unavoidable. The report also comments that like military systems, the complexity of private sector systems has increased over the years. Unlike the military counterpart, however, the private sector systems cycle time has decreased. [28]

The Defense Science Board [39] identified six causes of increased cycle times for defense acquisition. They are:

1. Overly ambitious initial requirements often exacerbated by requirements growth

during development

2. Over optimistic cost and schedule estimates
3. Immature technology
4. Lack of flexibility to adjust requirements when problems arise
5. Funding instability
6. Lack of consideration of affordability, producibility, or sustainability during early development.

The Defense Science Board noted many challenges to overcome within the defense enterprise. They include the decline in technical and program management expertise, budget pressure, bureaucracy and process replacing executive leadership, cultures that favor familiar approaches, and quality of workforce issues. The committee believes that high quality Pre-Milestone A system engineering almost certainly contribute to positive outcomes for a project.

John Griffin, a member of the Defense Science Board Committee on Pre-Milestone A Systems Engineering [39], identified thirteen important steps in the acquisition process. The steps are:

1. Defense Strategy
2. Joint Warfighting
3. Capabilities $\rightarrow$ Attributes Measures of Effectiveness
4. Gaps
5. Conceptual Studies
6. Concept/Systems

7. System Requirements
8. System Design
9. Build
10. Integrate
11. Test/Verify/Validate
12. Production
13. Operation

The idea is that there is a common thread starting from Defense Strategy (strategy) to Concept/System (Concept) to Operation (Initial Operating Capability). If the thread is broken, the result is often cost and schedule overruns or performance degradation.

A method called Workload Task Analysis (WLTA) has been created to guide training planning for new weapon systems. [61] WLTA is designed to address shortcomings in the early system engineering phase as is applicable to training. The method uses a missions-functions-tasks hierarchy which is similar to the task hierarchy used in RAAM.

70% is an often quoted number for the amount of life cycle costs that are accounted for with decisions that occur during system concept studies. Only 85% and 95% of life cycle costs are accounted for at the end of system design definition and full scale development respectively. [6] With such a large percentage of life cycle costs occurring during the conceptual phase it is critical to make the right decisions early.

1.1.5 Problems with the Acquisition Process

1.1.5.1 Lead Systems Integrators

A Lead System Integrator is a company that takes on the role of acquiring and integrating a collection of systems that may not be created by the company. Traditionally,

the government is responsible for system selection in a system of systems. The author believes that the Lead System Integrator concept has led to the failure of at least two System of Systems programs. The Coast Guard's Deep Water program and the Army's Future Combat System (FCS) both have failed. The Deep Water program was reorganized to be under the Government's supervision. Another example of the failure of a lead system integrator is found in [28]. The total system performance responsibility (TSPR) of the Space Based Infrared Systems (SBIRS) program was delegated to the prime contractor. The government was reduced to asking the prime contractor for information about program execution and decisions. In the same document, the committee was not completely sold on the Lead System Integrator (LSI) concept but was hopeful. The Future Combat Systems program failed in part due to the disregard of good system development concepts detailed in the System Engineering checklist from [28].

The research detailed in subsequent chapters is focused on conceptual design and Pre-Milestone A activities because many of the problems with system of systems architectures have been a result of improper conceptualization. The Committee on Pre-Milestone A Systems Engineering [28] declared, "The government's focus should be on developing requirements, on Pre-Milestone A activities, and on monitoring and assessing the contractor's performance during Pre-Milestone A and throughout programs." These statements are partially in response to the issues that came up with Lead System Integrators.

1.1.5.2 Novel Systems

This research will also address issues with trying to analyze novel systems. The following definition of a novel system is drawn from [10]. There is an increase in uncertainty and new challenges for the acquisition system when procuring a novel system. Novel systems are different from conventional systems in five dimensions.

The five dimensions are:

- Design
- Operational Employment
- Outcomes
- Production Run
- Operational Life

The outcomes, production run, and operational life are all uncertain. The design is necessarily new and the technology is disruptive. The military does not completely understand how to operationally use the systems. There should be an environment that, "fosters new concepts for systems and new concepts of operations."[10]

The research will attempt to provide an analysis framework that helps prove the case for novel systems by allowing for an apples to apples comparison of novel systems and their operation to be compared in the same framework as existing conventional systems.

1.2 Transition from Threat Based to Capability Based Planning

The Department of Defense has been transitioning from a threat based planning process to a capability based planning to address current and perceived future conflicts. The National Academies writes:

> In the past 15 years, the Department of Defense (DOD) has faced a constant stream of new challenges. Now, rather than being prepared to face a major Soviet threat and a few major regional contingencies (e.g., North Korea) in conventional warfare scenarios, the United States must be prepared both to deal with a larger number of more diverse threats with

> varied attributes and to do so in circumstances involving complex and uncertain risks. [29]

With the increase in complexity and uncertainty of threats to the nation, the Department of Defense is interested in ways to use system of systems to counter the variety of future threats with a diverse set of capabilities.

Military systems such as command and control have been known to be complex and require special attention. Long range requirements may be elusive and may be best satisfied with an evolutionary approach. [65] Greene and Mendoza document the transition from a stovepipe development to an interoperable system of systems for command and control. [73]

1.3 Concept Generation and Selection

To properly manage the increased complexity of systems of systems, military systems of systems architectures are becoming the norm. The system of systems is described by its architecture, which the Institute of Electrical and Electronics Engineers (IEEE) defines as, "*The fundamental organization of a system embodied in its components, their relationships to each other, and to the environment, and the principles guiding its design and evolution*" [78]. Also, Taylor, Medvidovic, and Dashofy define an architecture for software systems to be, "*the set of principal design decisions made about the system*" [136]. The common thread between these definitions is that the analysis of a system of systems architecture can involve numerous types of different trades when making architectural design decisions. A large number of alternatives can be the result of military system of systems architectures having thousands of tasks. Often, the analyst is interested in which system does which task and they create a model to explore the different options. This leads to large design spaces that suffer from the combinatorial explosion of alternatives. Assuming that only two systems are vying for each of one thousand tasks then 2^{1000}, or a number with three hundred

and two digits, different architectures are available to analyze. For the remainder of this manuscript, the aspects of an architecture related to the ways and means, described by the task hierarchy and the system to task assignments, will be included in the architecture alternative space. In this way, both the organizational structure (task hierarchy) and the system to task mappings are able to be modified, further increasing the number of available options that must be analyzed. For example, more alternatives are created when different task hierarchies are modeled.

Fortunately, real world analyses are not as bleak. Only millions or billions of alternatives must be analyzed, because compatibility constraints can significantly reduce the number of possible combinations. This takes the 302 digit number (2^{1000} = 107150860718626732094842504906000181056140481170553360744375038837035105112493612249319837881569585812759467291755314682518714528569231404359845775746985748039345677748242309854210746050623711418779541821530464749835819412673987675591655439460770629145711964776865421676604298316526243868372056680693 76) to something more manageable that is only in the billions (such as 2987228160). However, the design space is discrete and the system of systems performance can vary widely depending on the constituent systems. For these reasons, both computational and manpower resources become heavily taxed by this class of design exploration problem.

1.4 Financial Pressure

The current economic climate is forcing a review of the roles and missions of the United States military. With the national debt at 13.5 billion dollars in 2010, or 93.4% of the national GDP, there is less tolerance for failing or inefficient military systems. It is likely that the military budget will be cut in the future in an attempt to reduce the yearly deficit and national debt. The potential reductions in funding require that system acquisitions have better justification of need. Military decision

makers will need more certainty that the right system or collection of systems is being acquired.

1.5 Complexity

One of the issues facing the acquisition and design of system of systems is increased complexity. The dictionary definition of complex is, "a whole made up of complicated or interrelated parts" [30]. The definition is not very useful as it depends on the word complicated. Complicated is defined as either, "consisting of parts intricately combined" or "difficult to analyze, understand, or explain".[31] Neither of the definitions really distinguish something complex from a system. Complicated is also problematic because it is solely descriptive. Using the definition for complicated, we can not know if something is complicated until we attempt to analyze, understand, or explain it.

Bar-Yam describes complex systems as having six characteristics.[7] The six characteristics are

- Elements
- Interactions
- Formation/Operation
- Diversity/Variability
- Environment
- Activities

Complex systems have components that are interdependent. Subsets of a complex system require more information to characterize than the whole, perhaps more clearly, "the smaller the parts that must be described to describe the behavior of the whole, the larger the complexity of the entire system." [7]

Many definitions of complexity start with how to measure complexity. Gell-Mann's paper, *What is Complexity?* [71], adds to our understanding of complexity. People measure computational complexity with time and space measures. Informational complexity is measured with information content measures. The complexity measures depend on the level of detail used to describe the entity.

Acquisition practitioners are worried about current requirements that may be pushing systems toward overwhelming integration. Acquiring the complex capability as a service is appealing but may lead to headaches later on. A soft start of the program is recommended for IT systems. [122] In addition, cultural differences can impact computational models. Cognitive dimensions may be required to estimate the impact of complexity arising from the human element in the phenomena of interest. [92]

A phenomenon is complex when it emerges from a collection of interacting objects. [83] Emergence is another hard to define word which depends on context and human understanding of a phenomenon. Complex systems are often associated with emergent effects.

Complexity is always determined by your frame of reference. It is often hard to separate the difference between the complexity of the way of describing something and the complexity of the description. The complexity of analyzing a system of systems architecture is reduced by intelligent additions of complexity to the analysis method.

1.6 Computer Advances

Improvements in runtime of analysis come from two main sources; improvements in computing power and improvements in the underlying algorithms. In one study of linear programming problems, computing power improvements led to three orders of magnitude improvement in runtime. Algorithmic improvements made the other three orders of magnitude for a speed up of near one million times. [11] Martin Grötschel

has documented a speedup of 43 million times over 15 years, with approximately three orders of magnitude due to improvements in computing power and a factor 43,000 due to algorithmic improvements [120]. This clearly demonstrates that there is a great potential for algorithmic improvements.

The paradigm of increasing processor clock speed to improve performance has ceased to work in modern processors. CPU manufacturers have turned to other technologies such as pipelining to improve processor performance. Those technologies are reaching their limits and current increases in computing power are coming from using multiple processors. The advantages of new computing systems are going to come from devising ways to transform our problems into parallelizable problems, or problems that can be run on multiple processors.

Properly structuring your problem can free it from the shackles of serial computation. The current research takes advantage of the recent advances in parallel computing by a judicious application of parallel computing principles that drive algorithmic changes.

1.7 Architecture Related Definitions

The manuscript will cover two main types of definitions:

- Architecture Related
- DoD Related

The architecture related definitions are discussed in this section. DoD related definitions are described in Section 2.1.

Many readers will be familiar with a subset of these areas but will require a refresher of the nomenclature and jargon used in this manuscript. There may be multiple accepted definition of a term. In these instances, the applicable one for the research herein has been clearly denoted as such.

Framework The Merriam-Webster Online Dictionary defines a framework as, "a basic conceptual structure (as of ideas)." [69]

For the purposes of the research the word framework means a conceptual structure of ideas that provides a way to think about a problem.

Methodology The Merriam-Webster Online Dictionary defines a methodology as, "a particular procedure of set of procedures." or "a body of methods, rules, and postulates employed by a discipline." [106]

Methodology is defined by Sage [125] as, "an open set of procedures for problem solving", he continues, "a methodology involves a set of methods, a set of activities, and a set of relations between the methods and the activities.".

For the purposes of the research a methodology is a set of procedures used to solve a specific problem.

1.8 Architectures

1.8.0.3 Model

Dori defines a model as, "an abstraction of a system, aimed at understanding, communicating, explaining, or designing aspects of interest of that system." [58]

The DoD Architecture Framework considers a model to be, "a template for collecting data." [48]

Bouvier, Cohen, and Najam [13] discuss how a model is a reduction and an abstraction of the system that you are attempting to model. The model is simpler than the system itself so that we can solve the problem.

For this work, a model is an abstraction of a system (or system of systems) that provides a way to simulate metrics of interest about the system.

1.8.0.4 System Engineering

The report *Pre-Milestone A and Early Phase System Engineering* considers system engineering to be, "the translation of a user's needs into a definition of a system and its architecture though an iterative process that results in an effective system design". Later it breaks down system engineering into a detailed three part definition based on the Systems Design and Operational Effectiveness 625 Class Notes from the Stevens Institute of Technology. The extended definition is:

1. SE [Systems Engineering] is the translation of a need or deficiency into a system architecture through the application of rigorous methods to the iterative process of functional analysis, allocation, implementation, optimization, test, and evaluation.

2. SE is the incorporation of all technical parameters to ensure compatibility among physical and functional interfaces, and hardware and software interfaces, in a manner that optimizes system definition and design.

3. SE is the integration of performance, manufacturing, reliability, maintainability, supportability, global flexibility, scalability, interoperability, upgradability, and other special capabilities into the overall engineering effort.

NASA defines system engineering as:

> a methodical, disciplined approach for the design, realization, technical management, operations, and retirement of a system. [111]

1.8.0.5 Architectures

Architectures and architecting are fundamental to the current discussion. As such,a definition of an architecture is in order.

Maier defines architecting as, "the art and science of designing and building systems". The deliverable is a set of abstracted designs of the system. [121]

The Department of Defense considers a systems architecture to be: "(DOD) Descriptions, including graphics, of systems and interconnections providing for or supporting warfighting functions." An architecture is, "A framework or structure that portrays relationships among all the elements of the subject force, system or activity." [84]

Dori defines architecture as, "The combination of structure and behavior". [58]

IEEE standard definition for architecture is, "The organizational structure of a system or component". [79] A slightly more in depth definition from IEEE (in IEEE 1471) for software intensive systems defines an architecture as, "the fundamental organization of a system embodied in its components, their relationships to each other and to the environment and the principles guiding its design and evolution" [78]. The IEEE 1471 document has been updated to an ISO/IEC/IEEE standard, ISO/IEC/IEEE 42010:2011(E) [80]. The definition of architecture in the new document is, "fundamental concepts of properties of a system in its environment embodied in its elements, relationships, and in the principles of its design and evolution."

The ISO/IEC/IEEE 42010:2011 standard is concerned with architecture frameworks and architecture description languages. A distillation of the ISO/IEC/IEEE 42010:2011 standard is contained in the ISO/IEC/IEEE 42010 FAQ. [98] They share five insights from the document:

- The architecture of a system of interest is what is considered fundamental about that system in the context of its environment
- An architecture description documents an architecture
- architecture descriptions should demonstrate how an architecture meets the needs of the system's diverse stakeholders
- The architecture concerns of the diverse stakeholders can be addressed by an

architecture description constructed with multiple architecture views of the system, where each view covers an identified set of those concerns

- The rules for well-formedness, completeness and analyzability of each architecture view should be explicit to readers of an architecture description via an architecture viewpoint

- These ideas can be captured via a conceptual model or, metamodel, establishing the key concepts and terms for talking about architectures and architecture descriptions

The standard recognizes that an architecture is a conception of a system. An architecture is distinct from its description. The description is a concrete object, but the architecture is something that exists in the human mind.

In *Pre-Milestone A System Engineering* [28] architecture is defined as, "the partitioning of the system into separately definable and procurable parts, the structuring of interfaces between the system and the outside world, and the structuring of interfaces (physical, function, and data among the segments." Later in the document, an architecture is defined as, "multidimensional representations or combinations of 'what, how, where, who, when, and why'."

Mavris and Dickerson [43] define architecture as, "The fundamental organization of a system, embodied in its components, their relationships to each other and the environment, the principles governing its design and evolution, its purpose, and its attractiveness."

For this dissertation, we will consider the architecture to be the complete description of a system of systems including the tasks, systems, and connections between the tasks and systems. This allows for further system design and acquisition.

1.8.0.6 System of Systems

Before discussing the definition of a system of systems we need to determine the characteristics of a singular system.

System Dieter defines a system as, "the entire combination of hardware, information, and people necessary to accomplish some specified mission." [44]

The IEEE in IEEE Std 610.12-1990 defines a system as, "A collection of components organized to accomplish a specific function or set of functions." [79] Their focus was for software engineering but the definition is useful in the general case.

The International Council on Systems Engineering (INCOSE) definition for a system is: "a combination of interacting elements organized to achieve one more stated purposes" and "an integrated set of elements, subsystems, or assemblies that accomplish a defined objective". Their definition clarifies that the elements can be products, processes, people, information, techniques, facilities, services and other support elements. [135]

Maier and Rechtin state: "System: a set of different elements so connected or related as to perform a unique function not performable by the elements alone."[121]

Chen and Stroup define a system as, "an ensemble of interacting parts, the sum of which exhibits behavior not localized in its constituent parts." [22]

The Department of Defense defines a system as, "a functionally, physically, and/or behaviorally related group of regularly interacting or interdependent elements; that group of elements forming a unified whole." [84] The same definition appears in the DoDAF v2.0 Manager's Guide [48].

Dov Dori explains that, "All systems exhibit a common feature: they carry out some *function*." He continues, "A system consists of a collection of related objects, represented by the system's structure that interact with each other via processes in a coordinated way, accounting for the system's behavior." He also comments on how,

"system is relative to the domain of discourse or task at hand." A key point that Dori highlights about systems is the subjective nature of the categorization of an object into the class system. A system's definition is dependent on a function which is subjective. [58] The proposal will revisit the relative nature of the concept of a system when the issues with scope are discussed.

NASA defines a system as, " a construct or collection of different elements that together produce results not obtainable by the elements alone." [111]

The definitions include the requirement to have a mission, specific function, or a purpose. The definitions all mention a variety of smaller parts, elements, or components that come together for the purpose. A system of systems is an extension on the definition of a system.

System of Systems Even though there have been attempts at defining a system of systems, different fields have their own definitions. For the purposes of this dissertation, the following definitions and concepts will be used.

Maier defined a system of systems [99] as having two characteristics of its component systems, both "valid purposes in their own right and continue to operate to fulfill these purposes if disassembled from the overall system" and "the component systems are managed (at least in part) for their own purposes rather than the purposes of the whole". He distills this into the "operational and managerial independence of the system components". The component systems must both be capable of and actually operate independently.

Maier further proposed four architectural design heuristics for system of systems based on previous work. They are shown in bold in the current paragraph. **Stable Intermediate Forms**, where at each stage of the development both the component systems and the system of systems should be usable. **Policy Triage**, where there must be a balance between over and under design of the component systems.

Leverage at the Interfaces, where Maier states, "The greatest leverage in system architecting is at the interfaces. The greatest dangers are also at the interfaces." **Ensuring Cooperation**, where Maier states, "If a system requires voluntary collaboration, the mechanism and incentives for that collaboration must be designed in".

These heuristics will be useful when designing a system of systems architecture in the conceptual phase. They will also serve to determine if the studied architecture is a system of systems.

Sage and Cuppan [126] build on Maier's definition of a system of systems. The collection of systems is "often formed from a variety of component systems: newly engineered from the "*ground up*" custom systems, potentially tailored existing Commercial-Off-The-Shelf[COTS] systems, and existing or legacy systems"(emphasis in original). He mentions that formally, anything can be a system of systems. They further summarize Maier's paper into five characteristics:

- Operational Independence of the Individual Systems
- Managerial Independence of the Systems
- Geographic Distribution
- Emergent Behavior
- Evolutionary Development.

Despotou, Alexander, and Hall-May [41] discuss different definitions of system of systems. They identify a list of eight characteristics that define a system of systems:

- Autonomy
- Collaboration
- Complexity

- Heterogeneity
- Adaptability
- Emergent Behavior
- Dependability
- Distributed

They conclude that "a System of Systems is an organised[sic] complex unity assembled from distributed autonomous systems (capable of independent provision of services) collaborating to achieve an overall system purpose."

The Department of Defense has begun to synchronize its definition of a system of systems. Previously the Defense *Acquisition Handbook* [50] and the *Joint Capability and Integration System Manual.* [25] In the *Systems Engineering Guide for System of Systems* [116], the *Defense Acquisition Guidebook*, and the *JCIDS Manual* it is defined as:

> a set or arrangement of systems that results when independent and useful systems are integrated into a larger system that delivers unique capabilities.

The Committee on Pre-Milestone A System Engineering defined a system of systems as, "groups of systems, each of which individually provides its own mission capability, that can be operated collectively to achieve an independent, and usually larger, common mission capability." [28]

The previous definitions show general themes that define a system of systems. A component system within a system of systems has a certain degree of independence from the system of systems. The components systems are not defined by being included in a system of systems, but do work together for the purpose of the system of systems and are autonomous. The effects of a system of systems are often non-linear.

CHAPTER II

DOD BACKGROUND

2.1 DoD Related Definitions

To deliver a capability to the warfighter three different processes must work together. [19] These processes are as follows:

- Requirements process
- Acquisition process
- Planning,Programming, Budget, and Execution (PPBE) process

The following definitions will cover all three elements of a successful delivery of a capability to a warfighter.

2.1.0.7 Capability

Within the Department of Defense (DoD), capability is defined as:

> **The ability to achieve a desired effect under specified standards and conditions through combinations of means and ways across the doctrine, organization, training, materiel, leadership and education, personnel, and facilities (DOTMLPF) to perform a set of tasks to execute a specified course of action.** It is defined by an operational user and expressed in broad operational terms in the format of an initial capabilities document or a joint DOTMLPF change recommendation. In the case of materiel proposals/documents, the definition will progressively evolve to DOTMLPF performance attributes identified

> in the capability development document and the capability production document. [19] [emphasis added to the core definition]

It is important to note that the definition includes what is trying to be accomplished (the desired effect), how difficult the capability is (standards and conditions), what will be used (means), how the system will be used (ways), and a plan for their use (the course of action).

There are many ways to accomplish a capability. For example, for the capability of global strike can be accomplished with submarine launched missiles, precision weapons delivered by bombers, sabotage missions conducted by Special Forces, or other combinations of systems. Each of the options requires a system of systems architecture made up of a portfolio of systems to accomplish the capability.

Capabilities are important because of the capabilities-based assessment (CBA) process. The CBA process is mandated the DoD. By using capabilities, the DoD design community believes that better systems will be acquired. The Joint Capabilities Integration and Development System (JCIDS) defines capability based assessment:

> The CBA is the Joint Capabilities Integration and Development System analysis process. It answers several key questions for the validation authority prior to their approval: define the mission; identify capabilities required; determine the attributes/standards of the capabilities; identify gaps; assess operational risk associated with the gaps; prioritize the gaps; identify and assess potential non-materiel solutions; provide recommendations for addressing the gaps. [19]

A capability portfolio is defined as, "A collection of grouped capabilities as defined by JCAs and the associated DOTMLPF programs, initiatives, and activities." [47] A JCA is a Joint Capability Area. JCAs are a standardized set of definitions that cover the complete range of military activities.

Capability portfolio management is defined as:

> The process of integrating, synchronizing, and coordinating Department of Defense capabilities needs with current and planned DOTMLPF investments within a capability portfolio to better inform decision making and optimize defense resources. [47]

The DoD uses capability portfolio management to optimize capability investments across the defense enterprise and to minimize risk.

The intelligence community is pushing for making decisions based on capabilities rather than individual programs [102]. The Army desires unified capability sets rather than program driven planning and acquisition. The warfighter sees inefficiencies and capability gaps. There is a need for "back of the envelope analysis" that is sufficient for beginning architecture development [148].

2.1.0.8 DOTMLPF

DOTMLPF is an acronym commonly used to refer to the multitude of things that you can change to enable or improve capabilities. The seven things are **D**octrine, **O**rganization, **T**raining, **M**ateriel, **L**eadership, **P**ersonnel, and **F**acilities.

The current research will focus on materiel solutions. The decomposition is not orthogonal because different elements combine and the distinctions may be fuzzy. As you introduce new materiel, you often get changes in doctrine, organization, training, leadership, personnel, and facilities. For example, the new materiel of F-22s has influenced a change in the other elements. For this research we will assume that the analysis of materiel changes includes some of the second order effects from changing the other elements.

2.1.0.9 Capability Based Planning

Dori is quick to caution against going straight from the goals or requirements to systems. [58] The same ideas are seen in DoD documents establishing capability

based planning. There should be a process that converts capabilities into a portfolio of systems to procure.

Paul Davis defines capability based planning as, "planning, under uncertainty, to provide capabilities suitable for a wide range of modern-day challenges and circumstances while working within an economic framework that necessitates choice." [36] The concept was discussed in the 2001 Quadrennial Defense Review [46] where the report was designed to shift defense planning from a 'threat based' model to a 'capability based' model. The distinction is focusing on how the adversary will fight, rather than who they are. The capabilities reflect a set of desired effects on adversaries. With the Cold War over, creating American defense related systems specifically to counter the Soviet (which no longer exists) threat is no longer feasible nor desired. In threat based planning you design a system to accomplish a worse case scenario. It is assumed that the worst case scenario is sufficiently stressful that any other require capabilities naturally fall out. Capability based planning focuses on what you actually want to be able to do and makes sure that the systems can accomplish those goals.

Capability Based Planning may change names in the future, but the concept of beginning planning with desired effects rather than beginning planning with how to accomplish the desired effects.

Figure 2 shows the flow from national objectives to a portfolio of systems to procure.

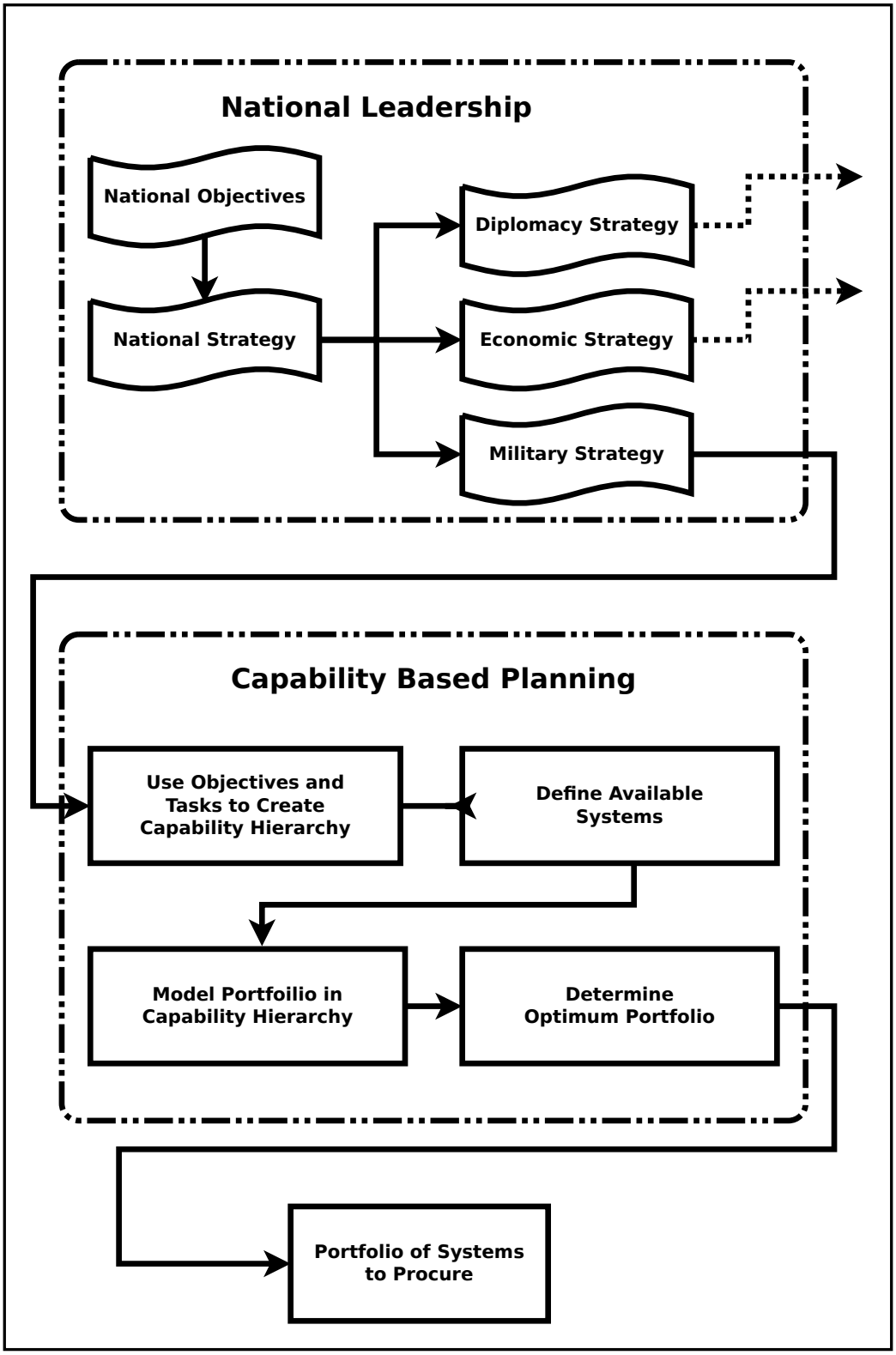


Figure 2: Capability Based Planning Process

2.1.1 Strategic Planning

A main goal of strategic planning is to determine how to invest resources between many objectives. The data, both qualitative subject matter expert generated and quantitative model or empirical based, required for strategic decision making can be overwhelming. The goal is not to optimize for a set of assumptions, but to find a portfolio that has adequate performance while being flexible, adaptive, and robust. [37]

Linear weighted sums are often used for strategic planning. In real life planning, the factors involved and how they are combined is non-linear. Davis and Dreyer [37] believe that aggregation into a single utility without traceability can lead to the analyst injecting their own assumptions and preferences into the decision over a Decisionmaker's. Normalizing scores into a utility from zero to one can remove intuition about the nature of differences. System effects can require the use of nonlinear aggregation rules.

2.1.2 DoD Program Milestones

The Department of Defense utilizes a series of Milestones to discuss what phase a program is in. This is documented in Figure 3. The research is concerned with Pre-Milestone A or conceptual activities.

Acquisition is broken up into two types, big 'A' Acquisition and little 'a' Acquisition. Big 'A' Acquisition integrates requirements, budgeting, and acquisition. Little 'a' acquisition focuses on cost, schedule, and performance. [28]

Detailed standards and integration are not possible Pre-Milestone A due to broadly defined requirements. In addition, the current culture of system specific development needs to change to be more capability based and result from an integrated architecture method. [17]

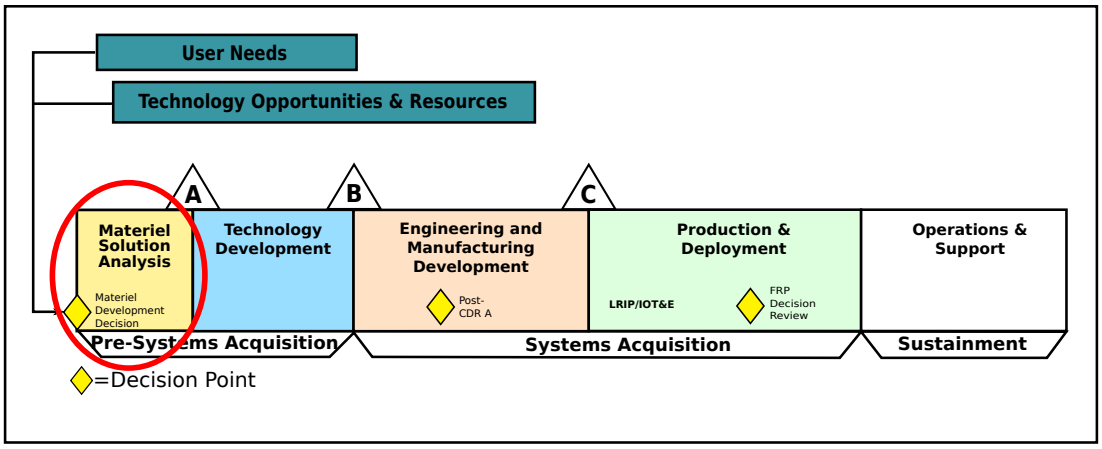


Figure 3: Lifecycle Framework View with Focus on Pre-Milestone A Adapted from [50]

2.2 Department of Defense Regulations and Frameworks

The goal of the following regulations and frameworks is to develop strategy for the United States. The strategy provides guidance as to how to accomplish the national goals. The Department of Defense describes strategy as, "A prudent idea or set of ideas for employing the instruments of national power in a synchronized and integrated fashion to achieve theater, national, and/or multinational objectives." [84] Strategy links ends, ways, and means. Ends are fashioned by the civilian leadership of the nation. Ways are the activities and methods in which to combine systems, organizations, and tactics. Means are the tools used to operationalize the ways. [123]

2.2.1 National Strategy Documents

The United States Government produces a set of documents that specify the high level strategy in diplomatic, economic, and military spheres.

The National Security Strategy of the United States is created by the executive branch for congress every four years. The document outlines the national security concerns and a general set of plans for addressing the concerns. The most recent National Security Strategy document was released on May 26^{th}, 2010. [113]

The Quadrennial Defense Review(QDR) defines the strategies and initiatives that

respond to the current security environment. The most recent Quadrennial Defense Review was released in February 2010. [51] The recent version of the QDR document aims to describe how to rebalance the capabilities of the armed forces and how to reform the current Department of Defense institutions and processes. This leads to the purchase of weapons that are usable, affordable, and truly needed. The Quadrennial Defense Review is a foundation for the approach of starting with objectives and moving to capabilities, to activities, to resources.

The National Military Strategy is created by the Chairman of the Joint Chiefs of Staff for the Secretary of Defense. The document outlines the strategic aims of the military. The document receives guidance from the National Security Strategy document, the Quadrennial Defense Review, and yearly reports from the Secretary of Defense to the executive and legislative branches. The National Military Strategy is published before February 15^{th} on every even numbered year. The document describes the challenges and strategic environment in addition to the methods of addressing the challenges.

2.2.2 Joint Capability Integration and Development System (JCIDS)

The Joint Capability Integration and Development System (JCIDS) is an instruction regarding the requirements process used to deliver capabilities to the warfighter. The information in the following section is drawn from [19] unless otherwise noted. The main purpose is to make sure that warfighters receive the required capabilities to execute their missions. The Joint Capability Integration and Development System also supports the capability portfolio management process (described in [47]) used for capability investments. The Joint Capability Integration and Development System has undergone revisions since its first version in 2003. The system was designed to create a joint requirements generation process that replaces service-oriented processes. The Joint Capability Integration Development System document contains high level ideas

with the details contained in a web based manual. The military is incorporating the agility advances to providing detail information via websites for rapid dissemination of changes.

Joint Capability Areas (JCAs) are used within the DoD as a capability manage language and framework. Joint Capability Areas are defined as:

> collections of like DOD capabilities functionally grouped to support capability analysis, strategy development, investment decision making, capability portfolio management, and capabilities based force development and operational planning. [19]

The Joint Capability Integration and Development System is designed to work with the Defense Acquisition System by determining capability needs and performance criteria. The Planning, Programming, Budgeting and Execution (PPBE) process is supported with affordability advice for development and production lifecycle cost.

Both DOTMLPF analysis and Capability Based Assessment (CBA) can be used to begin the JCIDS process. The Joint Requirements Oversight Council(JROC) has three options to pursue to remedy the capability gaps identified from the CBA or DOTMLPF analysis. The JROC can accept operational risk and take no further action, seek a non-materiel approach, or recommend a materiel solution. The Initial Capabilities Document (ICD) document summarizes a CBA and justifies the mixture of recommended materiel and non-materiel solutions. If a materiel solution is chosen an Analysis of Alternatives(AoA) is performed to downselect for Milestone A decision. After Milestone A, the technology development phase begins which ends with the Milestone B decision. In the Capability Development Document (CDD) the operational technical performance attributes of the system are documented. A KPP is a Key Performance Parameter that describes attributes or characteristics of a system that are critical or essential to a capability. The KPPs are validated by the JROC prior to Milestone B. The proposed system enters Engineering and Manufacturing

Development (EMD) and the JROC is presented with the capability production document (CPD). The capability production document describes the actual performance of the system that delivers the required capability. Milestone C determines if the system will enter production and deployment. The ICD, CDD, and CPD are all JCIDS documents.

2.2.3 Defense Acquisition System

The Defense Acquisition System is designed to manage the Nation's investments in technologies, programs, and product support necessary to achieve the National Security Strategy and support the United States Armed Forces. The objective is to be able to quickly acquire quality products that satisfy user needs with improvements to mission capability at reasonable prices. The principles and procedures are derived from DoD Directive 5000.01 and DoD Instruction 5000.02. The Defense Acquisition System documents a set of best practices for acquisition. The guidebook is an electronic reference rather than a book and is laid out to get information to the user in the quickest way possible. [50]

The Defense Acquisition Guidebook covers eleven main topics. They are:

1. **Department of Defense Decision Support Systems:** which support strategic planning and resource allocation. Includes the determination of capability needs and the acquisition of systems.

2. **Defense Acquisition Program Goals and Strategy:** discusses the strategy for acquisition programs for program managers. It explains the Acquisition Program Baseline, the Technology Development Strategy, and the Acquisition Strategy.

3. **Affordability and Life-cycle Resource Estimates:** covers the program affordability and resource estimation.

4. **Systems Engineering:** covers system design issues and the system engineering processes.

5. **Life-cycle Logistics:** covers the life-cycle logistics from concept to disposal.

6. **Humans Systems Integration:** discusses human elements of the system engineering process.

7. **Acquiring Information Technology, Including national Security Systems:** Details the requirements on Information Technology (IT) and other related topics

8. **Intelligence, Counterintelligence, and Security Support:** covers the responsibilities of a program manager about inadvertent technology transfer and ways to protect technologies.

9. **Integrated Test and Evaluation:** covers oversight, Developmental Test and Evaluation, Operational Test and Evaluation, and Live Fire Test and Evaluation.

10. **Decisions, Assessments, and Periodic Reporting:** covers information for the program manager and the Milestone Decision Authority on their reporting and oversight responsibilities.

11. **Program Management Activities:** explains any loose ends for program managers that has not been discussed in previous topics.

As is evident from the topic and chapter summaries, the Defense Acquisition Guidebook is aimed at program managers and their support staff. The subsequent RAAM research is designed to interact with the Defense Acquisition System so it takes into account both the Joint Capabilities Integration and Development System (JCIDS) and the Planning, Programming, Budgeting, and Execution (PPBE) process.

The DoDAF v2.0 Manager's Guide describes the Defense Acquisition System (DAS) as:

> The DAS exists to manage the nation's investments in technologies, programs, and product support necessary to achieve the National Strategy and support employment and maintenance of the United States Armed Forces. The DAS uses Joint Concepts, integrated architectures, and DOTMLPF analysis in an integrated, collaborative processes to ensure that desired capabilities are supported by affordable systems and other resources [48]

DoDAF is covered in more detail in Section 4.1.

2.2.4 Planning, Programming, Budgeting, and Execution

The Planning, Programming, Budgeting, and Execution (PPBE) process in the DoD allocates resources and establishes a framework and process for decision making on future programs. [48] A recent RAND report [104] contains a useful summary of the PPDE process which is reproduced below:

- Planning: assesses capabilities, reviews threats, and develops guidance
- Programming: translates planning guidance into achievable packages in a six-year Future Years Defense Program (FYDP)
- Budgeting: tests for feasibility of programs and creates budgets
- Execution: develops performance metrics, assesses output against planned performance, and adjusts resources to achieve the desired goals

The programing phase of the process is aided by the Program Objective Memorandum (POM) process. Often the programming and budgeting phases are combined. [104]

2.2.5 Universal Joint Task List

The Universal Joint Task List is described in the Universal Joint Task Manual. [24] The manual is designed to provide a standardized tool for describing requirements for planning, readiness reporting, joint military operations, and joint training processes. The manual details how to develop Universal Joint Task List (UJTL) tasks, conditions, measures, and standards. The manual also discusses how to use those constructs to describe joint capabilities needed to support joint missions.

CHAPTER III

SYSTEM OF SYSTEMS ARCHITECTURE MODELING STATE OF THE ART

In chapter three, a subset of current system of systems architecture modeling efforts are discussed. Many different groups are working on aspects of early conceptual design of system of systems architectures. The following discussion is not exhaustive but it provides the reader with an understanding of the approaches and techniques in use today.

3.1 Assignment Problem

The RAAM methodology is most closely related to the assignment problem. The assignment problem is a combinatorial optimization problem from operations research. In the assignment problem, there are systems and tasks. Any system can be allocated to any task. Each system to task allocation incurs a cost. In the assignment problem, the costs are summed to produce a total cost. The assignment problem is a special case of a linear program that allows for specialized optimization beyond the simplex algorithm.

The generalized assignment problem is similar to the assignment problem. Any system can be assigned to perform any task. Each system and task assignment incurs a cost and produces a profit. Each system has a budget that can not be exceeded. The cost of all of the system to task allocations for a specific system can not exceed the budget of that system. Different approximation algorithms can be used that are efficient to solve the generalized assignment problem. [26] [66]

A more military based version of the assignment problem is called the weapon

target assignment problem. [2] The weapon target assignment problem is concerned with the optimal assignment of a set of weapons (systems) to targets (tasks) in order to maximize damage. The weapons target assignment problem can be formulated as a nonlinear integer programming problem.

The assignment problem is NP-Hard. Branch and bound methods which use approximations can find exact solutions. The assignment problem is a starting point for the types of problems that RAAM is designed to solve. RAAM adds nonlinear combination functions to the problem in addition to changing the structure of the different tasks that must be accomplished.

3.2 DARPA META & META-II Program

The Defense Advanced Research Projects Agency (DARPA) is working on two programs called META and META-II. The DARPA META and META-II programs are designed to substantially improve the design, manufacturing, and verification of complex cyber-physical systems. The information in this section is drawn from [33] [35] unless otherwise noted. Specifically, the programs hope to impact defense and aerospace systems such as ground combat vehicles, airplanes, and rotorcraft. The programs specifically exclude any evolutionary improvements and aim for revolutionary improvements. DARPA notes that hierarchical abstraction is often used with complex systems but not defense ones. DARPA would like to see a defense system created in one fifth of the time it normally takes. [35] The META program is a three phase effort: (1) design flow, metrics, and tools development and implementation, (2) component and manufacturing model library development, and (3) a rapid development demo.

The META program is interested in five different technical areas. They are:

- Metric of complexity
- Metric of adaptability

- Metalanguage for system representation
- Design flow and tools
- Verification and tools.

A metalanguage is a language that is used to discuss or examine another language. For the metalanguage for system representation, the language should enable and support seven uses. The relevant uses related to this work are: (1) the introduction of hierarchical abstraction layers into the design process and (2) the rigorous exploration and use of advanced optimization methods for large multi-dimensional design trade spaces.

The Appendix F associated with the META-II Broad Agency Announcement (BAA), *Abstraction Based Complexity Management* [34], contains seven papers. The first, second, third, and sixth papers are explored in greater detail below. It should be noted that the fourth, fifth, and seventh papers in the series were not deemed relevant by the author. In the overview of these papers, they mention two areas of research that are relevant to the RAAM work. They are:

- "Define an abstraction-based design method to provide formalism to the definition of the system"
- "Determine a method of architecture synthesis that can be used to explore the complete design space available during early conceptual design"

The overview discusses an approach to developing architectures that utilizes a filter based method. As an overview, the filter based methods first identify the complete set of possible instantiations and then finds the instantiations that are feasible. The feasible sets are then further analyzed. In a fighter aircraft design, over 27,000 feasible architectures were identified. The current RAAM research work will have similar motivations with different methods.

In the first of the seven papers associated with the META-II BAA, *Design System for Managing Complexity In Aerospace Systems*, one possible design system uses four key elements to design advanced aerospace systems:

- Abstraction Based Design Tools
- Quantitative Complexity Metrics
- Advanced Architecture Synthesis Methods
- Robust Uncertainty Management

The two most relevant elements to the subsequent research enclosed in this manuscript are the Abstraction Based Design Tools and the Advanced Architecture Synthesis. The Abstraction Based Design tools provide a way to design and evaluate complex heterogeneous systems. The Advanced Architecture Synthesis uses a set of tools that enable formal and automated architecture synthesis, enumeration, and evaluation of feasible architecture options. Advanced Architecture Synthesis Methods have three challenges, next generation systems becoming more and more complex and multidisciplinary, the superior evaluation of architecture options early in the design cycle, and to understand how the architecture can be partitioned into sub-domains that limits the spread of complexity. The first paper calls for semantics and a language that allow for the combination of Platform Based Design and model based design methods. [34]

The second paper, *Correct-By-Construction Design of Aircraft Electric Power Systems*, describes a bit of the Platform Based Design. The framework distinguishes between the function and architecture. This allows for automatic design space exploration. The author's work distinguishes between the function, architecture, and computational model which also allows for automatic design space exploration. As described in this second paper, other researchers are designing languages to design complex systems, such as electronics, avionics, power subsystems, and aircraft.

The third paper, *Assessing performance uncertainty in complex hybrid systems*, discusses ways to handle uncertainty in complex systems. The paper highlights similar motivations to the author's which is that an increase in system complexity comes from an increase in the number of parts, an increase in the number of interactions, and the integration of multiple technologies. The paper concludes that the selection of appropriate system architectures is critical for system robustness.

The sixth paper, *System Complexity Reduction via Spectral Graph Partitioning to Identify Hierarchical Modular Clusters*, discusses ways to locate lower complexity architectures. They mention that many methods do not attempt to enumerate all of the potential options of an architecture as the design space is exponential. The feasible set is sparse, which may make the problem tractable.

DARPA is looking to improve the design (all phases) of a system. The RAAM research is aiming to improve the conceptual design phase of a system of systems architecture. The META programs are designed to work on a system, whereas the enclosed research is attempting to go up a level of scope and work on a system created from systems, a system of systems.

3.3 Architecture Evaluation and Enumeration

United Technologies Corporation has created a process called Architecture Evaluation and Enumeration [133]. The framework is used to look at millions or billions of design configurations downselect to determine the thousands of feasible configurations and then downselect again to find the promising concepts that will be modeled in higher fidelity environments. The complex and multi-disciplinary nature of emerging systems requires new ways to make the initial downselection. The paper mentions that enumeration is not commonly used due to the large size of the design space. The framework initially uses compatibility information to narrow the design space down to hundreds or thousands of architectures.

3.4 Architect

Griendling and Mavris [75] have created a process for system of systems architecting entitled Architecture-Based Innovation, Technology Evaluation, and Capability Tradeoff (ARCHITECT) process. The overall motivations and goals of the methodology are the same as the current research. The methodology uses DoDAF-like products and modifies them to become inputs to modeling and simulation in support of early phase decision making.

Executable models are created from the DoDAF-like products. The generation of executable models is automatic or semi-automatic. Since system of systems architectures do not often have useful metrics for modeling and simulation, the methodology includes a method for determining the proper set of metrics to use for analysis on the systems of systems. There are a variety of modeling methods for the architecture including Markov chains and Petri nets. RAAM has been integrated with ARCHITECT.

3.5 Portfolio Analysis Tool

3.5.1 Strategy to Tasks

The Strategy to Tasks framework was developed at RAND and was created by Lt. General Glenn Kent[1]. The framework is used for force planning that consists of a hierarchy of objectives. The framework is traceable, helps with communicating how Service activities support security needs,is a common frame of reference between the Services, provides a structure for operational alternatives tradeoffs, and assists in the development of new concepts for improved military capabilities. [137] The main idea is to link the strategies all the way to tasks and onward to the systems that actually accomplish the tasks.

[1]It is recommended that everyone interested in analysis read his memoir, *Thinking About America's Defense: An Analytical Memoir* [90]

3.5.2 Portfolio Analysis Tool

The RAND Corporation has created an analysis tool designed to assist in portfolio planning called the Portfolio Analysis Tool (PAT). The following information is drawn from *RAND's Portfolio Analysis Tool* [37] unless otherwise noted. More information is also available at [118] and [60]. The purpose is to facilitate strategic portfolio analysis that involves uncertainty and differences in perspective. Different portfolio options are compared using different measures and cost. The Portfolio Analysis Tool is not a model, but rather it is a tool for creating models.

The tool is designed in a hierarchical manner which permits a "drill down" into higher levels of detail as a means of understanding issues. The datasets at the lower levels of scope are aggregated using combining rules or aggregation rules. A commonly used set of aggregation rules is defined but the analyst may define new rules. Different assumptions about the relative weights of the objectives and assessment of capabilities are considered to be perspectives. The different portfolios can be compared across alternative perspectives.

The Portfolio Analysis Tool can create a variety of outputs. The Summary sheet provides a scorecard view of the different options and their performance in different scenarios across different measures of effectiveness and cost. Cost-Effectiveness Landscape are used to provide a visual method to compare top level metrics of different portfolios across multiple perspectives. They are designed to be used after the decision maker has discovered their preferences and selected a subset of portfolios and perspectives to compare. Risk management is another output of the Portfolio Analysis Tool. Risk can be shown in using top level metrics, lower level metrics that are associated with a demanding test case, different aggregation methods, or warning flags.

The tool allows the consideration of different levels of scope of the problem at hand. The RAND document authors note that different metrics should be evaluated

at different levels of scope. For example, technical risk should be evaluated at a high level of scope. Drill down is especially important for traceability and to allow decision makers to properly balance across the criteria. Top level aggregate measures are useful after the decision makers are oriented to the problem at hand.

The tool promotes traceability by providing the drill down option so that scorecards are not the only information available in briefs. Raw values from subject matter experts, analysis tools, and empirical methods is converted into a score that is on a common scale that is shared between the different measures. The scorecards do not show the numerical value of the score but map a numerical value to a color range. The set of colors is kept at five so that the decision maker will not be confronted with a rainbow that camouflages the difference between portfolio options. From worst to best the colors are red, orange, yellow, light green, and green. Davis and Dreyer note that five colors is enough to separate options but does not induce cognitive overload. Currently the color scale is a linear scale with ranges of equal size. An example is shown in Figure 4 from [60]. In addition, the Portfolio Analysis Tool has other output displays to help understand information. This is shown in Figure 5 from the same source.

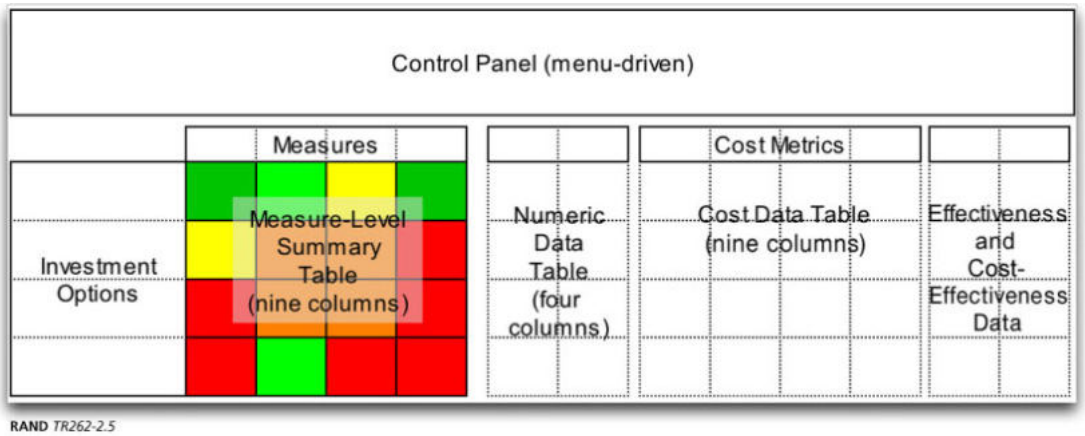


Figure 4: RAND PAT-MD Summary Sheet [60]

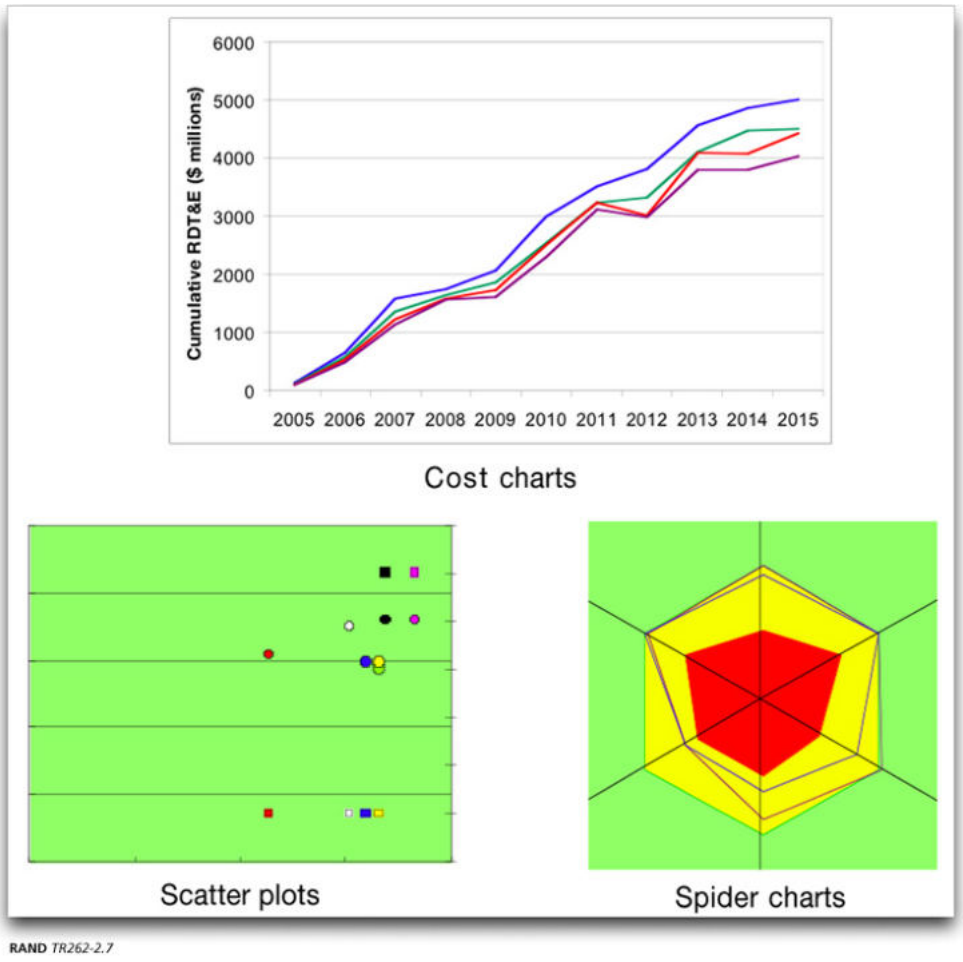


Figure 5: RAND PAT-MD Output Displays [60]

Different aggregation methods have been shown to be useful in practice. Raw values are transformed using a one to one mapping into scores. The scores are aggregated into a one level higher score using a many to one mapping. There are five built in aggregation methods as follows:

- Goal based
- Goal-based with weak thresholds
- Goal-based with thresholds
- Goal-based with weakest link

- Rankings-based

The RAND authors caution against the inclination to use the Portfolio Analysis Tool to optimize resource allocation mathematically. They view the main outputs of the tool to be the portfolio scorecards and the ability to drill down into the next level of scorecards. This allows for the ability to change assumptions and priorities. The use of an overall cost effectiveness number should be limited to a refinement and communication stage of decision making.

3.5.2.1 Portfolio Analysis Tool Limitations

The Portfolio Analysis Tool is an spreadsheet (Microsoft Excel) based tool. Certain limitations arise from using a spreadsheet when creating multiresolution models. Due to the manual nature of most spreadsheet based coding, a static architecture is typically chosen for analysis.

The Portfolio Analysis Tool is limited to four different levels of resolution. The analyst is tasked with ensuring consistency between the different sheets within the spreadsheet based model. Even within the Portfolio Analysis Tool reference manual, there are hints and notes pointing to workarounds for spreadsheet limitations. Once the data is populated, it seems hard to change formulas and add more portfolio options.

Raw values are converted into a common scale so that they can be combined together. The common scale is between zero and one. All of the models that you can create with this framework use this common scale at the submeasure, measure, and overall-effectiveness score levels. This limits the ability to create more complex combinations of metrics that better match the requirements of decision makers.

The developers of the Portfolio Analysis Tool acknowledge many of the limitations of the tool. The tool is not "industrial-strength software" to use their choice of words. It is also not "gorilla-proofed" (which nixes plans to use cheap gorilla based labor for

strategic planning and capability assessment).

CHAPTER IV

DESCRIBING ARCHITECTURES

There are two main ways that people model system of systems architectures, system models and architecture models. The systems models tend to be used later in the design process as compared to the architecture models.

The system models tend to be higher fidelity. The system models are typically agent based models that interact with each other in a scenario and focus on the performance of the system. Naval system of systems workflows have been modeled using agent based models. [97] Another example is a multi-objective model for Humanitarian Infrastructure. [127] The system modeling efforts require detailed information about the systems that is not often available in conceptual design. The system models often are not designed to handle novel system concepts within the existing modeling frameworks.

Architecture models tend to be lower fidelity. The architecture models often describe the connections between the systems but are limited in their modeling capability. DoDAF, UML, and SysML have been proposed to be used for system of systems architecture analysis.These tools are elaborated upon in the subsequent sections. Those three modeling languages and frameworks are useful after the architecture has been designed as they serve as both the documentation and description tools. Other industries are developing architecture frameworks, for example the automotive industry. [15] Enterprise architecture models are used in industry to determine the costs of changing enterprise wide systems. The accuracy can be quite high and they are useful for decision making. [93] Architecture based approaches are also used in the health care industry. [96]

The RAAM research will not consider the system models. Agent based models require too much information for the conceptual phase of system of systems design. The system models also tend to require large amounts of computing resources making them prohibitive to use to evaluate all of the architecture alternatives.

4.1 DoDAF

The Department of Defense Architecture Framework, DoDAF, was designed to be an, "overarching, comprehensive framework and conceptual model enabling the development of architectures". DoDAF version 2.02 was released in August 2010. The following section will draw on information contained within the DoDAF v2.0 Manager's Guide [48] unless otherwise noted. DoDAF was created to help managers make decisions more effectively by sharing information across institutional boundaries such as the Department, Joint Capability Area, Mission, Component, and Program. DoDAF provides architecture concepts, guidance, best practices, and methods associated with architecture development. For a program to be in compliance with DoDAF, it must have the architecture described using the DoDAF Meta-Model (DM2) and the architecture data must be transferable using the Physical Exchange Specification (PES). The architectures created with DoDAF v2.0 can be used to support the Joint Capabilities Integration Development System (JCIDS), Defense Acquisition System (DAS), Planning, Programming, Budgeting, and Execution (PPBE), System Engineering (SE), and Portfolio Management (PfM) processes. DoDAF v2.0 utilizes concepts and constructs from the UK Ministry of Defence Architecture Framework (MODAF), NATO Architecture Framework (NAF), and Open Group Architecture Framework (TOGAF). The standard has moved to a continuously updated website for specifying the standard. In addition, there is a private DoDAF website that requires a government sponsor and an account to access. [151].

DoDAF v2.0 is designed to help domain experts, program managers, and decision

makers locate, identify, and resolve definitions, properties, facts, constraints, inferences, and issues using viewpoints created from raw data. What-if analysis is possible to analyze when something is redefined, redeployed, deleted, moved, delayed, accelerated, or no longer funded. The authors remind us that, "Architectures are a means to an end... not an end to themselves" [ellipsis in the original]. They also state, "DoDAF is the structure for organizing architecture concepts, principles, assumptions, and terminology about operations and solutions into meaningful patterns to satisfy specific DoD purposes." [48]

4.1.1 DoDAF History

DoDAF v2.0 is the current version of a set of ideas that had been described in the Technical Architecture for Information Management (TAFIM). The TAFIM had three major upgrades ending in TAFIM v3.0 [68]. The TAFIM transitioned into the Command, Control, Communications, Computers, and Intelligence, Surveillance, and Reconnaissance (C4ISR) Architecture Framework v1.0 in 1996. The C4ISRAF was updated to version 2.0 in late 1997 [16]. The C4ISRAF v2.0 was based on defining operational, systems, and technical architecture views. In 2003 DoDAF v1.0 [52] was released based on the C4ISR v2.0 in an attempt to utilize the useful concepts about architectures for all of the Joint Capability Areas and not only support C4ISR. In 2007 DoDAF v1.5 [53] was released as a transitional version of DoDAF. DoDAF v1.5 included guidance on incorporating net-centric concepts and an updated version of the Core Architecture Data Model (CADM). DoDAF v2.0 resulted in significant changes from previous versions. These prior releases, v1.0 and v1.5 were much more similar. DoDAF v2.0 can include many of the concepts from the previous version, but does not require a given set of views or products. [48] The new version of DoDAF (version 2.0) is designed to shift the focus from products to data. They predict better analysis and decisions from the shift in focus. [150]

4.1.2 DoDAF Overview

A major change from the previous versions of DoDAF [52] [53] occurred in version 2.0 with an emphasis on architectural data in support of decision making and not an end unto itself. The old products are replaced by the concept of "Fit-for-Purpose" where the architectural views are made in support of a specific project or mission objective. The visualization of the architectural models is done using viewpoints which are a collection of views. A viewpoint with the necessary definitions is an Architectural Description. The writers hope that the users of DoDAF will gather the necessary data only at the required level to enable known decisions or objectives. Two models for visualization are supported, DoDAF-described Models and Fit-for-Purpose Views. The DoDAF-described Models are described, appropriately, in the DoDAF v2.0 Volume 2. [49]. The Fit-for-Purpose Views are designed to allow for agency customization and the incorporation of existing views that may be useful to the agency.

The new DoDAF Meta-Model (DM2) replaces the Core Architecture Data Model (CADM) from previous version of DoDAF. The DM2 contains a Conceptual Data Model (CDM), Logical Data Model (LDM), and a Physical Exchange Specification (PES). The change is part of the DoD movement away from the older Command, Control, Communications, Computers, and Intelligence Surveillance Reconnaissance Architecture Framework (C4ISRAF) and DoDAF v1.0/v1.5 ideas. The approach is so data-centric that the, "creation of architectures to support decision-making is secondary to the collection, storage, and maintenance of data needed for efficient and effective decisions." Volume 1 continues later in the document, "*DoDAF does not prescribe any particular models, but instead concentrates on data as the necessary ingredient for architecture development.*" [quote was underlined for emphasis in the original].

In addition, the newer version does not include a description of the physical data

model, which allows for freedom to each software vendor to store the data how they choose to do so. The data is exchanged between departments and software using the Physical Exchange Specification. The Physical Exchange Specification is designed to help with the federated approach advocated in DoDAF v2.0. The different architectures within the DoD are distributed using a tiered accountability model over four tiers: Department, Joint Capability Area, Component, and Solution.

DoDAF v2.0 is currently in flux and appears to be headed toward a moving target of common practices. The three volumes describing DoDAF are complemented by an electronic journal on the Defense Knowledge Online website [45] (A publicly accessible website is at [54]). The electronic journal contains examples from the DoDAF volumes, best practices, lessons learned, and reference documents. As the standard is currently being developed, there are elements of DoDAF that will change. As DoDAF v2.0 is used throughout the DoD, the author envisions changes to the standard as a result of the new push for a data-centric approach. DoDAF v2.0 does not throw out the previous versions and the products can continue to be used as views in the new architecture framework. DoDAF has been evaluated against other defense industry frameworks such as the Ministry of Defense Architecture Framework (MoDAF) and the NATO Architecture Framework (NAF). [4]

There are two types of architectures defined in DoDAF v2.0, Enterprise-level Architectures and Solution Architectures. An enterprise architecture defines the mission, the required information, the required technologies, and the transition process to handle additional technologies. An enterprise architecture is used to roadmap change and describe how the different programs fit into the larger enterprise. A solution architecture is, "a framework or structure that portrays the relationships among all elements of something that answers a problem." Out of the two architectures, solution architectures are most used by the DoD. The solution architectures are required for solution evaluation, interoperability and resource allocation.

DoDAF v2.0 describes a six step process for architectural development. The six steps are:

1. Determine the intended use of the architecture
2. Determine the scope of architecture
3. Determine data required to support architecture development
4. Collect, organize, correlate, and store architecture data
5. Conduct analysis in support of architecture objectives
6. Document results in accordance with decision maker needs

Figure 6 shows what the decision maker needs do to in the DoDAF Six Step Architecture Development Process. The process allows for flexibility during the execution of the process as it remains top level. Fitting into the data centric ideas, the process focuses on determining the uses for the architecture and then collecting and using the data. The process explicitly contains a step where the results of analysis are documented for future use.

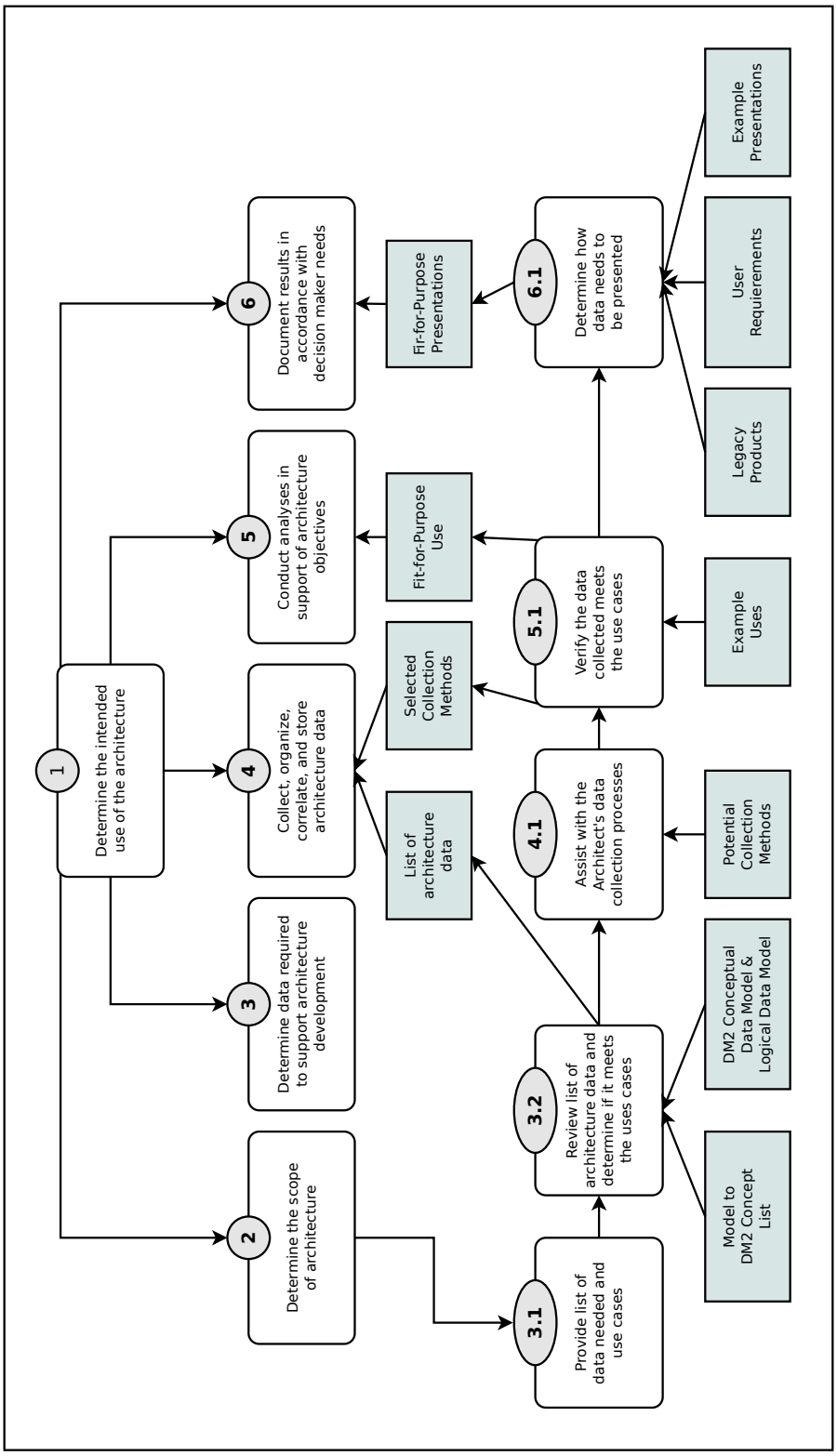


Figure 6: DoDAF Six Step Architecture Development Process Adapted from [48]

DoDAF v2.0 defines an architecture development methodology as a methodology that "specifies how to derive relevant information about an enterprise's processes and business or operational requirements, and how to organize and model that information." The three different types of analysis are typically used to evaluate architectures are as follows:

- Static Analyses
- Dynamic Analyses
- Experimentation

Static analyses are applied to attributes of the architecture that are independent of temporal, spatial, or performance aspects of the architecture. Dynamic analyses analyze the temporal, spatial, or performance attributes of the architecture.

There are eight architecture viewpoints in DoDAF v2.0. The eight viewpoints are the all, data and information, standards, capability, operational, services, systems, and project viewpoint. The eight viewpoints offer different information about the architecture. The viewpoints should share the relevant data. Volume I continues to describe fifty two different models used to describe an architecture.

The RAAM research focuses on a few of the models as the information contained within the following models is useful for architecture analysis. The OV-1 model which is a high level operational concept graphic is often used to provide a picture of an architecture to get people in the right frame of mind. The OV-5a is an operational activity decomposition tree that shows the capabilities and activities in an hierarchical structure. The SV-5a is the operational activity to systems function traceability matrix which provides a mapping of system functions back to operational activities.

Different tools are usable within DoDAF v2.0. There are two main concerns about software tools used for DoDAF v2.0. The first of which is that the software must be able to utilize the Physical Exchange Specification (PES). The second concern is that

it must be capable of XML data transfer to and from the DoD Architecture Registry System (DARS) and the DoD Metadata Registry (DMR).

DoDAF v2.0 is a flexible architecture framework that should be used with other frameworks, tools, and techniques. The framework is a beginning for the analysis of architectures and decision making. DoDAF has been extended to allow for Discrete Event System Specification. This adds a simulation capability to DoDAF by adding two new operational views. [108]

4.1.3 DoDAF Definitions

DoDAF v2.0 defines twenty five concepts in the Conceptual Data Model (CDM). The relevant definitions are for activity, capability, measure, and system. An activity is defined as, "Work, not specific to a single organization weapon system or individual that transforms inputs (Resources) into outputs (Resources) or changes their state." A capability is defined similarly to the JCIDS definition, "The ability to achieve a Desired Effect under specified (performance) standards and conditions through combinations of ways and means (activities and resources) to perform a set of activities." A measure is defined as, "The magnitude of some attribute of an individual". A system is defined as, "A functionally, physically, and/or behaviorally related group of regularly interacting or interdependent elements.". [48]

4.1.4 DoDAF Limitations

Despite the benefits of DoDAF, there are limitations that should be considered. Due to the new data centric push in DoDAF v2.0, there is a large amount of flexibility in being in compliance. DoDAF v2.0 is designed to be useful for decision making, but only considers analysis between two architectures, the "As-Is" and the "To-Be". Decision makers are interested in evaluating more than two architecture alternatives. DoDAF v2.0 is designed to help communicate information about an architecture

within the DoD enterprise and can be used to describe the transition between a current architecture and a future architecture. Before decision makers can be presented with a current and future architecture, someone must do analysis and decide on the future architecture from a set of alternatives. DoDAF v2.0 is not well suited to analyzing many different architectures with different functional breakdowns and different systems.

DoDAF v2.0 does not specify the method of analysis used to support architecture objectives. The users gain flexibility in using the architecture to support their own decision processes, but the lack of a standard can make the federation goal difficult. Comparing the results of different analyses using different analysis methods can lead to different top level results.

There are a few problems with the currents plans for DoDAF. The Physical Exchange Specification Developer's Guide shows a translator between the UML XMI XML document to the DM2 PES XML document. The translator is described as not existing yet. [150] A user is not required to use UML but many existing architectures are in UML.

DoDAF v2.0 lists five ways that architecture frameworks support organization change, two of which are highlighted here. The first is to facilitate the design of future states of the enterprise. The second is to establish a baseline architecture for solutions under development. A baseline is required for comparison to how we accomplish the capability today. Future states of the enterprise would include the new ways to accomplish the capability. DoDAF is designed to look at a low number of alternatives due the manpower intensive nature of generating new architectures. The conversion of an architecture into an executable model is not addressed within the standard.

DoDAF v2.0 does not contain a uniform representation of the semantic content

within an architecture model. The semantic representation will be known as Architectural Modeling Primitives and will be a standard set of modeling elements and symbols that map to the DoDAF Meta-Model (DM2). DoDAF v2.0 recommends utilizing the Business Process Modeling Notation for the OV-6c models. DoDAF v2.0 specifically notes, "While DoDAF does not require specific models or views in an architecture, several JCS and DoD publications do require specific views in response to their stated requirements." The managers are responsible for their compliance of other requirements placed on their architectures. The lack of a standard for creating a visualization of architectural data means that different organizations within the DoD will have trouble communicating using architecture visualizations.

DoDAF v2.0 can utilize many different tools from multiple vendors to collect, organize, and store architectural data. The different viewpoints are not standardized and the examples shown in the standard are only for example. The authors note that a limited set of models is normally created which are used to capture and collect the architectural data. The limited set of data should conform to the Conceptual Data Model (CDM) and the Logical Data Model (LDM). Because there is "no single, correct way to visualize any view" [48] the decision makers may have a hard time comparing architectures. With different types of visualizations the useful realization of a federated enterprise may require more work than originally envisioned. DoDAF v2.0 mentions, "Architects must be able to communicate architectural information in a meaningful way to process owners and other stakeholders, or the discipline of an enterprise architecture will soon meet an untimely demise". Complex ideas are often conveyed with diagrams (such as Feynman Diagrams [87]) and architectures should be no different.

The DoDAF is still in development. Currently, the DoDAF 2.0 Work Group is working on examples for DoDAF and the ability to do a viewpoint/metamodel cross-check [55]. The rapid development of the standard makes compliance efforts

difficult due to the disparity between the time to make changes to the framework and the average lifetime of a program. The military is still working toward a common language, but it still needs to clarify definitions for things such as mission architecture, joint mission threads, and command enterprise architecture [42].

Since the development of RAAM, the DoD has moved toward similar ideas as RAAM by developing the capability Taxonomy viewpoint(CV-2). The capability is broken down into a hierarchical structure. [94]

The required architecture artifacts for DoDAF are evolving rapidly. Figure 7 shows the required DoDAF artifacts during different phases of the acquisition process. The red marks are for new requirements in the most recent version of CJCSI 6212.01F (Chairman of the Joint Chiefs of Staff Instruction). The big change is that Capability Views are now required. Activity diagrams (OV-5a and OV-5b) are also required. The Initial Capabilities Document (ICD) used to only require the OV-1 which is a high level graphic of the architecture in operation.

Overall, it is difficult to use DoDAF for Pre-Milestone A decision making as the framework does not apply before Milestone A. DoDAF is designed to document two viewpoints on an architecture, an "as is" and a "to be" rather than billions of architecture alternatives.

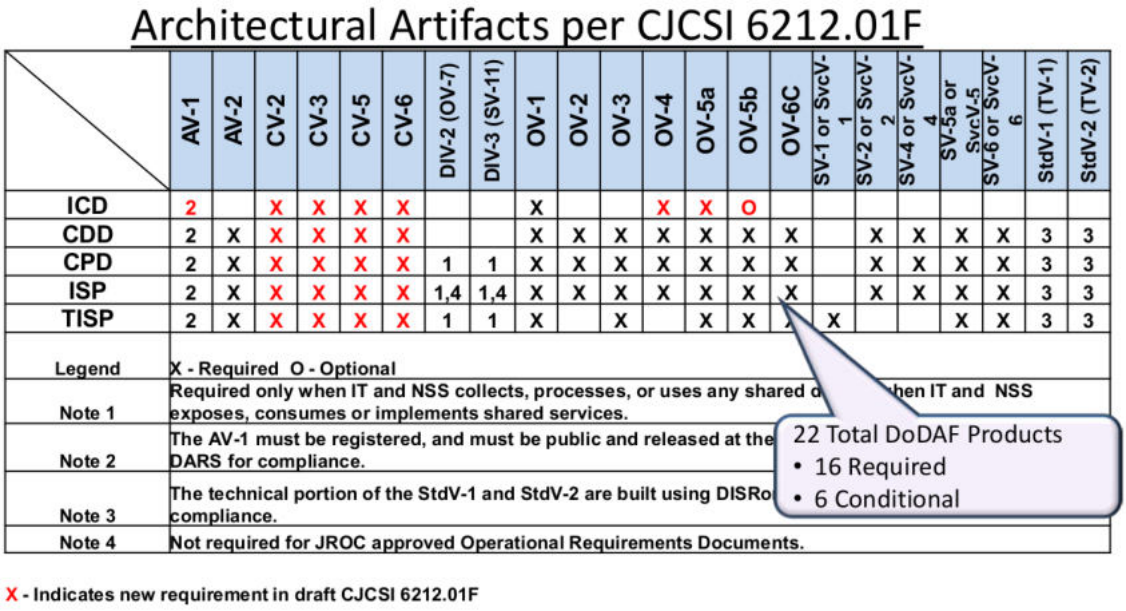


Figure 7: Required DoDAF Architectural Artifacts [42][1].

4.2 UML Related Models

4.2.1 UML

The Unified Modeling Language (UML) has been in constant development since 1996. Version 2.3 was released in May of 2010. [115] The Unified Modeling Language is designed to document the structure or architecture of an enterprise application. It is designed to specify, visualize, and document models of software systems. As its use has progressed, new derivative languages such as the Object Constraint Language, Systems Modeling Language (SysML), and the Unified Profile for DoDAF and MoDAF (UPDM) have been created. A UML model can be transferred between multiple proprietary tools using the XML Metadata Interchange (XMI) standard. The Unified Modeling Language is methodology independent.

There are thirteen types of diagrams defined in the Unified Modeling Language. The thirteen types of diagrams are organized into the following three groups:

[1]CJCSI 6212.01F is unreleased as of February 14th, 2012.

- Structure diagrams
- Behavior diagrams
- Interaction diagrams

The Unified Modeling Language is traditionally used to describe a software system, so more detail will not be provided here. The Unified Modeling Language was never designed to help with an analysis of alternatives, but rather it was designed to help with describing a system.

Researchers have adapted UML for the creation of domain specific modeling environments. UML is used to describe a meta-model that is composed and create graphical environments. [88] UML object diagrams are also used to describe automatic model transformations. The use of UML promotes reuse and results in shorter production time. [107] Process models are often modeled using UML. Osmundson et. al demonstrated a method for architecture analysis of system of systems using process modeling with UML. [117]

There have been quite a few critics of the Unified Modeling Language. They normally do not think that the language itself is very flawed, but the use of the language often is. Alex Bell wrote a paper entitled, *Death by UML Fever* where he compares the improper use of UML and belief in UML to a viral infection that has many effects on the development process. [9] He was associated with the Boeing Company when he wrote the article, calling upon twenty two years of experience. He uses UML diagrams to describe his taxonomy of false beliefs in UML. There are four main types of 'metafevers' called delusional, emotional, Pollyanna, and procedural. Each of those types is further broken down. As you can see, UML can create intense feelings for and against its use. The most relevant message from the paper is to not force UML or similar tools to fit a specific problem if it does not actually fit the needs of the problem at hand.

4.2.2 Object Constraint Language

The Object Constraint Language is used to build software models. The following information is from [149] unless otherwise noted. The Object Constraint Language is part of the UML standard. The Object Constraint Language is a query and constraint language at the same time. It is also declarative and specifies what should be done, not how.

4.2.2.1 Object Constraint Language Limitations

The Object Constraint Language book [149] brings up a few points about the current state of automatic translation that bear repeating. There is hope for an efficient and correct way to do the translation from a Platform Independent Model to a Platform Specific Model (PSM). The Platform Specific Model (PSM) can then be translated into code to run on the computer. Robust translators do not exist. The Object Constraint Language does not have a readily available mechanism to change the model which would be required for system of systems architecture modeling.

4.2.3 SysML

The following is taken from [114] unless otherwise noted.

The Systems Modeling Language (SysML) is a general purpose modeling language variant of the Unified Modeling Language that is designed for system engineering applications. SysML supports specification, analysis, design, verification, and validation of systems and systems of systems. There are three main types of diagrams, the behavior diagram, the requirement diagram, and the structure diagram. Activity, Sequence, State Machine and Use Case diagrams are all behavior diagrams. The requirement diagram is a new diagram type from the Unified Modeling Language. The Structure diagrams include Block Definition, Internal Block, Parametric, and Package diagrams.

The website contains many papers and presentations that discuss the use of SysML

to solve real world problems. Many of the examples are for continuous dynamical systems or physics based CAD/CAE models.

One of the originators of the SysML standard tells a different narrative. The following comments in this paragraph on SysML are the personal opinions of one of the originators of SysML and not representative of the SysML originators as a whole. The tone is overall very negative and may be biased. The following is taken from [134] unless otherwise noted. The standard is currently under revision due to UML 2.x bloat and 'voodoo semantics'. The documentation on the website points to problems with the language. SysML is noted as being marketed as a 'smaller, simpler' language on its website, but also talks about language bloat due to its additions to UML and notable lapses in removing unused elements of UML. Remarkably, continued on the same website FAQ is the mention of a disconnect between the 'marketecture' descriptions of the SysML support for precise semantics for specifying parametric constraints versus reality. The website continues discussing a few small problems with SysML, the parametrics diagrams in SysML are incomplete and lack precise syntax and semantics. The ports/interfaces are complex and muddled.

The website is not completely negative, it lists four advantages to SysML over UML for systems and systems of systems. The systems engineering semantics in SysML are expressed better than UML. SysML is explained to be smaller and easier to learn than UML. SysML has constructs such as allocation tables that allow for automated verification and validation and gap analysis. Fourth, the model management constructs are aligned with the IEEE-STD-1471-2000.

Six different SysML tools are shown on the website and ranked in terms of usability, drawing, simulation/excitability, standards compliance, value, and overall. None of the six tools have an overall score much over three out of five stars (the overall score seems to be an average of the ranking metrics so it does not have to be an integer number of stars). This suggests that the tools required for SysML are still

being developed to their full potential. As a testament to the utility of SysML, even with the lower scores for the SysML tools, SysML is being used successfully in many projects. SysML can be used for both software and hardware modeling. [77]

A SysML based methodology has been developed for manufacturing purposes. [8] SysML provides a way to support the design of complex systems. A hierarchy of models is proposed to manage the complexity and allow for designers to have information about all levels of scope in the system. Other engineers are concerned with creating early design models from requirements. Colombo et. al. have created a method for generating a design model from transformations applied to a requirements model. [27] Architecture design has been partially automated using SysML for specification and modeling. [67] In addition, formal methods have been used for aerospace applications. Consistency in aerospace design is key, and formal logic can be added to SysML to ensure consistency. [72] Requirements analysis is becoming more difficult as the complexity of the constituent systems increases in a system of systems. Extensions to SysML diagrams can provide the required elements to describe a requirements modeling language. This is proved with a Road Traffic Management System. [59] SysML has been directly used for system engineering. [152]

4.2.4 UPDM

UPDM is the Unified Profile for DoDAF/MoDAF designed by the Object Management Group (OMG). UPDM can use many of the same tools as UML and SysML. UPDM contains a set of common core elements between DoDAF and MoDAF in addition to specializations for each architecture framework. UPDM is designed to provide overarching architecture concepts, guidance, best practices, and methods for architecture development. [62] The motivation for UPDM is summarized in [76], which is to enhance the quality, productivity, and effectiveness of system of systems architecture modeling, promote architecture model reuse and maintainability, improve tool

interoperability, and reduce training impacts.

UML does not have the semantics, constraints, and rules that ensure a correct model in DoDAF. [76] Due to the deficiencies in UML for DoDAF, UPDM can be used to achieve full DoDAF compliance.

Walt Okon, the current Chief Information Officer (CIO) for the Office of the Secretary of Defense, describes a strong support within the Department of Defense and Ministry of Defense (UK) for UPDM. [62] He continues to mention that the DoD will push for UPDM to become a mandated standard and will promote internationalization of UPDM. UPDM will be another layer on top of DoDAF v2.0 designed to improve the exchange of architectures between organizations and architecture tools.

Beyond the DoDAF Physical Exchange Specification, the UPDM provides additional mechanisms for tool interoperability. As DoDAF has transitioned to version 2.0, UPDM has been updated to support the new direction. UPDM v2.0 can optionally use the Business Process Modeling Notation (BPMN). There are eight mandatory requirements defined in UPDM: the Domain Metamodel, Metamodel, Profile, Notation, DoDAF/MoDAF artifacts, additional views and viewpoints, an element taxonomy reference, and data interchange. UPDM compliance level 0 includes part of SysML. UPDM compliance level 1 includes SysML, UPDM compliance level 0 and parts of UML. SysML blocks, activities, parametrics, and allocations are all included in UPDM. [76]

4.2.4.1 UPDM Limitations

UPDM was not designed to help with the conceptual analysis of system of systems architectures. The goals of analyzing different allocations of system to tasks and multiple structural changes to the architecture are infeasible. Manual processes are used to create the architecture views and computational models that are dependent on the system to task mapping.

The current plan for DoDAF v2.0 and therefore UPDM are in flux. UPDM was originally conceived with the intent of satisfying the DoDAF 1.5 criteria. UPDM is useful with DoDAF v2.0 because DoDAF 1.5 views are valid DoDAF v2.0 viewpoints. There is strong industry and organizational support for UPDM.

4.3 Object Process Methodology

Dov Dori developed OPM, the Object-Process Methodology [58] to represent the interactions within a system and be applied to a variety of components including electrical, informational, mechanical, optical, thermal, and human. The system is broken into the constitute objects and the processes relating the objects. OPM is a modeling language consisting of two parts, Object-Process Diagrams (OPD) and Object-Process Language (OPL). The ability to translate the graphical into the textual and vice versa is cited as a strength of the OPM methodology. The new methodology that will be developed in the following pages will have the same property.

OPM was designed to express the function, structure, and behavior of a system. OPM defines function as what the systems do, structure as how the systems are constructed, and behavior as how systems change over time. One of the design goals of OPM was that it would be designed to be useful and not forced into compromises due to limitations in computer languages. A system in OPM is built from objects, processes, and states (of the objects). In contrast to the object-oriented paradigm, the processes/functions are not second class, they are not subordinate to an object.

OPM was designed similarly to the other architectural modeling tools in that it is designed to document one architecture at a time. New diagrams or descriptions will need to be created for each new architecture. In addition, it does not have a method for storing the values of a system accomplishing a task for a given metric. OPM is a declarative language which is used for storing relationships between different objects, processes, and states. The following RAAM research is interested in going farther

than modeling the relationships to modeling system of system metrics of interest using the information contained in the system relationships and system metric values.

To help manage the complexity involved in architecting, OPDs can be hierarchical. OPM includes a zooming process for OPDs that allows for the removal of detail across OPDs as long as the diagrams do not contradict each other. The authors recommend the use of zooming as practical architectures would result in too much complexity. The architectures described in OPM are directed acyclic graphs and therefore do not contain loops or cycles. The hierarchical links can be explicit or can be hidden when objects are contained within other objects.

OPM was designed to accomplish two goals. The first of these goals is to help with analyzing and designing a system. The second goal is to help with generating the desired application. The author of OPM seemed to be concerned with partially automatically generating systems that were computer programs. OPM was envisioned to contribute to executable code and the database schema. OPM was not designed to help generate system-level computer models to predict attributes of the system from sub-system attributes.

The author of OPM also discusses some of the limits of UML. UML does not treat attributes of an object as objects. The ramifications are that the attributes can not be decomposed or have attributes. Due to UML not providing a direct way to decompose attributes, a user must add more attributes at the top level to describe the decomposed attribute. OPM's *detail decomposition* is contrasted with UML's *aspect decomposition*. UML decomposes the structure, behavior, and states separately while incorporating the concrete, detailed, and abstract levels of abstraction within the previous three decompositions. OPM looks at a given level of abstraction and combines the structure, behavior, and states aspects. Dori also comments on multi-model methods such as UML by noting that "a complete mental picture of the system needs to be created from information that is distributed across multiple views with

different graphical syntax." OPM has a single model while UML uses many models for the different parts.

4.3.1 OPM Limitations

RAAM adds to the object/process breakdown by including the computation of the model into the description of the system. The Object Process Methodology does not address the computational model used for modeling the system of interest on a computer. The Object Process Methodology was not designed for and is ill suited for modeling architectures with different structures from the same base description.

4.4 Summary of Gaps

System of systems architecture modeling has room for improvement. Multiple groups are calling for ways to improve system of system architecture analysis. The analysis takes too much time, in terms of both computational and analyst time. It is currently not feasible to analyze large enough portions of the system of systems architecture design space. The process must be traceable and rigorous. Ideally, the solution should fit with existing Department of Defense frameworks.

Others, such as Mercer [105], have noted that executable architectures are desired for defense related architectures. He believes that, "executable architectures are unlikely to be developed within the purview of DoD architecting until the foundations of such practice become more formal and scientific in nature." The subsequent RAAM research hopes to address those concerns. In addition, Sage [125] wrote that often problem solutions are only recommended at the level of the symptoms and not at higher levels. A defense planner needs a way to move between different levels of scope.

Computing trends indicate that any new analysis methods should take advantage of parallel processing. We will be able to analyze more systems and in more detail.

Dori in his Object Process Methodology book [58] remarks, "Complexity is inherent to real life systems" and later, "An integral part of a system development methodology must therefore be a set of tools for controlling and managing this complexity". He continues, "the need for systems analysis and design strategies stems from complexity. If the systems or problems were simple enough for humans to be grasped merely by glancing at them, no methodology would be required." The research addresses not only the complexity issue but also includes a methodology to address it.

A system of systems architecture can be extremely complicated which can lead to its downfall. The Future Combat Systems (FCS) program is just one example. The Future Combat Systems (FCS) program can be considered a "system of systems of systems." [28]. More integration of systems into a system of systems will be the norm for the future. We need a way to deal with the complexity of such systems if we are going to be able to design and deploy system of systems.

CHAPTER V

RESEARCH QUESTIONS AND HYPOTHESES

There is a hierarchical nature to how the research objective is connected to research questions, hypotheses, and experiments. To organize the research questions, hypotheses, and experiments, the manuscript starts with one of each that addresses the framework and methodology. The next three, numbers 2-4, of the research questions, hypotheses, or experiments are subordinate to the first one. The subordinate research questions, hypotheses, and experiments are designed to address the capabilities of RAAM. For a visual representation of the research objective, see Figure 8.

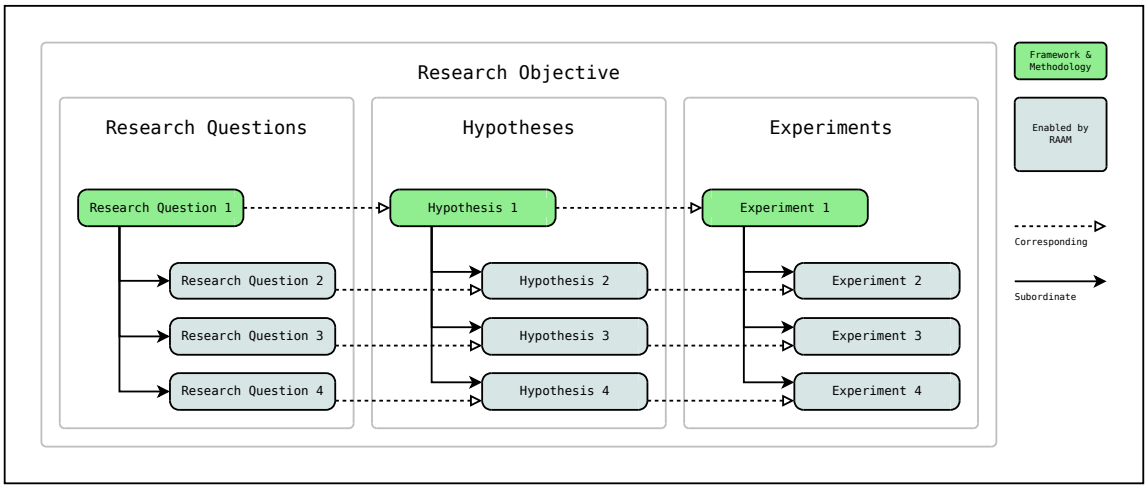


Figure 8: Hierarchical Decomposition of the Research Objective

5.1 Research Objective

The previous discussion allows for the creation of a research objective to guide the work. The following research objective will serve as the overarching goal behind this thesis.

Develop a new methodology for compactly describing and evaluating architecture alternatives. This will improve capability based conceptual analysis by reducing runtime and model creation complexity as compared to existing executable architecture models with limited to no fidelity loss.

Capability based conceptual analysis is improved by reducing both computer and human analysis time while reducing the complexity of the analysis. The improvements will be enabled by a compact method for describing the architecture alternative space. There is a push toward more analysis in the defense architecture analysis community. System of systems analysis by definition involves many systems which typically incorporates multiple services in joint operations. As funding is reduced for the military, there is a greater need for traceability to support the selection of an architectural alternative. Traceability for a system of systems architecture selection requires analyzing more alternatives than ever before. The Department of Defense is struggling with the current analysis burden. The current research is searching for ways to analyze more alternatives with fewer resources, both runtime and analyst time. Early phase architecture design does not need high fidelity analysis, but only needs enough fidelity to make the same decisions as would have been made with higher fidelity results.

To achieve the research objective, the manuscript focuses on three main elements:

- System and Operational Level Trades
- Generation and Execution of Architecture Models
- Data Manipulation

5.2 Research Questions

The research objective leads to a series of research questions which when answered should lead to the success of the research objective.

Portfolio analysis of system of systems architectures uses metrics of interest that invariably include performance, schedule, and cost. As noted previously, current techniques used by the DoD do not allow for the full exploration of the system of systems architecture portfolio design space. Full exploration of the design space entails examining every possible architecture alternative. Both system level and operational level trades must occur. A system level trade is one where the decision maker is trying to decide which system should accomplish a given task. An example would be when a decision maker is interested in choosing between using an F-16 or an F-18 for a combat air patrol. An operational level trade is when a decision maker is trying to decide which set of subtasks should accomplish a given task. The operational level trades are different ways of decomposing the tasks into subtasks. The down selection to one system of systems architecture portfolio occurs before the Milestone A decision for the system of systems or the component systems. The answer to the first research question addresses the need to develop a new methodology for system of systems architecture analysis.

Research Question 1:

How can an analyst use system and operational level trades at the same time to model the cost and performance of a set of system of systems in support of Pre-Milestone A decision Making?

An often stated "want" from Department of Defense customers is the ability to analyze both operational and system level trades. Analyzing both at the same time is difficult because of the combinatorial explosion of alternatives. The proposed framework should be compatible with existing early phase analysis.

The models used for performance, schedule, and cost are not very useful without a set of alternative system of systems architecture portfolios to evaluate. The number of potential system of systems architecture portfolio alternatives can be staggering and infeasible to generate one at a time. The answer to the second research question will enable the reduction in analysis time and a reduction in analyst cognitive load.

> **Research Question 2:** *How can an analyst automatically generate system of systems architecture alternatives?*

Since the number of alternatives can approach billions and trillions of options, humans can not manually generate each alternative. A computer algorithm for generating architecture alternatives should be developed to automate this process. The current methods are typically applied to problems that do not have operational level decisions but they do have system level decisions. In that case, a matrix of alternatives with compatibility constraints is sufficient.

Once we have a set of system of system architecture portfolio alternatives, we need to be able to utilize computers to compute the performance, schedule, and cost. Creating executable models can be a time consuming process so automation will increase the number of alternatives evaluated. The answer to the third research question enables a reduction in computer and analyst time spent.

> **Research Question 3:** *How can an analyst make executable models for system of systems architectures?*

Current executable models are typically created on a per-architecture basis, for example when using DoDAF. The creation of the models is not automated, and there are often restrictions on the type of metrics that can be used. For example, a discrete event simulation (DES) model of the architecture may be created from architectural projects. The discrete event simulation would not be useful to calculate the cost of the architecture. A separate cost model would need to be developed.

The set of executable models of alternative system of systems architecture portfolios must be able to run efficiently on available computing resources. Due to the large problem spaces from system of systems analysis, data storage and manipulation becomes an issue. The answer to the fourth research question will address strategies for coping with the data deluge.

> **Research Question 4:** *How can an analyst deal with the data storage and manipulation requirements of a complete system of systems analysis?*

The data storage and manipulation problem is realized once executable models become fast enough to explore the full design space. Strategies to accomplish filtering of the design space will need to be developed. Retrieving the architecture alternative information from storage may be slower than recalculating the score. Recommendations are made for visualization techniques to help with the large data sets.

5.3 Hypotheses

The research questions naturally lead to formulating hypotheses that attempt to answer the research questions.

The first research question requires a method to take into account both system and operational trades. To keep complexity of the analysis low, the operational and system level trades will be unified into the same type of trades. RAAM was developed to address the gaps in current architecture modeling frameworks for the system of systems architecture alternative analysis problem. By decomposing the system of systems architecture description into partial descriptions, the system and operational level trades will be able to be represented in the same manner. The partial descriptions allow for a compact representation of the dependencies that arise due to operational decisions.

> **Hypothesis 1:** *By modeling system of systems architectures using partial descriptions that are assembled from architectural decisions, different*

systems and variable operational structures both are modeled in a unified way that allows for supporting Pre-Milestone A decision making.

The second research question addresses a problem with complex system of systems architecture portfolios. Mapping a system to a specific task will become tedious for thousands of system to task mappings. When adding operational trades, generating architecture alternatives becomes infeasible for humans with hundreds of millions or billions required. An automatic way of generating feasible alternatives is required to reduce analysis time and analyst cognitive load. Due to the magnitude of the number of feasible alternatives, it is not efficient to generate all of the alternatives at the same time. The generation of alternatives must be done in a streaming fashion that generates the next alternative on demand.

Hypothesis 2: *Separating the architecture alternative description into partial descriptions and architecture decisions allows for the automatic generation of alternative architectures*

The third research question is concerned with executing the system of systems architecture portfolio models on a computer. The performance, schedule, and cost of a candidate system of systems architecture portfolio is modeled by aggregating the performance, schedule, and cost of its constituent systems. The different operations used for aggregation and the structure of the task hierarchy must be taken into account. Both computer and analyst time will be reduced with the automatic generation of executable system of systems architecture models. The current methods for making executable models from an architecture description are limited in the metrics that they can evaluate. In addition, the different computational modeling methods for the architecture alternative will require different information for the different models. An execution method that creates a common framework for creating computational

models is used to unify the different types of models that may be used in early phase design.

> **Hypothesis 3:** *Automatically created executable models are made feasible by using a system of systems architecture described by aggregations and transformations.*

The fourth research question arises due to the efficiency of the generated executable models. By enabling the exploration of the entire design space, the new data generated is massive. This research question is concerned with the large quantities of data that result from analyzing all of the architecture alternatives. Traceability is improved by exploring the entire design space.

> **Hypothesis 4:** *Within selected portfolios, by first examining a portfolio centric view and then looking at a system to task view, the data storage and manipulation requirements become feasible.*

A portfolio centric view considers all architecture alternatives with the same systems to be the same portfolio. By looking at the portfolio view and then only exploring the data from a select set of portfolios the analyst's problem becomes tractable. The portfolio view is used as a filter to summarize regions of the possible decision space.

5.4 Elements of RAAM

RAAM is composed of three different elements that answer the research questions and hypotheses. The different elements are shown in Figure 9. RAAM is composed of the methodology, syntax, and implementation. The RAAM methodology describes how to approach capability based analysis of system of systems architectures. To enable the analysis of the entire decision space that is available to the decision maker, a syntax was created to represent the concepts that are derived from the methodology. Once a methodology and syntax exist, an implementation is described which uses

both to generate and evaluate all of the architecture alternatives in the system of systems analysis problem.

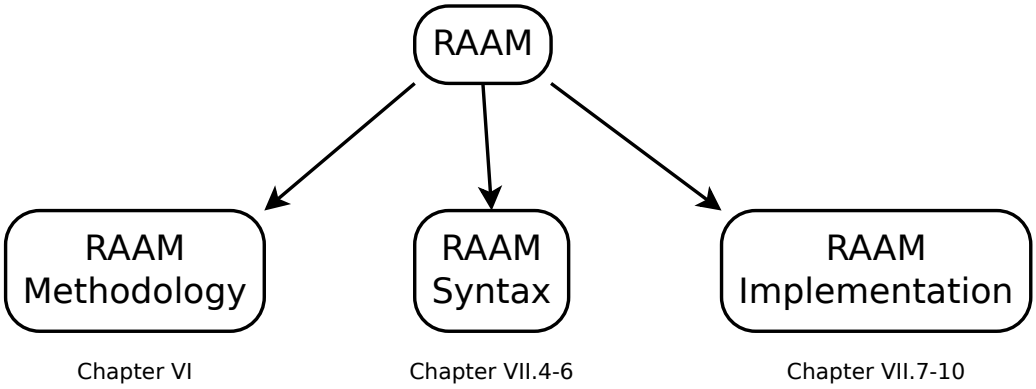


Figure 9: Three Elements of RAAM

CHAPTER VI

METHODOLOGY

> Simplicity does not proceed complexity, but follows it.
>
> Alan Perlis [119]

The Committee on Pre-Milestone A System Engineering recognized that, "The prerequisite for starting the systems engineering process is a user-defined need (or outcome)." The following methodology focuses on the Concept Creation, Performance Assessment, and Architecture Development functions identified as critical Pre-Milestone A systems engineering functions. The methodology focuses on the Supporting Models which are a critical output. [28]

The proposed methodology builds on the basic system engineering methodology described by Sage. [125] He notes three main steps to any system engineering analysis effort: formulation, analysis, and interpretation. An effective methodology, "must be capable of dealing with both quantitative and qualitative criteria representing costs and effectiveness from their economic, social, environmental, and other perspectives." The methodology enables the creation of models to determine the cost and effectiveness of a system of systems architecture.

Coordinated analysis is made possible with a common reference point. [17] The RAAM methodology provides a common reference point in the task hierarchy. This allows for analysis of the system of systems architecture across multiple metrics.

6.1 Methodology

The methodology begins with a set of required capabilities that are derived using the previously discussed methods. The required capabilities are inputs to the methodology. The methodology has six main steps with four in the formulation stage and one each in the analysis and interpretation stages. The output of the methodology is a portfolio of systems to procure that address the required capabilities. The methodology is summarized in Figure 10. The six steps are:

1. Determine Required Derived Capabilities
2. Create Capability Hierarchy
3. Define Candidate Systems
4. Define System of Systems Computer Models
5. Analyze Potential System of System Architectures
6. Determine Optimum Portfolio

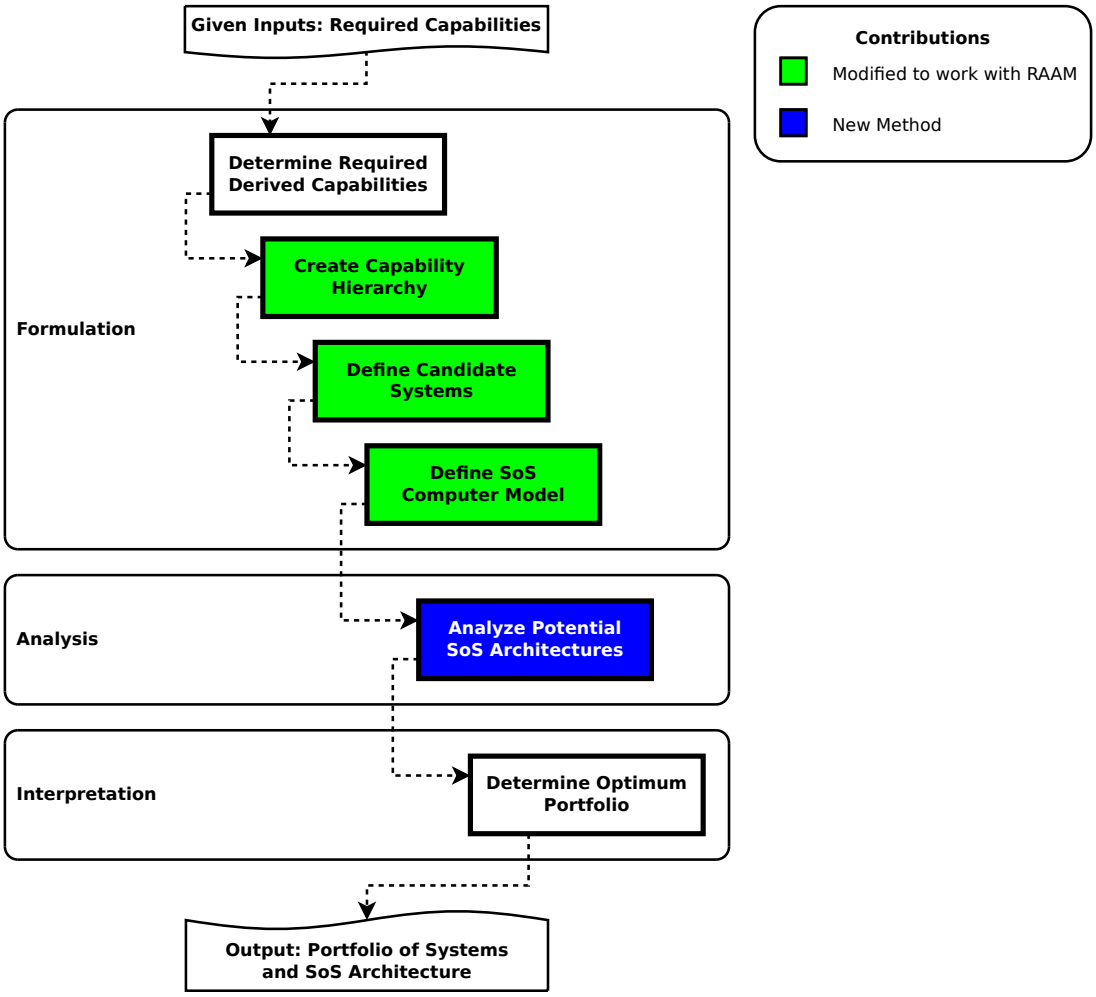


Figure 10: RAAM Framework and Methodology

As stated above, steps 1-4 (Determine Required Derived Capabilities, Create Capability Hierarchy, Define Candidate Systems, and Define System of Systems Computer Model) are part of the formulation step. In the formulation step the analyst is defining the scope of the analysis. Step 5, Analyze Potential System of Systems Architectures, is part of the analysis step. The analysis step is used to create new information about the system of systems of interest. Determine Optimum Portfolio, step 6, is part of the interpretation step. In the interpretation step the analyst is making sense of the generated information and data from the analysis.

From the high level required capabilities the first step is to **Determine Required Derived Capabilities**. Once the problem that the analyst is trying to solve is understood they can **Create Capability Hierarchy**. With the set of tasks to accomplish being known, the analyst needs a set of systems to evaluate within the capability hierarchy, so they **Define Candidate Systems**. At this point, the analyst has to determine how to model the different metrics of interest, so the analyst has to **Define System of Systems Computer Model**. Those four blocks finish the formulation stage of the process.

The analysis stage of the process is composed of one block, **Analyze Potential System of Systems Architectures**. The analysis will allow for the generation of metrics of interest that include metrics on performance, schedule, and cost. The metrics of interest feed into the next stage.

After the analysis of the different system of systems architectures, we must compare the performance, schedule, and cost. The interpretation stage is composed of one block, **Determine Optimum Portfolio**. In the Determine Optimum Portfolio block is where decision support activities occur.

6.1.1 Determine Required Derived Capabilities

The Determine Required Derived Capabilities block is where the analyst takes the required capabilities and figures out what the derived capabilities are. The current methods for achieving the list of derived capabilities is sufficient for the purposes of the research. The derived capabilities are the capabilities that the system of systems architecture portfolio will be responsible for delivering.

When beginning a system of systems architecture analysis study, the analyst will have a required capability in mind. To achieve the required capability there may be derived capabilities that enable the required capability. Each derived capability will have its own capability hierarchy. The different capabilities can share tasks and

systems although the tasks and systems may be unique to a capability. As part of determining the required derived capabilities, the analyst must determine the metrics used to evaluate the performance of a capability. Both mission dependent and mission independent metrics are associated with a capability.

The proof of concept example is the Suppression of Enemy Air Defenses mission. In that case, there is only one derived capability and it is to provide Suppression of Enemy Air Defenses to the combatant commander.

6.1.2 Create Capability Hierarchy

Once the required capabilities are described and understood, a capability hierarchy is created. The capability hierarchy is similar in concept to the Capability Taxonomy (CV-2) from DoDAF, among others. It is a hierarchical description of tasks that are required to accomplish a capability.

The Create Capability Hierarchy block is the first step in the methodology that will require changes from the current state of the art. The current methods for describing a capability hierarchy tend to be focused on describing a single architecture alternative. RAAM requires describing the entire architecture alternative space. Therefore, RAAM requires a new way to specify the capability hierarchy.

The capability hierarchy is a decomposition of the required tasks to achieve a specified capability. The required tasks are the 'ways' that are discussed in the JCIDS definition for a capability. The root of the hierarchical tree is the task that provides the desired effect. Subordinate to the root is a variety of tasks that are required to achieve the root task. Each of those tasks can be further decomposed. This is shown in Figure 11. The decomposition continues until the analyst is satisfied with the level of detail. The hierarchy does not have to have the same depth throughout. Analysts can add or remove detail as is appropriate for different parts of their problem.

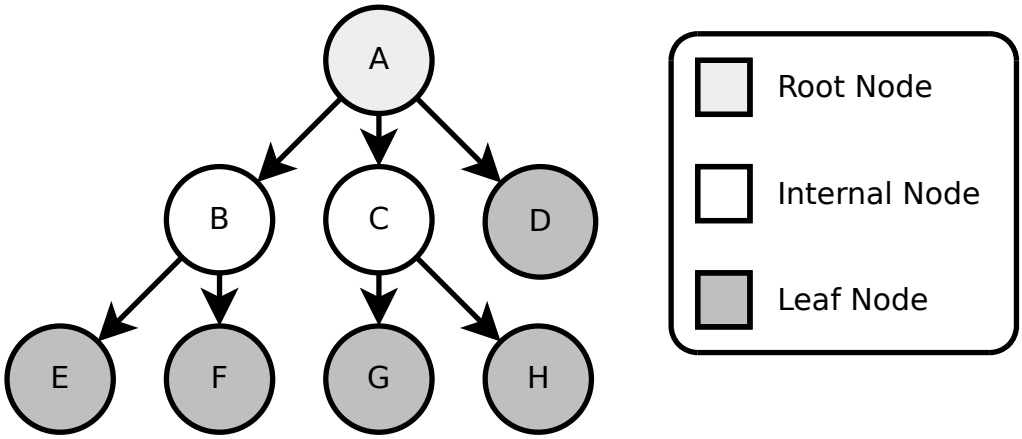


Figure 11: Root, Internal, and Leaf Node Relationships

The capability hierarchy is described in the RAAM input file. The initial description is often created using visual tools although text based formats are more precise and may be simpler. The capability hierarchy is defined in pieces that are referred to as partial descriptions. The partial descriptions describe the all of the possible task hierarchies that accomplish the capability. In this way, the architecture alternative design space can be fully specified.

The Capability Taxonomy (CV-2) is a DoDAF viewpoint that presents a hierarchy of capabilities. The viewpoint is often used to show both current and future capabilities. A capability taxonomy does not describe how a capability is implemented. The same terminology of using root and leaf nodes is used in DoDAF as in RAAM. In this case, a system provides the ability to accomplish the leaf tasks of the capability hierarchy. The capability taxonomy can have a measure defined for the leaves, but does not map a system to the measure. The Capability Taxonomy has no mandated structure beyond supporting the representation as a hierarchical list. Text, tables, or graphics may be used. [146]

6.1.3 Define Candidate Systems

Once the capability hierarchy has been created, the different possible systems are enumerated. In this case, a system provides the ability to accomplish the leaf tasks of the capability hierarchy. The Define Candidate Systems block is modified from current best practices to fit into the RAAM methodology. The essential information about a system is gathered in this step.

Each system that can accomplish one or more tasks from the capability hierarchy is documented. Mission independent and mission dependent metrics are defined at this stage. Mission independent system attributes, for example cost or schedule, are documented along with the list of systems. The mission dependent metrics scores are defined in a triplet uniquely specified by a system, a task, and a metric. A system can do a certain task which results in a specific metric score. Each task must have a system that accomplishes the task although multiple systems can accomplish a single task. For each metric each system to task pairing is scored by subject matter experts or computational models.

Rather than define a matrix of systems to tasks to describe the system to task mappings, only the valid pairings are recorded. This reduces the burden of data collection. For example, if there are eight systems and twelve tasks, gathering the data in a matrix form requires asking for eight times twelve, or ninety six different scores per metric. If each system only is capable of doing three of the twelve tasks, there are only eight times three, or twenty four data points to gather per metric.

6.1.4 Define System of Systems Computer Model

The last block in the formulation stage is the Define System of Systems Computer model. The analyst needs to have a way of using the data collected in the previous steps. The Define System of Systems Computer Model step of the methodology is where the analyst describes how to combine the system and task information to

provide architecture metrics from the lower level system metrics.

For each task and metric pairing, a computational model is chosen. The computational model is a combination of the way to aggregate the sub-tasks and a transformation that scales the result of the aggregation. For the tasks that are at the leaves of the tree, the computational model is chosen to return the score of the associated system. The leaf computational model can also be a distribution if you are doing probabilistic analysis.

The computational model converts the system level scores into architectural alternative level scores. A different computational model is defined for each metric. Just as with the capability hierarchy step, the computational model is defined in a piece wise fashion. Each task has an associated aggregation and transformation function. A computational model that can compute the score of the entire architecture alternative is created from the relevant task computational models.

6.1.5 Analyze Potential System of Systems Architectures

The formulation stage of the methodology describes a set of system of systems architectures with enough information to proceed with the analysis of alternatives. The Analyze Potential System of Systems Architectures block generates the performance, schedule, and cost information about each system of systems architecture.

The description provided by RAAM is detailed enough for the analysis block. The numerous alternatives are automatically generated and filtered for interoperability and valid system to task mappings. Each alternative is converted automatically into an executable model that will produce the metrics of interest for the system of systems architecture.

An executable model for each metric is created for each architecture alternative from the partial descriptions of the task mappings, the system to task information, and the task computational models. With RAAM it is possible to gather the scores

of every architecture alternative rather than exploring limited portions of the design space. The analysis step is automated, with both the generation and evaluation of an architecture alternative done by a computer. The resulting architecture alternative metric scores are used in the last step of the methodology.

6.1.6 Determine Optimum Portfolio

With all of the metrics of interest generated for a set of valid system of systems architecture alternatives, it is possible to then move to the Determine Optimum Portfolio step. The Determine Optimum Portfolio block utilizes current methods for multi-objective decision making (MODM) and visualizations of alternatives.

The different metric scores for each alternative can be combined using an overall evaluation criterion. If creating a single value is too limiting to the decision makers, a decision support environment should be created to explore the design space. The volume of data that can be created causes issues for decision support environments. A filtering method is normally required to pare down the decision space to a manageable size for human comparison.

The architecture alternative metric scores can be compared with a variety of visualizations. Bar charts, pie charts, and radargrams provide summary visualizations. A scatter plot matrix can be used in an interactive environment to show relationships between the metrics and to find patterns in the design space.

The outcome of the methodology is a portfolio of systems to procure with the operational decisions. It is not often the case that an entire system needs to be designed from scratch, so in most cases it only the new systems in the portfolio will need to be acquired. If there will be heritage systems in the system of systems, information about how to upgrade those systems will also need to be available.

An example decision support environment is shown in Figure 12. The environment has a scatter plot matrix with the metrics of interest and the number of systems and

the number of alternatives. The data is aggregated into a portfolio viewpoint, which is discussed later in the document. Each point in the scatter plot matrix is an aggregation of all architecture alternatives that have the same portfolio of systems. The decision maker can color the scatter plot matrix based on a metric or they can filter the alternatives based on what is included in the portfolio. Both colorizing features allow for the customization of the visualization. An Overall Evaluation Criterion is often used to combine the different metric scores of the architecture alternatives. The OEC function in this case is dynamically calculated and the weightings are changed on the display.

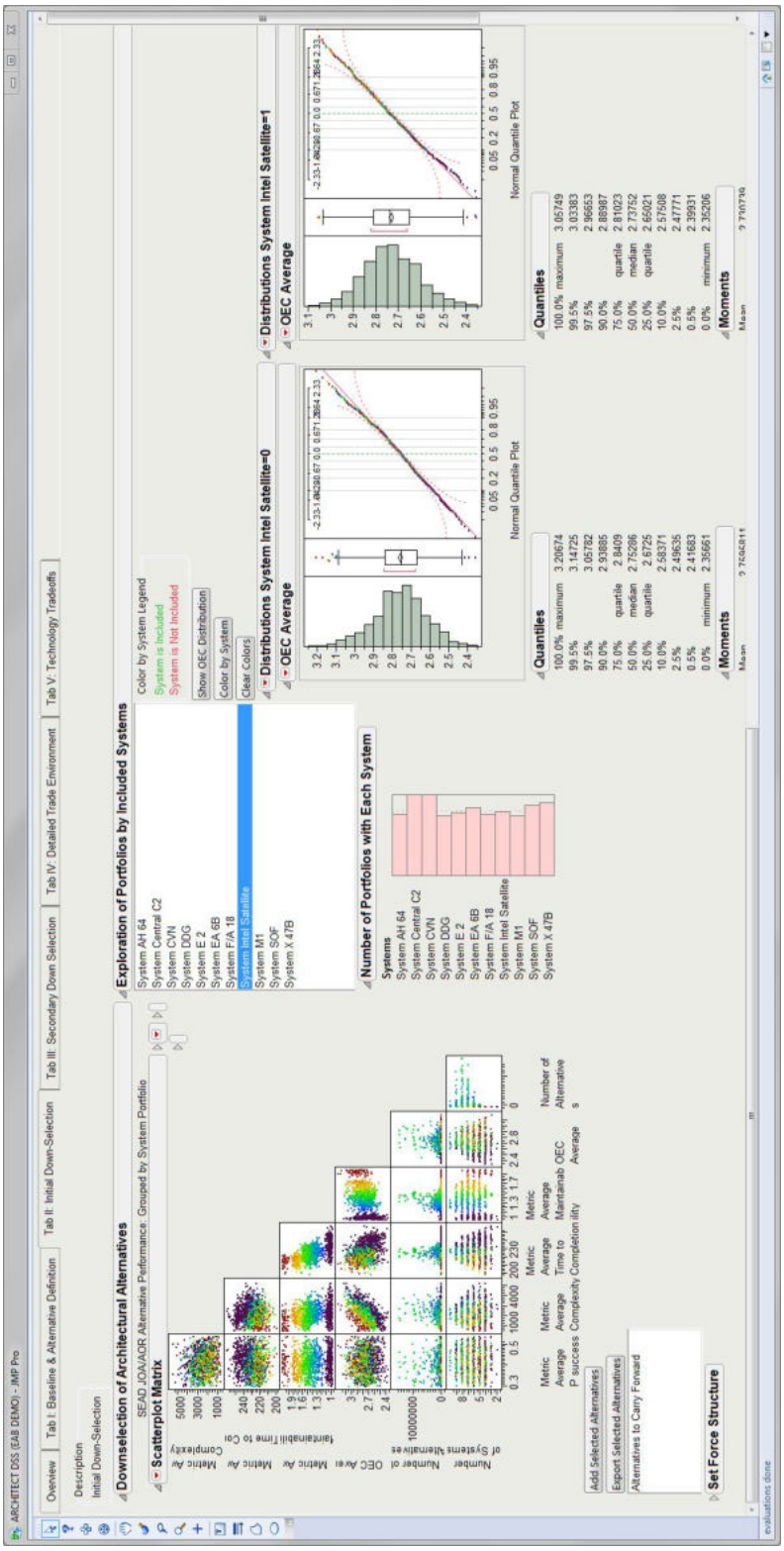


Figure 12: Decision Support System Example Layout

CHAPTER VII

THEORY AND FORMULATION

> However, a higher level language than LISP might have such a large declarative component that its texts may not correspond to programs. If what replaces the interpreter is smart enough, then the text written by a user will be more like a declarative description of the facts about a goal and the means available for attaining it than a program per se.
>
> John McCarthy [103]

7.1 Formal Description of the Problem

The RAAM methodology is designed for a specific use case which is the early phase of conceptual design of a system of systems. As system of systems architectures are not normally designed and implemented in one fell swoop, the analyst is often tasked with looking at evolutionary options to a system of systems architecture. The customer and stakeholders are interested in reuse of existing assets whenever possible in order to accomplish the mission.

The RAAM methodology takes a specific viewpoint on the system of systems architecture alternative problem. The viewpoint can add constraints on the types of analysis that is able to be performed. Many types of analysis are possible with the methodology, but must be converted to a form amenable to the mental and executable model. Since the RAAM methodology is designed to be used in early phases of conceptual design (Pre-Milestone A), the limitations still allow for enough fidelity to make early decisions. More detailed models that produce higher fidelity estimates of the metrics of interest require higher quality input data. The higher

quality input data requires decisions to be made about the architecture configuration that are not often made at this early stage in system of systems architecture design.

Two types of system of systems problems are addressed with RAAM. The first of these problems is determining the value of a metric for a specific system to task allocation(system centric view). The other type of problem that can be addressed with RAAM is determining the value of a metric for a system portfolio (portfolio centric view) that can accomplish a set of tasks. Depending on the problem, the analyst will be interested in one or both views on how to use RAAM. Determining the value of a metric for a system portfolio accomplishing a set of tasks is determined by combining the scores from determining the value of a metric for a specific system to task allocation. The system centric view is a model where a system does a specific set of tasks in support of the architecture alternative. The portfolio view is a model where the systems in the portfolio are static, but the allocation of system to task can vary during use.

Table 1 shows the notation used in this section. There are n total systems within a portfolio for a given architecture alternative. A system is in S, and a given system is identified with the index i, so a system is S_i. The architecture alternative has m tasks. A task is in T, and a given task is identified with the index j, so a task is T_j. Leaf tasks are identified with L_j, and are also in T. A leaf task can have systems allocated to it.

Table 1: Symbols Used

Symbol	Description
n	Number of Systems
m	Number of Tasks
S_i	System i
T_j	Task j
L_j	Leaf Task j

The analyst is interested in determining which allocation of systems to tasks provides the 'best' portfolio for achieving a capability. The 'best' architecture alternative is determined by examining the values of metrics of interest for the architecture alternative providing the capability. The metrics of interest are estimated by creating computational models based on a task hierarchy and systems used to accomplish the capability.

A capability is decomposed into a task hierarchy that is composed of tasks, $T_1, \cdots, T_j, \cdots, T_m$. The leaf tasks, L_j are included in the list of tasks ($L \subseteq T$). A portfolio of systems, $S_1, \cdots, S_i, \cdots, S_n$, provides the capability by accomplishing the tasks. In this model, a one system S_i accomplishes one task L_j. A system from the portfolio can accomplish more than one leaf task. A different architecture alternative is formed when a different system accomplishes a given leaf task. A non-leaf task must have subtasks that are required to accomplish the non-leaf task. Just as a leaf task can have multiple possible systems that can accomplish the left task, a non-leaf task can have multiple possible sets of subtasks that can accomplish the non-leaf task. When calculating the value of a system accomplishing a task, the value is uniquely determined with a system to task pair. Not all pairs of system and task are possible (i.e., it is not a full Cartesian product).

7.2 Aggregation and Transformation

The work was heavily influenced by RAND Corporation's Portfolio Analysis Tool [37] which was detailed in Section 3.5. The Portfolio Analysis Tool comes with five built-in aggregation methods with the ability to extend the tool to use additional aggregation methods. The aggregation methods are used to combine the scores of lower level tasks into the score of a higher level task. These methods will be expanded upon in the subsequent sections. Aggregation methods combine the values of the subtasks of a task. A leaf task results in the value of the system that accomplishes it. The five

methods use the concepts of **thresholds**, **goals**, and **non-linearity**.

The RAAM implementation currently has the five aggregation methods defined by the Portfolio Analysis Tool available. The concept of aggregation and transformation were combined in the Portfolio Analysis Tool, but this work separates the two concepts. By separating the Portfolio Analysis Tool's aggregation concept into aggregation and transformation, the analyst has more flexibility in choosing the computational model and the separation promotes greater reuse of functions between or within computational models. Aggregation is the combination of multiple sub-task scores to provide a score for the task. Transformation is a one to one function that essentially is used to scale or clip the resulting task score. In addition, the Portfolio Analysis Tool specifies different computations between leaf and inner nodes (non-leaf tasks) of the hierarchy. RAAM unifies the computation of a subtask score and a system score so they are handled in the same way in the execution environment. By unifying the computations of the leaf nodes with the inner nodes, the implementation can become more flexible and allow for task hierarchies with different depth levels along different branches.

7.2.1 Threshold Method

The threshold method is used to determine the score of a task from its subtasks. The threshold method from the Portfolio Analysis Tool returns zero if any of the subtask scores are less than a specified threshold value. If any subtask scores are above a specified goal value then the aggregation function returns a specified maximum value. Otherwise, the aggregation function returns a weighted sum of the subtask scores. This aggregation function is used when the task fails if any of the subtasks fail. In RAAM, the threshold aggregation from the Portfolio Analysis Tool is represented with a weighed sum aggregation function with a transformation function to provide the necessary clipping. The transformation function is detailed in Figure 13 and with

Equation 1. The aggregation function is defined in Equation 2.

$$T(x) = \begin{cases} 0 & \text{if } x < threshhold \\ goal & \text{if } x > goal \\ x & \text{otherwise} \end{cases} \tag{1}$$

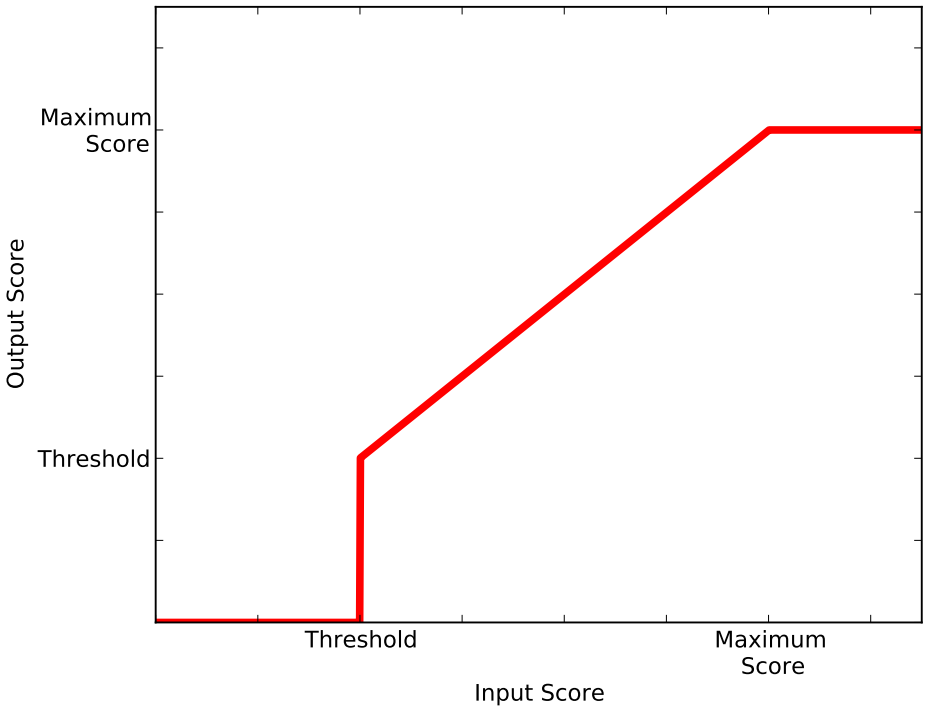


Figure 13: The Threshold Transformation Function

$$A(x_1, \cdots, x_n) = \begin{cases} 0 & \text{any } S_i < G_i \\ \frac{\sum_{i=1}^{n} W_i S_i}{\sum_{i=1}^{n} W_i G_i} & \text{otherwise} \end{cases} \tag{2}$$

7.2.2 Weak Threshold Method

The weak thresholds method is similar to the thresholds method. The difference is that the aggregation does not result in zero if a subtask value is below its threshold.

This is shown in Equation 3. The transformation function is the same as in the threshold method.

$$A(x_1, \cdots, x_n) = \frac{\sum_{i=1}^{n} W_i S_i}{\sum_{i=1}^{n} W_i G_i} \tag{3}$$

7.2.3 Weakest Link Method

The weakest link aggregation function returns the minimum value of the subtask scores. This aggregation function is often used when the score of a group of tasks or systems depends on the weakest link. Many system of systems architectures have tasks with weakest link properties. Related to the weakest link is taking the maximum value of the subtask scores. The maximum can be used to simulate the time to complete parallel processes, the longest time from the subtasks should be used as the time to complete the parallel tasks. The equations are shown in Equation 4. The weakest link method normally is not transformed with a transformation function. The transformation function is the identity function in that case.

$$\begin{gathered} A(x_1, \cdots, x_n) = min(x_1, \cdots, x_n) \\ or \\ A(x_1, \cdots, x_n) = max(x_1, \cdots, x_n) \end{gathered} \tag{4}$$

7.2.4 Goals Method

The goals method provides a way to require a minimum performance out of each individual task. Each subtask score must meet a defined goal or its value becomes zero. The aggregated score with the goals method is found by a weighted sum of the subtasks divided by a weighted sum of the ideal situation where all of the subtasks meet their goals. This is detailed in Equation 5, with goals (G_i), subtask scores (S_i), and weights (W_i). The goals method is used when the analyst would like to sum the

fractional contributions of the subtasks.

$$A(x_1, \cdots, x_n) = \frac{\sum_{i=1}^{n} W_i S_i}{\sum_{i=1}^{n} W_i G_i} \tag{5}$$

7.2.5 Rankings Method

The fifth method from the Portfolio Analysis Tool is the rankings method. The different options for a subtask are simply regarded as being ranked from best to worst. The aggregation function is described in Equation 6. The ranking method is used to provide a task score that is the result of the qualitative ranking of the subtask scores. This method can be chosen when the scores are qualitative in nature.

$$A(x_1, \cdots, x_n) = \frac{\sum_{i=1}^{n} W_i R_i}{\sum_{i=1}^{n} W_i} \tag{6}$$

7.3 Domain Specific Languages

van Deursen, Klint, and Visser [147] concisely explain that a domain specific language as, "a small, usually declarative, language that offers expressive power focused on a particular problem domain". Formally, they define a **domain specific language** as:

> A *domain-specific language* (DSL) is a programming language or executable specification language that offers, through appropriate notations and abstractions, expressive power focused on, and usually restricted to, a particular problem domain.

In addition to domain specific languages, subroutine libraries and object-oriented/component frameworks are used to solve problems in specific domains. Domain specific languages can be embedded into a general purpose language to utilize the full power of the host general purpose language. In general, domain specific languages are declarative and

must be interpreted or compiled to create applications. Summarizing the literature, van Deursen, Klint, and Vissser cite six benefits of domain specific languages:[1] [147]

1. Expression of the problem is natural to the domain which allows for subject matter experts to understand, validate, modify, and develop applications
2. The program source is concise, self-documenting, and reusable
3. Productivity, reliability, maintainability, and portability are enhanced
4. Domain knowledge is captured and can be shared
5. Validation and optimization are available at the domain level
6. Testability is improved

They also cite six disadvantages of domain specific languages that should be considered:

1. Designing, implementing, and maintaining a domain specific language can be costly
2. Education of users can enact a cost
3. Domain specific languages often have limited availability
4. Proper scope can be difficult to establish
5. Properly designing a domain specific language for balance between domain-specificity and general-purpose constructs
6. There is a potential for a loss of efficiency

[1]Anyone interested in further information about domain specific languages would do well to investigate the references in [147]

The idea of a domain specific language is not new. Hundreds of domain specific languages exist across a multitude of domains. van Deursen, Klint, and Vissser note PIC, SCATTER, CHEM, LEX, YACC, SQL, BNF, and HTML as domain specific languages. They summarize the development of a domain specific language into a three step process; analysis, implementation, and use. In the analysis phase, first, the problem domain is identified. Next, the relevant data is gathered about the problem domain. Further, the knowledge within the data is clustered into a set of semantic notions and operations. Lastly, the domain specific language is designed in a manner which concisely describes the application. In the implementation phase, a library is created that implements the semantic notions. A compiler is created which translates the domain specific language into calls to the library. In the use phase, the domain specific language is used to create useful applications which are compiled. [147]

Domain specific languages have been used to capture design synthesis knowledge for model based systems engineering. [91] van Deursen, Klint, and Vissser use graph based transformations to convert the systems engineering models into a model for a specific design alternative. They argue that general purpose models such as SysML can be inconvenient and less effective than using a domain specific language although SysML is used as an integration framework. A domain specific language must be created for each modeling effort in their work. The work creates a decision graph which can be traversed to generate a concrete analysis model. They use an evolutionary program to search the design space. Other efforts, such as Rosetta are designed to provide a declarative, heterogeneous, and formal domain specific language for system level design. [3] The Rosetta Language Reference Manual has five major language subsystems:

- Type System
- Expression Language

- Facet Language
- Reflection subsystem
- Domain Library

Model Driven Development promotes domain specific languages and converting the DSLs to executable models. [81]

Many domain specific language implementations are implemented by extending a base language. The three common ways to extend the base language are embedded languages/ domain-specific libraries, preprocessing or macro processing, or through an extensible compiler or interpreter. The following work utilizes a preprocessing/ macro processing approach to make the best use of the language runtime. The advantage is simplicity whereas the disadvantage is that feedback to errors are provided at the base language level. If the domain specific language is used correctly, it will not be an issue. If the language is not use correctly, however, the analyst will require knowledge of the base language.

The previous chapter discussed the new methodology which depends on a domain specific language called RAAM. The RAAM and the methodology were designed together and it is hard to define exactly where one begins and the other one ends. How one can use RAAM is the methodology while the semantics and syntax are part of the definition of the RAAM domain specific language and execution engine.

In the design of RAAM, the following tips from Steele[2] were used. [132] He discusses five "big messages" for future language designed to utilize future computing hardware:

1. Effective parallelism uses trees.
2. Associative combining operators are good.

[2]Guy L. Steele Jr. was involved with the design or standardization of Java, Scheme, Common Lisp, C, Fortran, Fortress, and Javascript.

3. MapReduce is good. Catamorphisms are good.

4. There are systematic strategies for parallelizing superficially sequential code.

5. We must lose the "accumulator" paradigm and emphasize "divide-and-conquer".

Tree like structures are appealing for parallelization because of the independence of branches of the tree. The different branches are linked through the roots of the tree, but sub-branches may be computed separately and combined at a later time. This allows for effective parallelization. In addition, by structuring the computation as a tree, the actual computation can be done serially or in parallel. A tree can be divided into a depth-first parallelization strategy or a breadth first parallelization strategy. Both are graph traversal algorithms. Distributed versions of depth-first and breadth-first traversals were documented in the early 80's. The breadth-first algorithm generates a shortest path spanning tree, but this property is not guaranteed to hold in distributed versions. [23] Many algorithms have been developed to use parallel computation in a breadth first manner on trees. For breadth first searches, efficient distributed breadth-first search algorithms have been developed. [100] [101] [21] A breadth first traversal visits all of the nodes at the same level before moving down to the next level. Image processing can be turned into a breadth first search. [130] The flexibility of trees allows for the structure of the parallelism to be decided at runtime and specialized to the current hardware.

Associative combining operators help with parallelization. An associative combining operator allows for computation to be split apart at any combining location and it still reaches the same result. Addition and multiplication are associative operators. For example, $1 + (2 + 3) = (1 + 2) + 3$. For an associative combining operator $*$ and set S, Equation 7 holds. Since the parentheses can be moved around the equation, different parts of the computation can be done at the same time to enable parallelization.

$$(x * y) * z = x * (y * z) \text{ for all } x, y, z \in S \tag{7}$$

Algorithms such as MapReduce [38] and catamorphisms (folds) are also helpful for parallelization. Both define ways to utilize a combining operator (reduce or folds) on a list or vector of data. This allows for parallelization as different processors or computers can operate on the elements of the list or vector. In addition, the reduction step is often able to be done in parallel. Fei and Lu recommend a dataflow based workflow that uses Map, Reduce, Tree, Loop, Conditional, and Curry. [64]

Many computations only look like they are serial. The computations can often be made parallel. Morihata et al. demonstrate the third homomorphism on trees in [110], which defines an algorithm to create a parallel algorithm from two serial algorithms. If a programmer can define a function that works serially from the beginning of a list to the end (rightward), and define a function that works serially from the end of the list to the beginning (leftward), then a parallel algorithm can be automatically created. Morihata et al. extend the third homomorphism from lists to also be valid for trees. Other systematic methods exist to convert seemingly serial code into parallel code.

The last 'big message' from Steele is to emphasize a divide and conquer approach over an accumulator approach. For example, if a person wants to add one million numbers together, one option is to use an accumulator. Each time the person encounters a new number, they add it to the accumulator. This algorithm does not allow for parallelization. If the computation is described as adding the first half of the million numbers to the second half, we can accomplish the same summation in half of the time with a divide and conquer approach. If we further divide each half of a million numbers into half again, we can approach a four times speedup. The divide and conquer algorithms typically divide and conquer until further division is impossible. This allows for massive parallelization.

In addition to the tips from Steel, it is beneficial to describe what is to be done rather than how it is to be done. By decoupling the what from the how, we can create efficient runtimes.

The analyst can describe his or her system of systems architectures using a text based model description which may map to visualizations. Crawley advocates in a foreword to [58] a modeling language that has both a graphical representation and a textual representation to engage both sides of the brain. Since RAAM strives to be visualization framework agnostic (while remaining standards con formant to ISO/IEC/IEEE 42010:2001), a graphical representation is not formally developed.

The text based model description is described in the following sections. The generation and execution of architecture alternatives occurs once the decision space is fully specified. The model description is written in a domain specific language that uses a combination of the host language and new keywords in a source code format.

The different RAAM concepts are shown in Figure 14. The different concepts are linked together as shown in the diagram. The solid lines show relationships between concepts. The white triangles point to concepts that are more general versions of the connected concepts. This diagram and similar ones shown later are intended to illustrate the main concepts in RAAM and the main generic relationships among those concepts. These diagrams could be further elaborated in future work if needed to become a full-fledged model of RAAM. For example, the Task concept is a general version of the three types of tasks, Main Task, Internal Task, and Leaf Task. Another way to explain the same idea is that the Main Task, Internal Task, and Leaf Task concepts are specialized versions of Task. The concepts will be linked to the domain specific syntax that is developed in the following sections. There are five main RAAM concepts:

- Computational Model
- Capability

- Task
- System
- Metric

The five main concepts are described with a domain specific language.

The following sections will use UML-like diagrams that convey relationships between RAAM concepts. UML can be extended with stereotypes, which are a way to create new model elements that have specific properties for a problem domain. In this case, two different stereotypes are used: <<keyword>> and <<concept>>. The stereotype <<keyword>> describes keywords from RAAM syntax. A keyword is a syntax element (word) that has a particular meaning to the computer language of interest. The meaning of the keyword is derived from the concepts that are brought together that define it. The stereotype <<concept >> is a way to designate the blocks that are concepts from the RAAM framework. The solid lines in Figures 15, 17, 18, 19, 20, and 21 are used to show which concepts feed into a given keyword. The keyword is informed by the connected concepts.

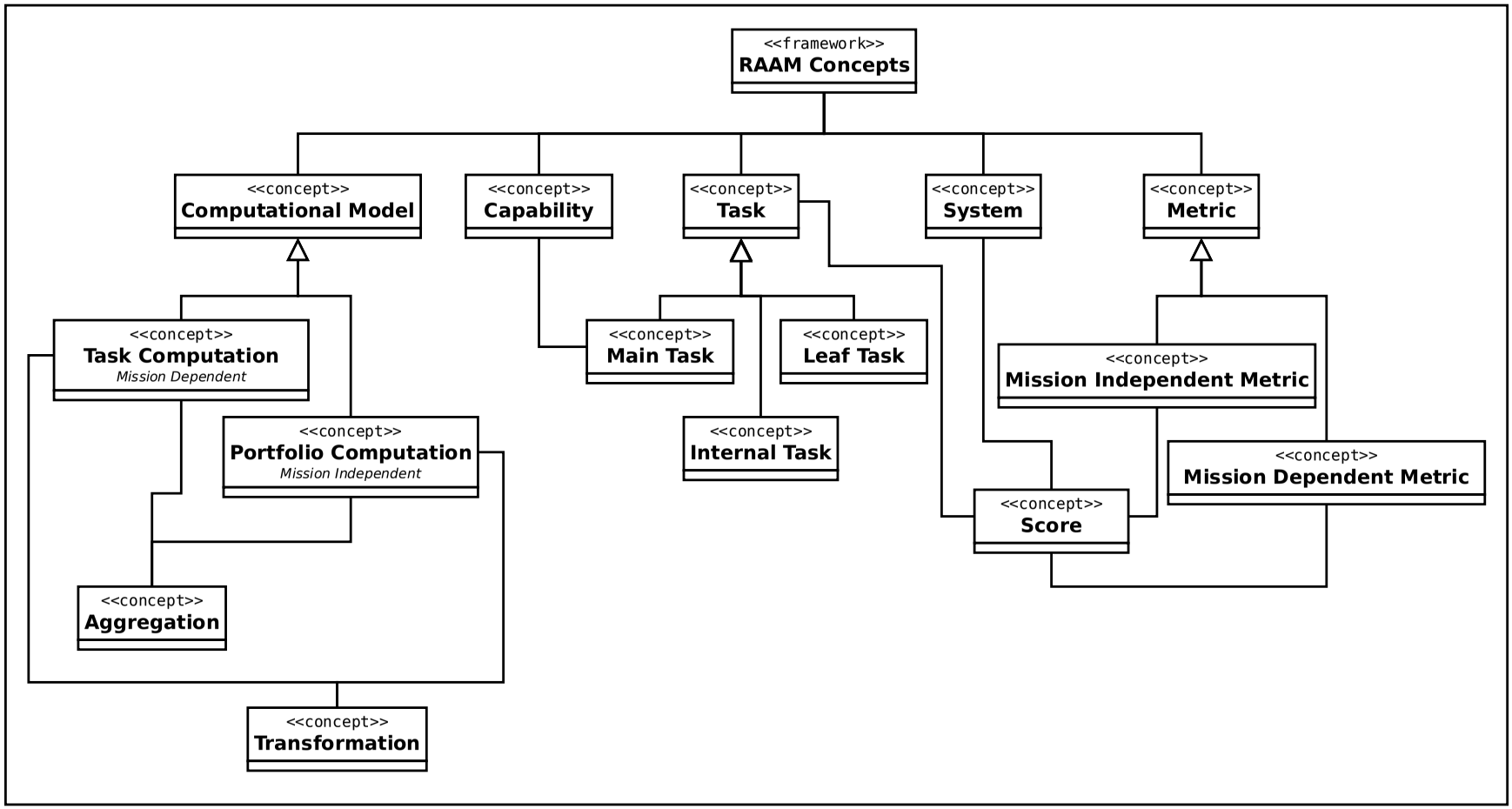


Figure 14: RAAM Concepts Diagram

7.4 Capability Hierarchy

The Capability Hierarchy of the system of systems architecture needs to be defined by the analyst. The Capability Hierarchy is a decomposition of the required tasks to accomplish a capability. Each task can be decomposed to the desired level of detail.

The Capability Hierarchy is a graph composed of tasks (nodes) and their connections (edges). There is a root to the graph which is the task that accomplishes the capability delivered by the system of systems architecture. The capability stores information about the root node of the task graph and the metrics of interest. Each task description is only concerned with its subordinate tasks.

Since each task only has information about its subtasks, the complete graph must be assembled in the generation portion of RAAM. The different architectural decisions dealing with which set of subtasks to use for a task are used to assemble the partial descriptions. The source code snippets are shown with the names of the attributes of the keyword and an example.

To convey the concepts we require a syntax. An S-expression based syntax typically used with lisp like programming languages is used although, a XML or CSV based syntax is possible. In an S-expression lists are natural to express in contrast to CSV based formats. S-expressions were chosen because they are used to represent both code and data. The architectural information in RAAM is a combination of code and data. A S-expression encloses data elements in parentheses. In this case, the comment character is ';' which makes the rest of a line a comment. Six keywords are defined which describe the syntax. Other types of modeling have been created in Common Lisp, such as discrete event simulations. [128]

7.4.1 Capability Hierarchy Text Based Source Code

The Capability Hierarchy is described with two keywords, **capability** and **task**. The **capability** keyword is used to keep track of the root task and associated capability

level attributes such as the description and metrics. The **task** keyword is used to store the possible subtasks of a task and a short description. The Capability Hierarchy is fully defined by a capability and a set of tasks.

```
(capability <NAME> <DESCRIPTION> <MAIN-TASK>
   <METRICS> <PORTFOLIO-METRICS>)


(capability complete-tasks
    "Complete the SEAD tasks"
    conduct-sead
    (P-success Complexity Time-to-completion
       Maintainability)
    (Cost Risk))
```

Listing 7.1: *capability* Keyword

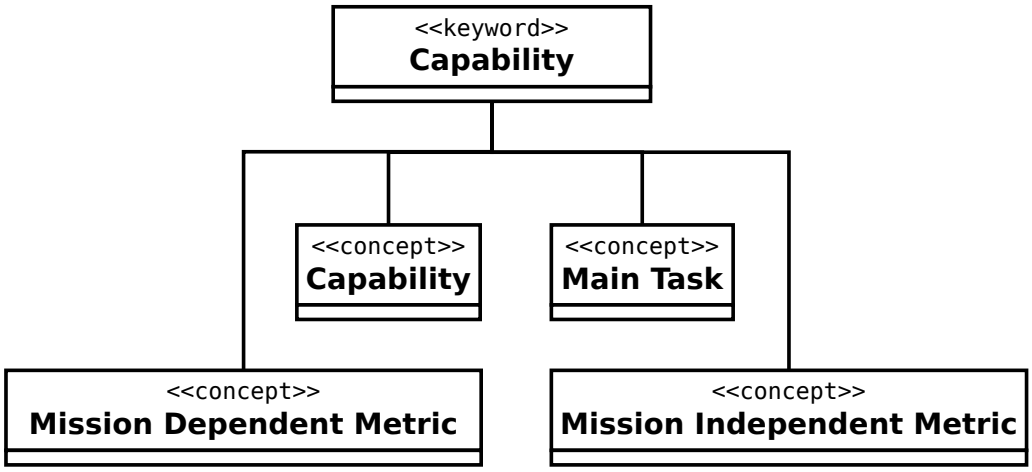


Figure 15: Capability Model

We need to know the NAME, DESCRIPTION, MAIN-TASK, METRICS, and

PORTFOLIO-METRICS of a capability. The MAIN-TASK is the root task for the capability. METRICS is a list of the mission dependent metrics. The METRICS list is composed of metrics that are dependent on the capability hierarchy and the portfolio. PORTFOLIO METRICS is a list of the mission independent metrics. The PORTFOLIO-METRICS list is composed of metrics that are dependent on the portfolio only. The **capability** keyword is shown in Listing 7.1.

The **capability** keyword is linked to the concepts of Capability, Main Task, Mission Dependent Metric, and Mission Independent Metric. This is shown in Figure 15.

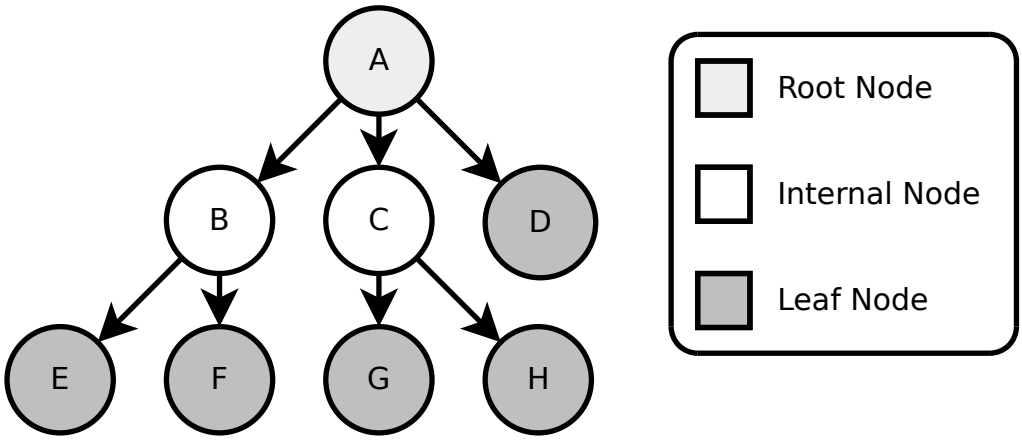


Figure 16: Node Terminology

```
task <NAME> <DESCRIPTION> (<SUBTASKS>)
                          (<SUBTASKS>) ...)

;Only one possible subtask decomposition
(task main-task "Main Task" (subtask1 subtask2))

;Multiple subtask decompositions
(task subtask2 "Subtask 2"
      (subtask3 subtask4 subtask5)
      (subtask6 subtask7))

;Leaf nodes have no subtasks
(task subtask3 "Subtask 3" ())

(task conduct-sead
  "Conduct SEAD to prepare battlefield for follow
     on attacks"
  (Detect Identify Correlate-and-Track
     Target-Assignment Weapon-Control
```

Listing 7.2: *task* Keyword

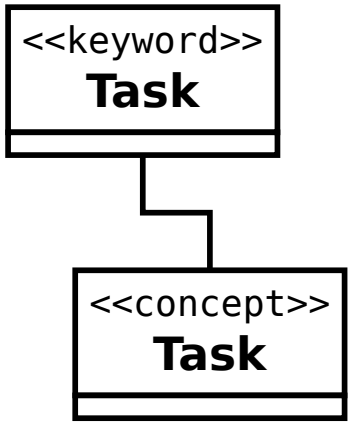


Figure 17: Task Model

For the **task** keyword we need to know the NAME, DESCRIPTION, and a series of lists of SUBTASKS. The purpose of the **task** keyword is to encode the relationships between the tasks by describing the subordinate relationships. Listing 7.2 shows four examples of using the **task** keyword. The task main-task shows an example with only one set of subtasks which is made of subtask1 and subtask2. The next example shows a task with multiple task subtask decompositions. Subtask2 can either be accomplished by subtask3, subtask4, and subtask5 or it can be accomplished with subtask6 and subtask7. The example with Subtask3 is an example of a leaf node task. There are no subtasks for subtask3 as it will have systems that accomplish it. The final example is an example from a Suppression of Enemy Air Defenses (SEAD) architecture study. The root task for that example is shown. Figure 16 shows the terminology used when describing tasks.

The **task** keyword is linked to the Task concept. This is shown in Figure 17. The **task** keyword is used in a recursive manner and refers to other tasks.

7.5 Candidate Systems

A candidate system is a system that may be used in the architecture. The main purpose of the **system** keyword is to store the mission independent metric scores for the system. These scores are used when calculating the mission independent metrics for the system of systems architecture alternative. A system can accomplish one or more tasks.

7.5.1 Candidate Systems Text Based Source Code

The candidate systems are described with the **system** keyword.

```
(system <NAME> <DESCRIPTION>
   <SYSTEM-ATTRIBUTE-PAIRS>)


(system Central-C2 "Local command and control"
  (Cost 15.0) (Risk 0.2))
```

Listing 7.3: *system* Keyword

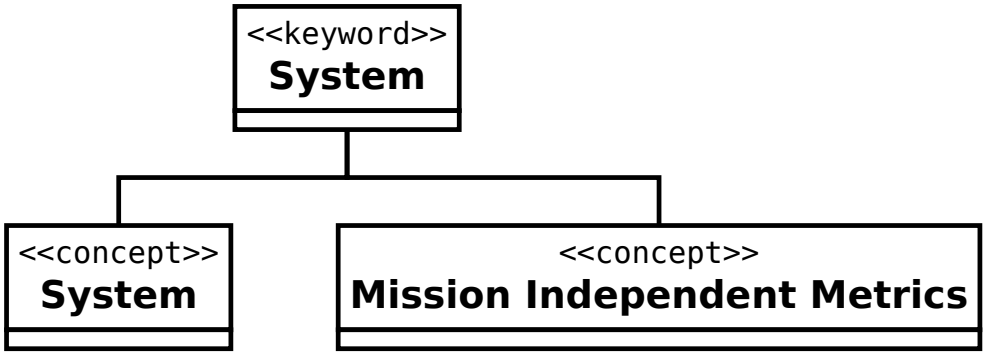


Figure 18: System Model

The **system** keyword requires the NAME, DESCRIPTION, and SYSTEM-ATTRIBUTE-PAIRS. NAME and DESCRIPTION are self explanatory but SYSTEM-ATTRIBUTE-PAIRS is a series of lists. Each mission independent metric is a list composed of the name of the metric and the value of that metric. Listing 7.3 shows the **system** keyword.

The **system** keyword encapsulated the System and Mission Independent Metric concepts. This is shown in Figure 18.

7.6 Computer Model Text Based Source Code

The computer model is where the computational elements are defined. The translator needs to know what the metrics of interest are and how to compute the aggregation and transformation on the subtasks.

The three keywords used are **metric**, **compute**, and **portfolio-compute**. The difference between **compute** and **portfolio-compute** is that **portfolio-compute** operates on metrics that are mission independent metrics. This means that they only require information about the portfolio and not the capability hierarchy. The **compute** keyword operates on the mission dependent metrics.

```
(metric <TASK> <SYSTEM> <METRIC-SCORE-PAIRS>)


;Subtask2 - Subtask3,Subtask4, Subtask5
(metric subtask3 S5 (A 5)  (B 5))
(metric subtask4 S1 (A 7)  (B 8))
(metric subtask4 S3 (A 11) (B 13))
(metric subtask5 S4 (A 13) (B 21))
```

Listing 7.4: *metric* Keyword

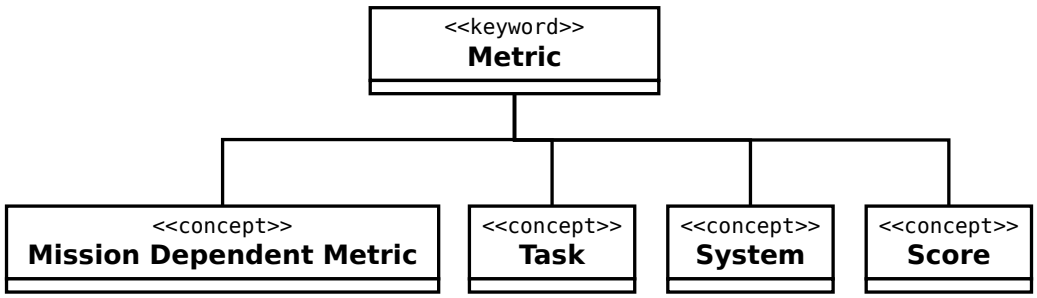


Figure 19: Metric Model

The **metric** keyword is used to document the metric to system to task score mappings. Listing 7.4 shows the **metric** keyword. A task and a system must be specified to associate the following metric scores with. Since multiple metrics must be considered, a series of metric score pairs are included for each system to task pair. In this example, A and B are mission dependent metrics.

The **metric** keyword can be used to provide data on many different types of models. There has been interest in modeling the effect of different numbers of the same system in an architecture alternative. In this case, the aggregation function accepts a score that is based on the number of systems included in the architecture alternative. The use of the word 'system' is general, in that it may refer to a set of systems, a system, a subsystem, or a component. Anything that accomplishes a task with different performance is considered a different system (e.g., one aircraft providing target geolocation vs two aircraft providing target geolocation).

The **metric** keyword is linked to the Mission Dependent Metric, Task, System, and Score concepts. This is shown in Figure 19.

```
(compute <TASK> <METRIC> <AGGREGATION>
         <TRANSFORMATION>)


;; ident is the identify function
(compute main-task A * ident)
(compute main-task B + ident)
(compute subtask2 A + ident)
(compute subtask2 B * ident)
```

Listing 7.5: *compute* Keyword

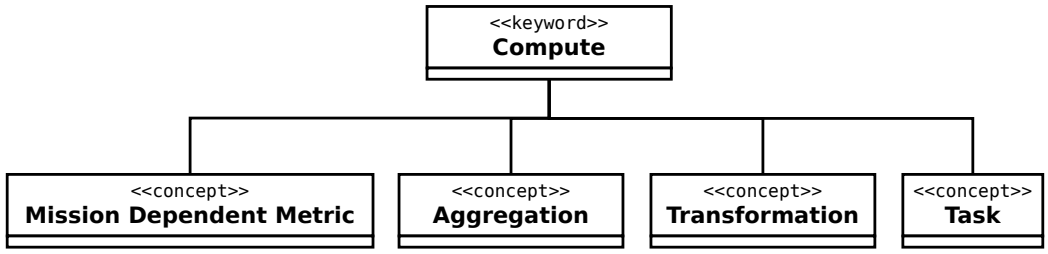


Figure 20: Compute Model

The **compute** keyword is used to show a relationship between a task, metric, and the associated aggregation and transformation. The **compute** keyword is shown in Listing 7.5 and operates on mission dependent metrics. To make an executable model, the method of computation for each task must be known. The task has an aggregation function that is used to combine the scores of the subtasks. The aggregation function is a many-to-one mapping that combines the metric scores from subnodes. Each task has a transformation function that is used to scale the result of the aggregation function. The transformation function is a one-to-one mapping. A different transformation and aggregation function pair can be used for different

metrics on the same task. The transformation and aggregation functions are any functions that meet the required arity (number of arguments, in this case many or one) and can be programmed into a computer. The compute keyword requires the task, metric, aggregation function, and transformation function. A **compute** keyword is made for each task that has subtasks (non-leaf tasks) for each metric.

Any aggregation function can be defined for the tasks as long as it takes a list of subtasks and returns a single value. The goal is to "roll-up" the metric from the system level scores to the capability level. In general, aggregation functions will return different values when given different sets of subtasks or systems. It is up to the modeler to determine the applicability of their aggregation functions. Analyzing a capability may use many different aggregation and transformation functions. A common subset of possible functions is currently defined for the initial proof of concept translator.

The **compute** keyword is linked to the Mission Dependent Metric, Aggregation, Transformation, and Task concepts. This is shown in Figure 20.

```
(portfolio-compute <NAME> <AGGREGATION>
   <TRANSFORMATION>)


(portfolio-compute Cost + ident)
```

Listing 7.6: *portfolio-compute* Keyword

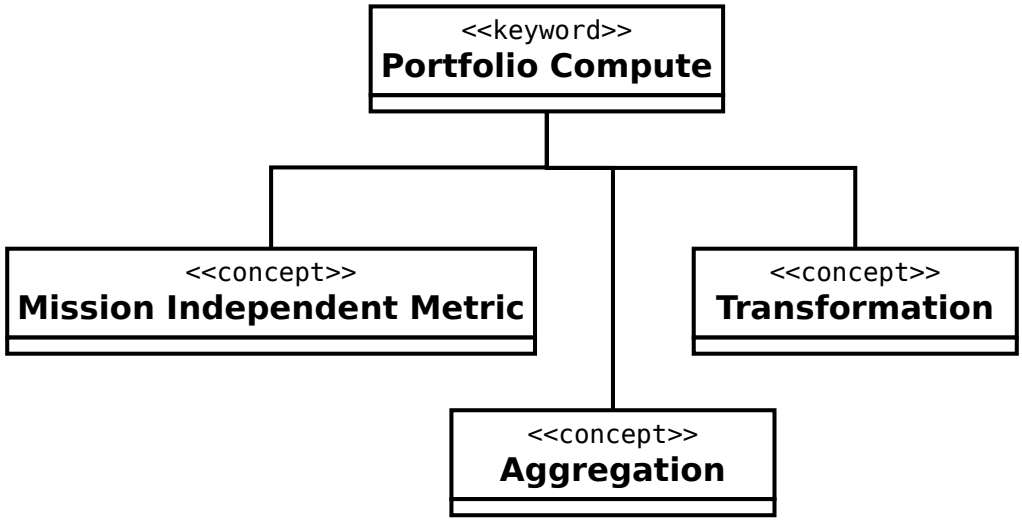


Figure 21: Portfolio Compute Model

The **portfolio-compute** keyword is very similar to **compute** but operates on mission independent metrics. Listing 7.6 shows the **portfolio-compute** keyword. It is possible to design a version of RAAM with only the **compute** keyword to handle both mission dependent and mission independent metrics. By separating the different semantic concepts with more syntax, the difference between mission independent and mission dependent metrics is made explicit. The **portfolio-compute** keyword combines the system attributes for a metric. A **portfolio-compute** is created for each mission independent metric.

The **portfolio-compute** keyword is linked to the Mission Independent Metric, Aggregation, and Transformation concepts. This is shown in Figure 21.

The syntax and semantics are compatible with ISO/IEC/IEEE 42010:2011. [80] RAAM does not attempt to conform to the entire standard, but is envisioned as being used with other architecture frameworks that are compatible with the standard. RAAM defines elements of an architecture description language. Since the standard is mainly concerned with documenting a system rather than the possible set of systems,

the applications of concepts is not one to one.

7.7 Interpreter/Compiler

Once the problem has been fully described by the domain specific language, it must be run on a computer. The translation of the domain specific language to machine code is done by an interpreter or a compiler. Since the domain specific language is specified separately from the interpreter or compiler, different versions can be created by different organizations to produce a common output.

The RAAM domain specific language is declarative, which allows for changes in the translation step done by an interpreter or compiler to incorporate advances in hardware and software without changing the RAAM syntax. This property enables the ability to explore new techniques to improve analysis runtimes without changing the architecture descriptions.

The automatic generation of alternatives and the automatic creation of executable models occur during the translation step. The previously described syntax is converted into a target language's source code and compiled. The code used to accomplish this is described in Appendix A. Distributed hierarchical algorithms are desired for this class of problem. [109]

The structure of the translation step is important when considering the magnitude of executing billions and trillions of architecture alternatives. Often when doing an architecture analysis, the different architecture alternatives are first generated and then executed. With large numbers of cases, storing the generated list of cases to run is on the border of feasible. An integer that specifies up to one billion requires thirty bits of information. One billion cases times 30 bits per case (in an ideal situation), results in 3.5 gigabytes of data just to store information about each case. Repeating the calculation with a trillion cases, it would take 4.55 terabytes. In addition, retrieving that amount of data currently requires using a hard disk. A hard disk is slow compared

to random access memory (RAM). Another option is to generate the different cases as they are executed in a streaming fashion. In that method of generating cases, only the previous case is stored. Generating the different architecture alternative cases in a streaming fashion is appealing and is the approach used in the RAAM runtime. In addition, if the execution of an architecture alternative is made fast enough, it may be faster to recalculate the metric scores for an architecture alternative than it would be to store the information and retrieve it. The author believes that streaming the generation of alternatives and recalculating the metrics on the fly is the fastest way to analyze large numbers (10^9 to 10^{12}) alternatives.

7.8 Generation

The generation step of RAAM is used to enumerate all of the possible architecture alternatives. A specific architecture is created when both the operational and system decisions are made. The operational decisions are decisions between different sets of subtasks in an architecture. The system decisions are decisions made when a system is allocated to accomplishing a specific task. Each task node has a decision function which selects a set of subtasks or a system to accomplish the task. This covers both operational and system decisions. The decision function can be changed depending on which method of generation is required. For example, if a Monte Carlo analysis is desired, the decision function can randomly pick between the task node choices. If enumerating all of the different architecture alternatives, the decision function is told which set of subtasks to choose. The number of generated alternatives is generally limited by the available computational resources to evaluate the alternatives. With current computing power, approximately 30 to 40 binary decisions are possible or billions to trillions of alternatives. Generation of alternatives is, in general, a small portion of the total runtime. The limitations on the generation of alternatives occurs when attempting to execute the large number of alternatives.

The current version of RAAM operates by only enumerating the feasible set of architecture alternatives. Previous versions enumerated all permutations of the possible decisions (including both operational and system). By enumerating the permutations of all of the systems, duplicates of architecture alternatives are possible. The duplicates arrive because decisions can be dependent on other decisions. Operational decisions can change the structure of the final task tree, which may change both the downstream operational decisions and system decisions. The simplest example of a dependent decision is when a task node has to choose between two sets of subtasks such as *Subtask 2* shown in Figure 22. The decision of what system should accomplish *Subtask 7* only needs to be made if the subtask set of *Subtask 6* and *Subtask 7* is chosen to accomplish *Subtask 2* rather than *Subtasks 3,4,* and *5*.

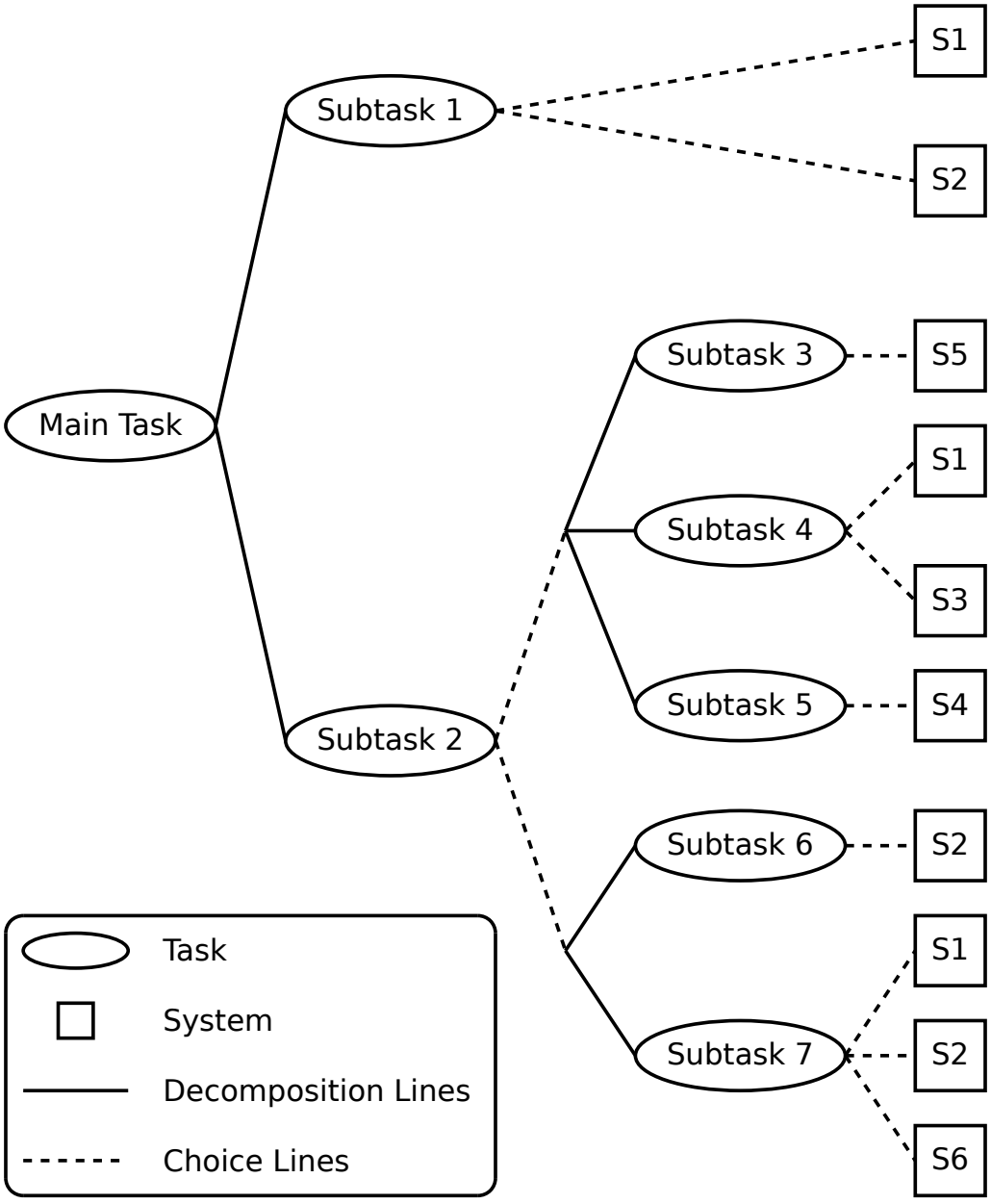


Figure 22: Dependent decisions example

The generation of architecture alternatives works on a data structure that contains the possible decision space. The data structure is created as the tasks are read in. A function is created to return the set of subtasks that corresponds to a decision made for a task. As the generation algorithm progresses, decisions are made and left as made decisions until changed.

Before the generation algorithm is discussed, the decisions available at each "level" of depth in the task hierarchy tree must be determined. There is only one task at the root of the task hierarchy which represents the capability. Even the root can have a decision about which tasks it should accomplish. Any decisions on the current level of the tree must be made. If they are, then we can return the tasks that only correspond to the subtasks of the tasks in the current level. By keeping track of the current level of the tree and requesting the next level of tasks only when all of the decisions are made at the current level, the issue of dependent tasks resolves itself. Once all of the decisions for a level of the hierarchy are made, the algorithm loops through all of the tasks in the current level and creates a new list of the subtasks of each of the tasks at the current level. By generating the list of decisions available at each level when required and not changing decisions from previous levels, the entire design space can be explored.

The generation algorithm is a recursive algorithm. The algorithm's flow chart is shown in Figure 23. The algorithm begins by being passed the possible decisions from the next level in the hierarchy. Both a list of decisions to be made (closed decisions) and a list of possible but not made decisions (open decisions) are stored. If the open decisions list is empty, then all required decisions for that level have been specified in the list of closed decisions. The listed closed decisions are made which causes the tasks to return the correct set of subtasks. If the next level in the hierarchy does not have any decisions left to make, then all of the decisions to uniquely specify an architecture alternative have been made and the executable model is executed. If

the next level in the hierarchy did have decisions left to be made (open decisions), then the permutate decisions algorithm is run again with the possible decisions of the next level. Going back to the original decision in the algorithm, if the open decisions list still has elements, then the permutate algorithm is run with the first open decision moving to the closed decisions list. Next, the algorithm checks if there is only one option of which set of subtasks is left. If there is only one choice left, then the algorithm is done and does nothing. If there is more than one choice of sets of subtasks, then the permutate decisions is begun again with a different decision for the first task in the open decisions list.

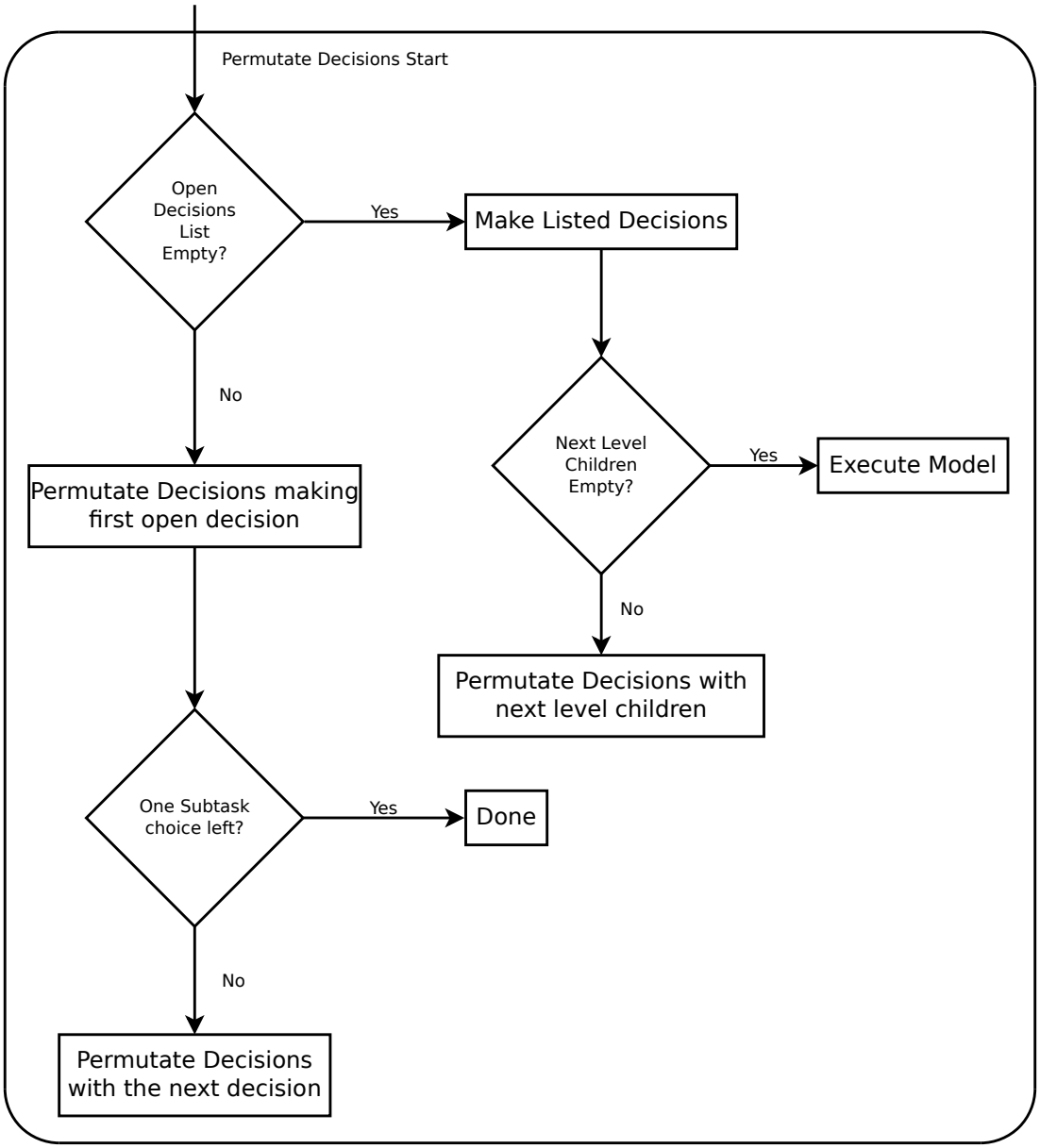


Figure 23: Permutate Decisions Flow Chart

The reader is encouraged to refer to the source code contained in Appendix A for a formal description of the algorithm in Common Lisp.

Since the operational and system choices are represented the same way in the RAAM implementation, the generation of alternatives uses one algorithm for the enumeration of architecture alternatives. This unification makes the process conceptually more simple, easier to maintain, and computationally efficient.

7.9 Evaluation

7.9.1 Mission Independent Evaluation

Mission independent metrics are computed from the list of systems that is assigned to the system of systems architecture alternative. The list of systems constitutes the portfolio. This list of systems is created after all of the decisions are made which specify a unique architecture alternative. Since only the leaf tasks have systems, the leaf tasks that are part of the portfolio are interrogated for which system they have allocated to them. To find only the leaf systems that are reachable from the root task, the program traverses the task hierarchy starting at the root and taking the decisions that have been made.

7.9.2 Mission Dependent Evaluation

Mission dependent metrics operate using the task hierarchy. The evaluation of an architecture alternative is straightforward once the decisions are made. As the RAAM model is read in, the task, metric, system, and compute information is translated into a method tree that will evaluate the required models. The details are shown in Appendix A with the evaluation step concentrated in the **compute** description. Each task node becomes a generic method that specializes on the current metric type. Once the decisions are made in the generation step, the evaluator runs the main task method on each of the metrics.

The evaluation occurs by translating the description of the architecture alternative

into a computer program. The task hierarchy stores information about how the different tasks are linked to each other through the subtask information. Each task has information about how to reach its subtasks. Which set of subtasks is reached is dependent on the decision. The system information is used to inform the leaf tasks that they will have to decide which system accomplishes them. Systems can only accomplish leaf tasks. For leaf tasks, a set of pseudo-subtasks is created that access the system metric scores. The pseudo-subtasks are named with the task name, a dash, and the system name. For example, for task Subtask1 capable of being accomplished by system S1 and S2, the pseudo-subtasks are a list of Subtask1-S1 and Subtask1-S2. The metric keyword is used to create functions that return the score for a system accomplishing a specific task, which also specialize on the metric type. In this way, the operational level decisions and the system level decisions are equivalent.

Each task has a task function for each metric that applies the aggregation function to the chosen subtasks of the task (many to one) and then applies the transformation function (one to one).

The evaluation begins with calling the method of each selected subtask with the current metric of interest. When all of the subtasks have returned their result, the aggregation function is applied to the results. The result of the aggregation function is transformed with the transformation function to be the final result for that task. The process is repeated until the entire task hierarchy has been evaluated. As explained above, leaf nodes do not have aggregation or transformation functions, they have a function that returns the metric score for that system to task pairing. The process for the evaluation stage is shown in Figure 24.

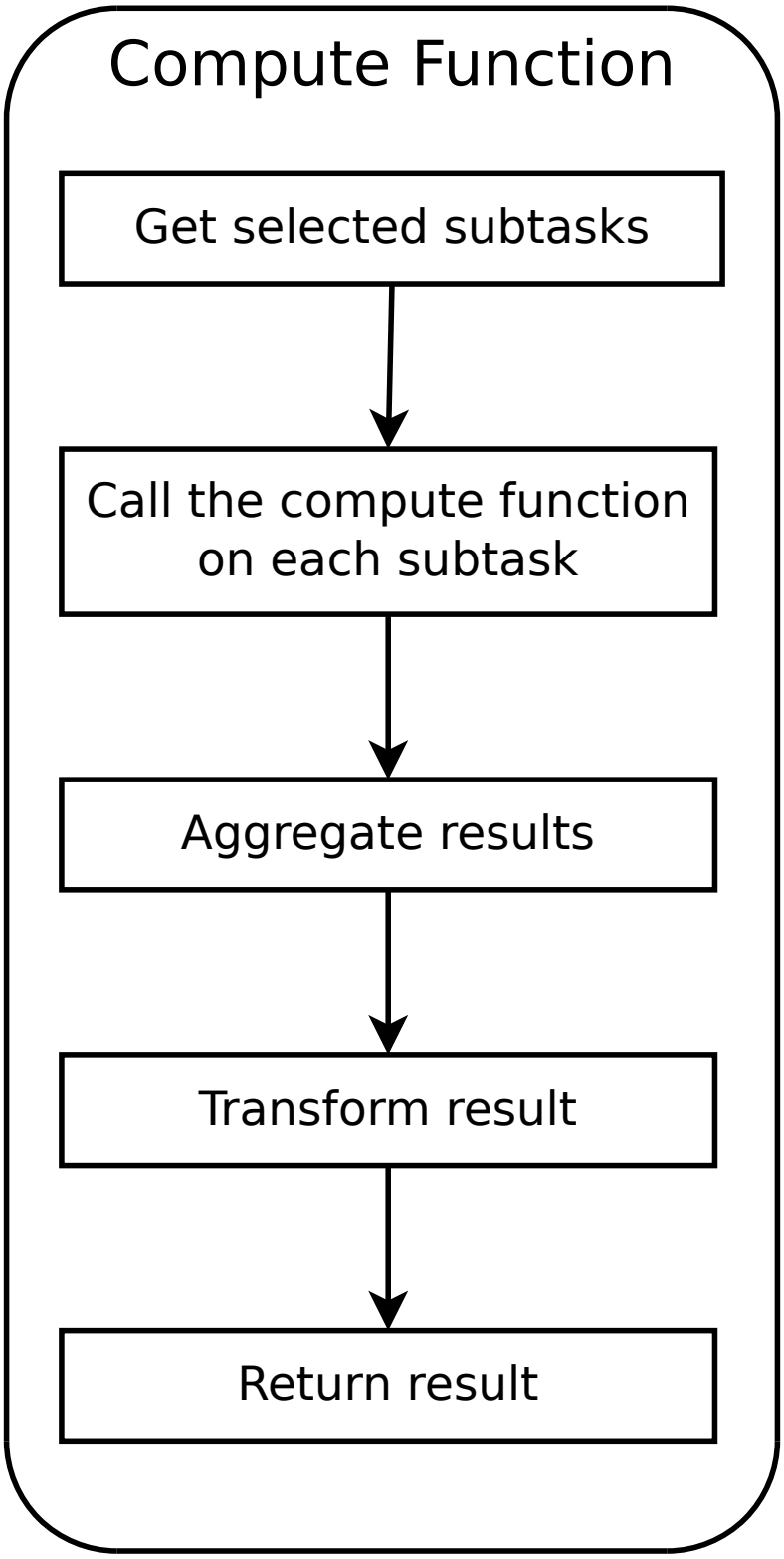


Figure 24: Mission Dependent Evaluation with Compute Flow Chart

The different tasks are stored as separate entities in memory until decisions have been made which make connections between the tasks. Figure 25 shows a set of disconnected tasks. Figure 26 shows the simplified task hierarchy ready for execution.

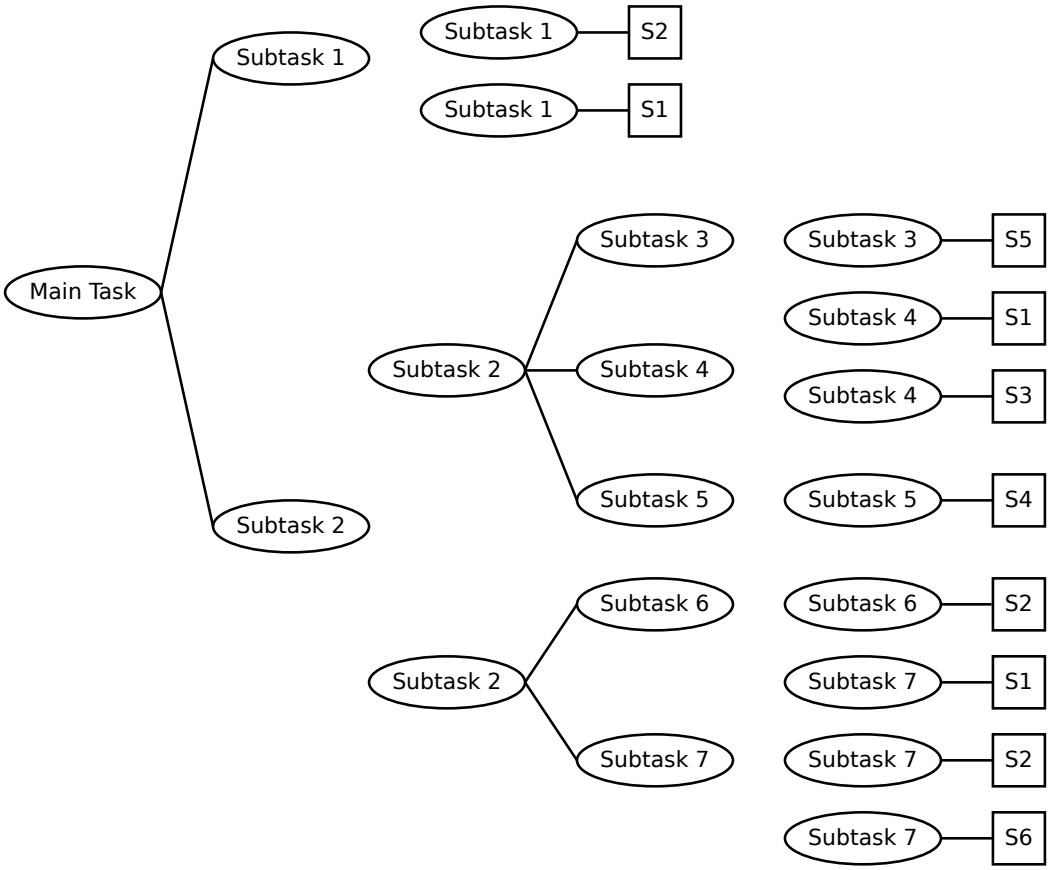


Figure 25: Set of Disconnected Tasks

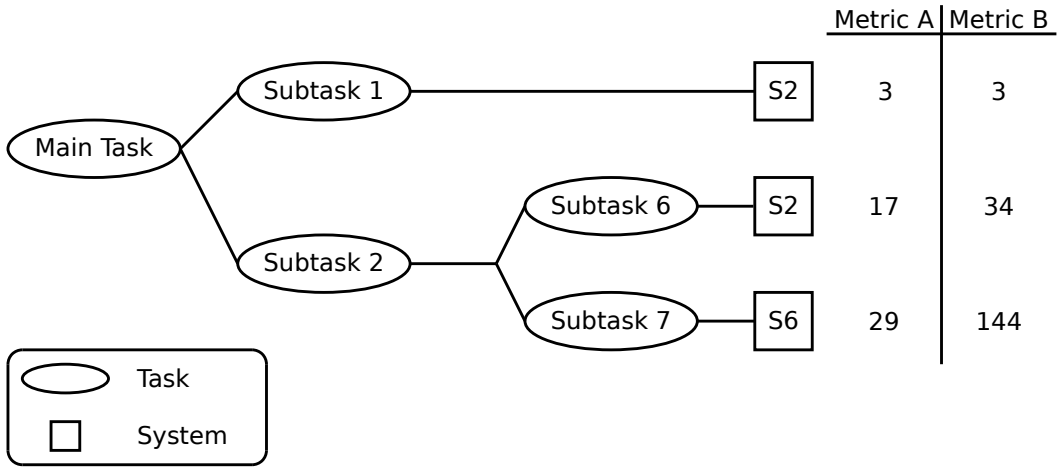


Figure 26: Task Hierarchy with Decisions Made

7.9.3 Possible Decision Space

The evaluation of an architecture alternative explores the possible decision space. The possible decision space is defined as the set of decisions that must be made to uniquely specify an architecture. Two different types of decisions are under consideration with this method; operational and system trades. The operational decisions are made when selecting how a given task is accomplished. A task can have one or more possible sets of subtasks. The different sets of subtasks describe different operational architectures. The allocation of a system to a task is another decision that is considered with this method. Allocating different systems to the same task will change the performance of the system when accomplishing a task. The method is designed to evaluate all of the possible architecture alternatives arising from the defined decision space. Since a given decision splits the decision space into equal sized regions, it is convenient to use those regions in parallelization strategies.

As compared to the Architecture Evaluation and Enumeration framework, RAAM does not count infeasible architectures. The Architecture Evaluation and Enumeration framework allows for hundreds or thousands of feasible architectures verses

hundreds of billions or trillions with RAAM utilized on the cloud. [133]

7.10 Parallelization of Evaluation

The era of continuous rises in CPU clock speed are in the rear view mirror. Parallel computation will be the only way to get speed increases in the future. [131] Luckily, each evaluation of an architecture alternative is independent from the other evaluations. This allows for parallelization of the evaluations. Although it is possible to parallelize an individual architecture alternative evaluation, that aspect is not explored in this dissertation. The different branches of the task hierarchy tree can be evaluated in parallel as they do not impact each other in a metric. Parallelizing the evaluation of an architecture is only desirable when the run time of one architecture alternative is large and there are few architecture alternatives to evaluate. Synchronization of the evaluation of an architecture alternative may introduce overhead that does not improve runtime. Two different ways for parallelizing the evaluation of an architecture alternative were explored. The first of these alternatives uses threading on one computer. The other method for parallelizing the architecture evaluation process is by deploying the analysis to a compute cloud or cluster. These two methods are explored further in the subsequent sections.

7.10.1 Thread based Parallelization

An operating system thread is the smallest unit of processing scheduled by an operating system. Most computing architectures have threads contained within a process. By being contained within a process, the threads can share memory implicitly. Modern CPU architectures such as Intel's x86 family have hardware support for threading. A significant portion of the time that a CPU spends can be utilized in accessing data. By using threading, the processor can do computations while waiting for data to arrive.

A set of decisions is determined to be made with in the main thread and a different set of decisions is made in each of the spawned threads. To make the evaluation step parallel using threads, a modification to the standard algorithm is required with a third argument to the permutate decisions added. The third argument is a list of decisions that will be made in parallel is called *dynamic-decisions*. The algorithm is the same except that when a decision has been made (closed) for each of the possible dynamic decisions a thread is created to run the evaluation algorithm with the dynamic decisions placed in the *not-made-decisions* (open) list. It is recommended to use the same number of threads as physical CPU-cores for best performance. Each thread is a unique combination of a set of decisions drawn from *dynamic-decisions*. Newer processors with hyperthreading may report more virtual CPU-cores, but can result in lower performance when using too many threads. The process to parallelize the execution using threads is detailed in Appendix B. Figure 27 shows how the decisions are allocated between threads. The RAAM code is not often memory bound, but it is CPU bound which is beneficial when parallelizing to cloud computing infrastructures. By only utilizing the same number of threads as physical CPU cores, context switching, or switching between different threads, is reduced.

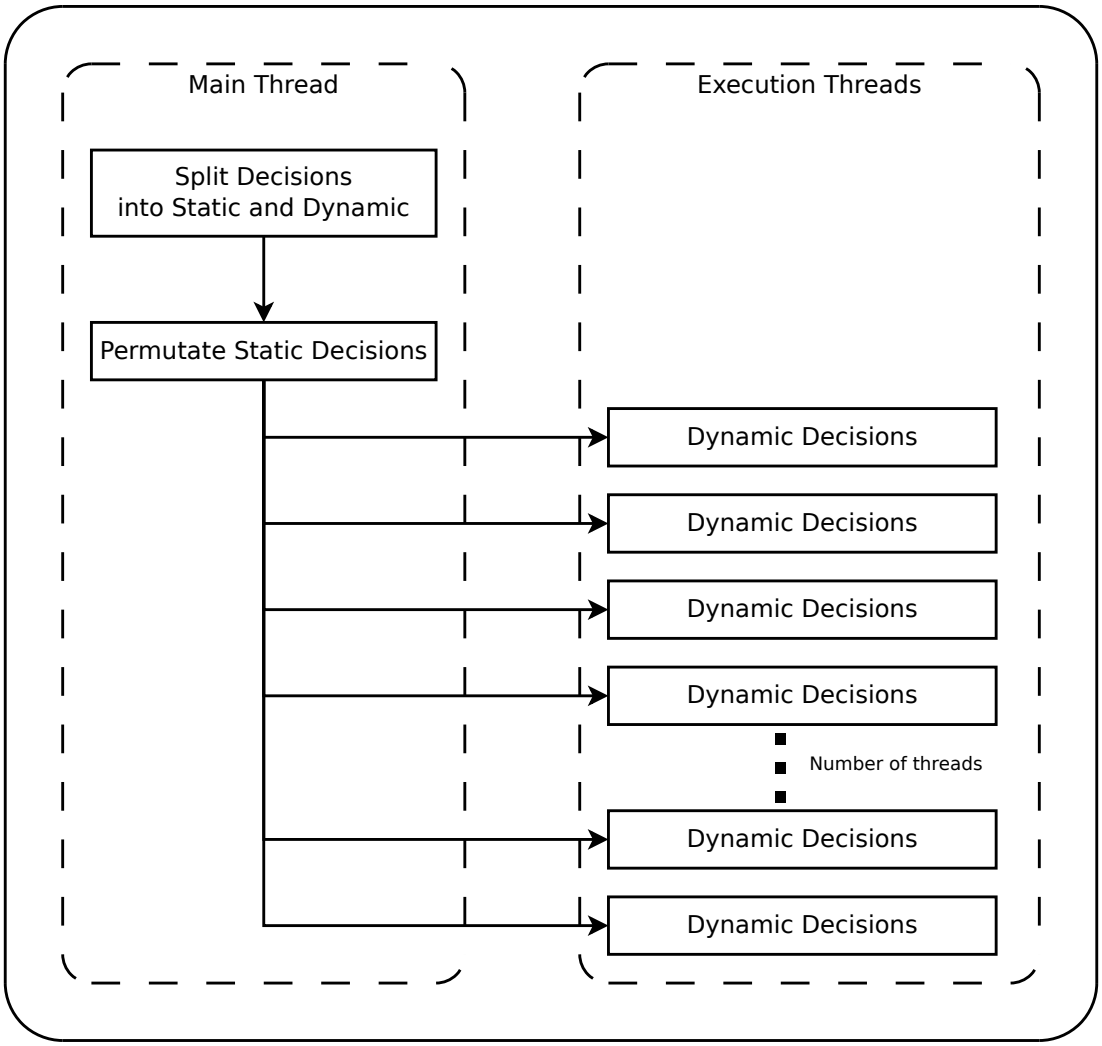


Figure 27: Decision Allocation between Threads

7.10.2 Cloud Based Parallelization

Computing power is now becoming available as a utility in the form of cloud computing. Providers such as the Amazon Elastic Compute Cloud (EC2) [5] or university clusters can provide multiple computers to researchers for low cost. Computational power is available for rent by the minute. Since the computation of the architecture alternatives is independent, the decision space can be split and sent to multiple computers and merged together in the end. Cloud computing allows organizations to "surge" computing power as needed, but not pay costs for upkeep when the computer

power is not required. System of systems architecture analysis occurs infrequently, so dedicated clusters will not be cost competitive with cloud computation. In addition, with more computers available, the nature of the problem leads to the computation time scaling close to linearly with minimal overhead.

In addition to a parallel algorithm using threads, portions of the decision space can be evaluated on different computers. The results of the architecture alternative evaluation are combined after all of the cases have been run. The algorithm is the same as was used for the evaluation of an architecture except that some decisions are pre-made and put into the *made-decisions* list at the beginning of the algorithm. The current demonstration version chooses a subset of decisions that splits the decision space into the number of pieces that corresponds to the number of available computers. Figure 28 shows how the decisions are distributed to computers on the cloud.

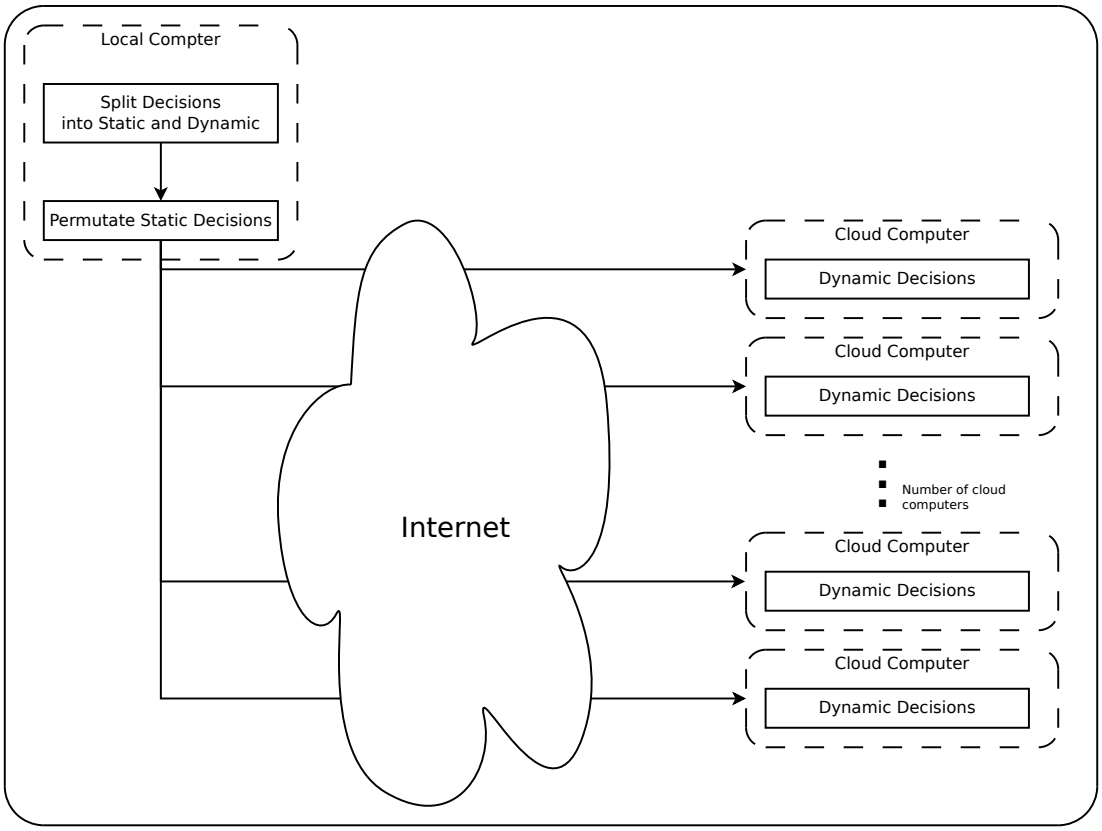


Figure 28: Architecture of RAAM on the Cloud

For evaluating alternatives on the cloud, the executable must be shared between the computers and the resulting data must be returned to the analyst. A computer program was created to control the computers on the cloud and combine the results. The demonstration implementation saves an executable with the possible architecture computational models. To control which part of the decisions space is explored, a file is created with the commands to run the one computer algorithm with some of the decisions pre-made. Both the executable and the control file are sent to a computer on the cloud. When the results are available they are returned to the originating computer.

The algorithm and program for running on cloud and cluster computing resources was tested at the Amazon Elastic Compute Cloud (EC2) [5] and the PaSTEC (Parallel Software Testing and Evaluation Center [124]) cluster at the Georgia Tech Research

Institute. The Amazon Elastic Compute Cloud is used to rent computers as needed at an hourly rate. This allows an analyst to only pay for the computing infrastructure when needed and to easily trade dollars for time. The PaSTEC cluster is designed for research, testing, and evaluation of parallel high performance computing software. Three different types of computers are used in the cluster with two instruction set architectures. Details of the implementation for parallel computing are discussed in Appendix B.

CHAPTER VIII

TEST PROBLEMS

8.1 Canonical Example

A canonical example was created to help with the development of the RAAM method. The example contains many of the characteristics of a full featured system of systems analysis. A full featured system of systems analysis has system choices, operational choices, multiple levels of hierarchy, and multiple metrics. There are system to task allocations that have decisions between what system to pick for which task. The operational structure of the architecture can vary which leads to operational decisions and dependent decisions. There are multiple metrics. The task hierarchy has different hierarchical depths. For this example, there are two metrics of interest, A and B. Six different systems are possible and eight tasks are defined.

The canonical example is detailed in Figure 29 and shows the full architecture alternative space for this example.

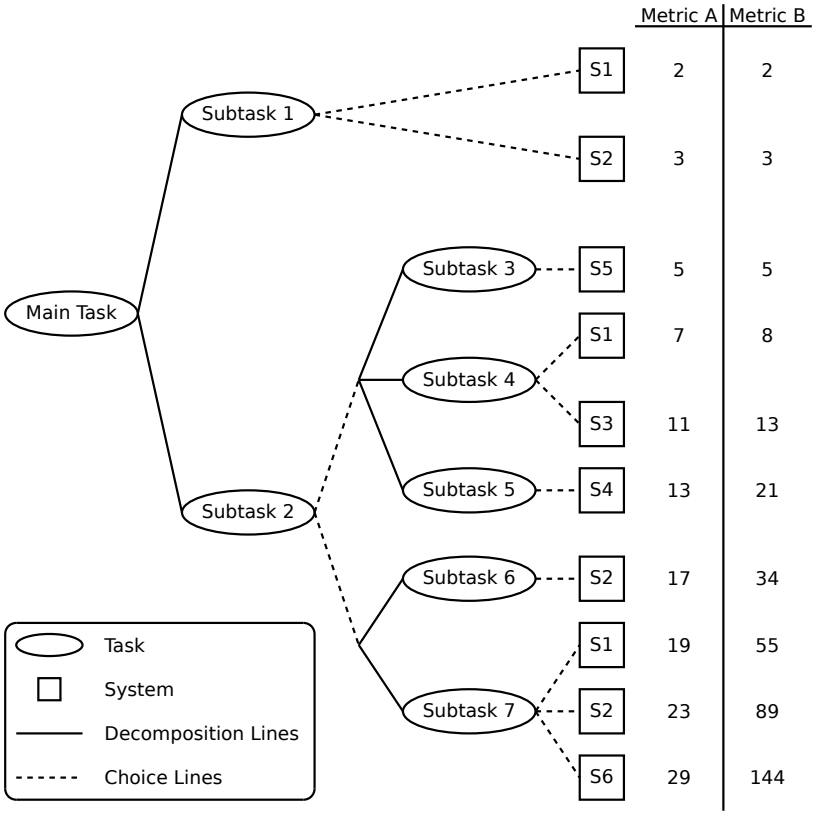


Figure 29: The Canonical Example of an Architecture Trade Space

Tasks are shown in ovals with the dotted lines representing possible decisions. The solid lines are decomposition lines that show the hierarchy. By making a set of decisions, as shown in Figure 30, an architecture alternative is defined, shown in Figure 31. To create the architecture alternative, three decisions had to be made. First, the decision associated with *Subtask 1* regarding which system would accomplish *Subtask 1*. In the example, the decision was for *Subtask 1* to be accomplished by *System 2*. Second, the operational decision for *Subtask 2* is required between using the set of *Subtask 3, 4, and 5* or *Subtask 6 and 7*. In this example, *Subtask 6 and 7* were chosen to accomplish *Subtask 2*. In the formulation, *Subtask 6* does not have a decision as there is only one option, but *Subtask 6* is accomplished by *System 2*. The third decision is for *Subtask 7* where *System 6* was selected to accomplish the task.

If for *Subtask 2* the set of *Subtask 3, 4, and 5* were chosen, then the third choice would be which system between *System 1 and 3* for *Subtask 4*. The possible decisions are summarized in Table 2. Once the decisions have been made, the architecture alternative is fully defined and is simplified to Figure 31.

Table 2: Task and System Decisions for the Canonical Example

Task	Possible Decisions
Subtask 1	S1,S2
Subtask 2	{ST3, ST4, ST5},{ST6,ST7}
Subtask 4	S1,S3
Subtask 7	S1,S2,S6

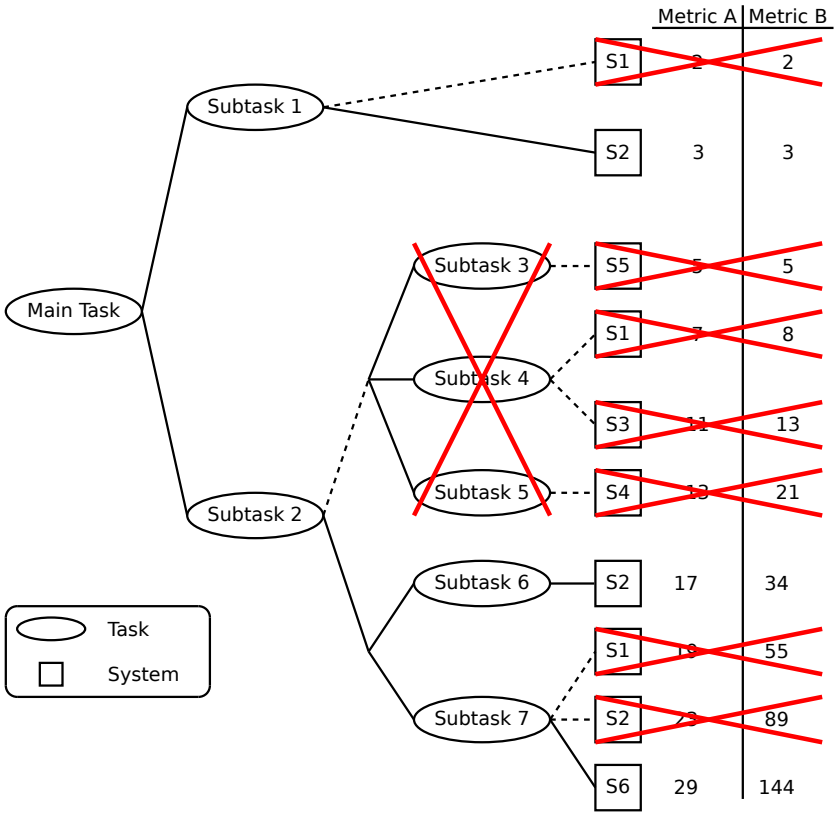


Figure 30: The Canonical Example with Decisions Made

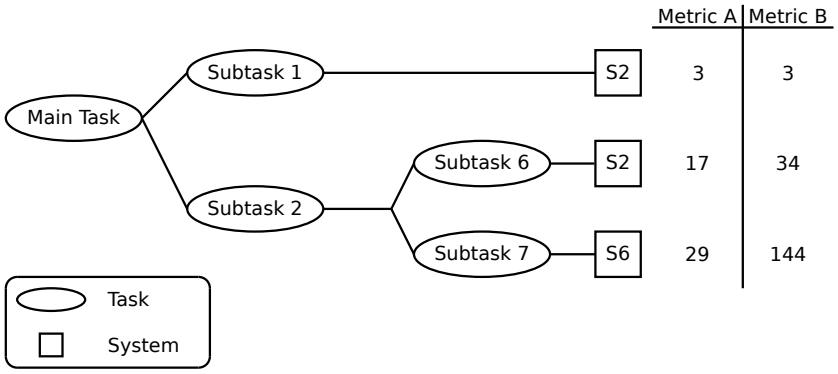


Figure 31: An Architecture Alternative Created with the Canonical Example.

The evaluation of the architecture alternative shown in Figure 31 requires three metric score lookups and two aggregation and transformation steps. First, the metric scores for a given system to task mapping are retrieved. The metric scores are typically generated by using Subject Matter Experts (SMEs) or system specific models. In this case, the score for *Subtask 1* using *System 2* is 3 and 3 for metric A and B respectively. *System 2* accomplishing *Subtask 6* results in a score of 17 for metric A and a score of 34 for metric B. *System 6* accomplishing *Subtask 7* results in a score of 29 for metric A and a score of 144 for metric B. For simplicity, both transformation functions are the identity function and do not modify their values. The aggregation function for *Main Task* is defined as a product for metric A and a sum for metric B. The aggregation function for *Subtask 2* is defined as a sum for metric A and a product for metric B. The calculation for metric A is $(3 * (17 + 29)) = 138$ and the calculation for metric B is $(3 + (34 * 144)) = 4899$.

8.1.1 Exhaustive Search for the Best Portfolio

There are ten architecture alternatives in the canonical example described in Figure 29. If we do not take into account the dependent decisions, there are twenty four different architecture alternatives. The ten architecture alternatives are summarized in Table 3. The scores for the metrics can be used later to determine the best

architecture alternative after discovering the decision maker's preferences.

Table 3: Architecture Alternative Description and Scores for the Canonical Example

ST7	ST6	ST5	ST4	ST3	ST2	ST1	Metric A	Metric B
S1	S2	–	–	–	(ST6 ST7)	S1	72.0	1872.0
S1	S2	–	–	–	(ST6 ST7)	S2	108.0	1873.0
S2	S2	–	–	–	(ST6 ST7)	S1	80.0	3028.0
S2	S2	–	–	–	(ST6 ST7)	S2	120.0	3029.0
S6	S2	–	–	–	(ST6 ST7)	S1	92.0	4898.0
S6	S2	–	–	–	(ST6 ST7)	S2	138.0	4899.0
–	–	S4	S1	S5	(ST3 ST4 ST5)	S1	50.0	842.0
–	–	S4	S1	S5	(ST3 ST4 ST5)	S2	75.0	843.0
–	–	S4	S3	S5	(ST3 ST4 ST5)	S1	58.0	1367.0
–	–	S4	S3	S5	(ST3 ST4 ST5)	S2	87.0	1368.0

8.1.1.1 Sample Calculations for Each Alternative

The canonical example has ten different architecture alternatives. The simplified alternatives are shown below in Figures 32, 33, 34, 35, 36, 37, 38, 39, 40, and 41. The computational model calculations are shown for each alternative. There are two different possible operational architectures. The two operational architectures are due to the decision for which set of subtasks to use for Subtask2. For metric A, the score is **(Subtask 6 + Subtask 7) * Subtask 1** or **(Subtask 3 + Subtask 4 + Subtask 5) * Subtask 1**. For metric B, the score is **(Subtask 6 * Subtask 7) + Subtask 1** or **(Subtask 3 * Subtask 4 * Subtask 5) + Subtask 1**. The scores for a subtask is determined by the system that is allocated to the subtask. The system to task scores are shown in Figure 29.

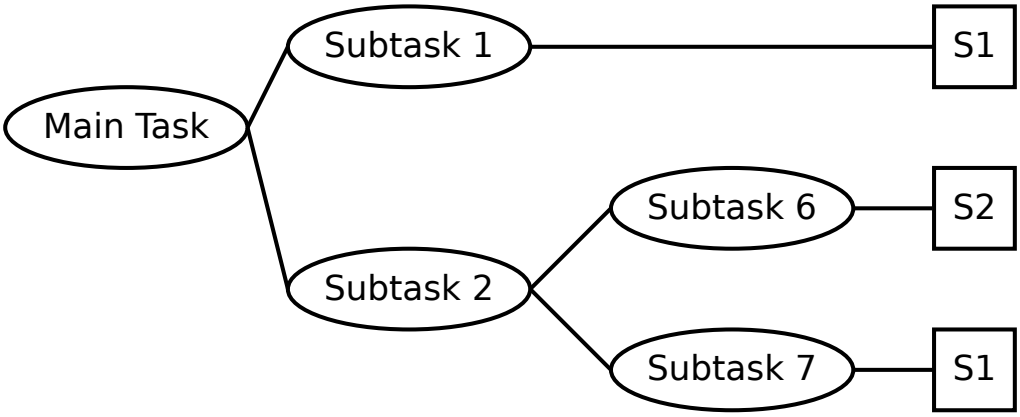


Figure 32: Canonical Example Architecture Alternative 1

The first architecture alternative has a metric A score of $(17 + 19) * 2 = 72$. The metric B score is $(34 * 55) + 2 = 1872$.

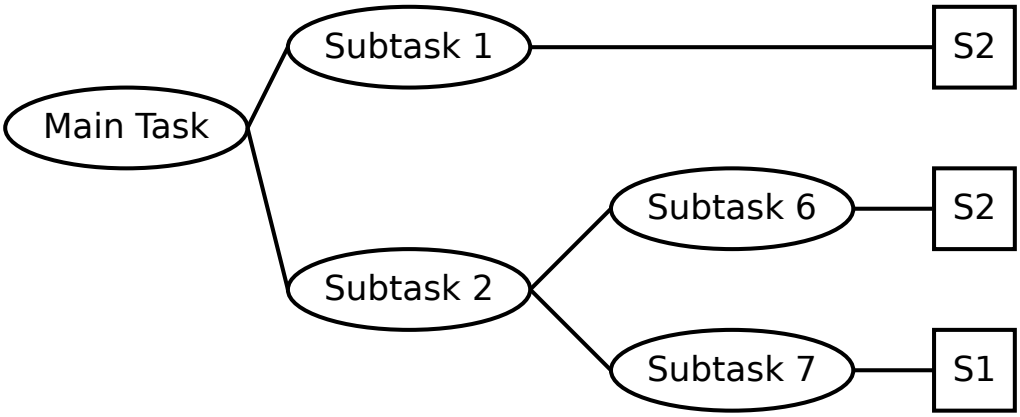


Figure 33: Canonical Example Architecture Alternative 2

The second architecture alternative has a metric A score of $(17 + 19) * 3 = 108$. The metric B score is $(34 * 55) + 3 = 1873$.

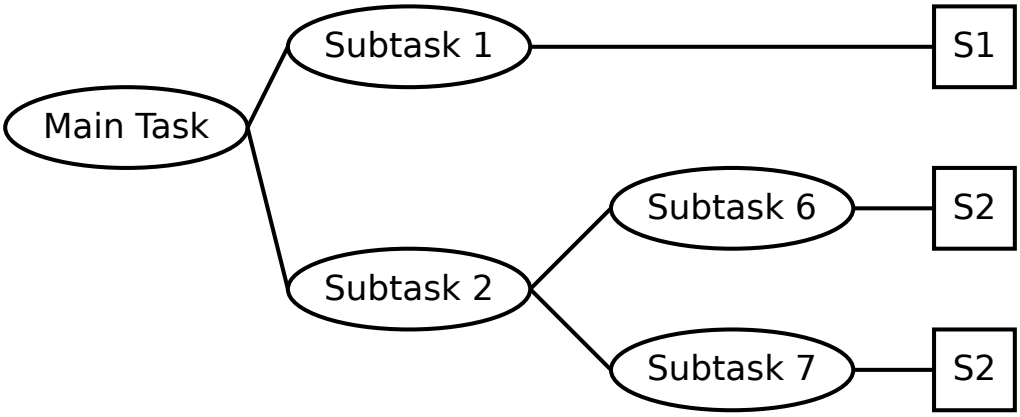


Figure 34: Canonical Example Architecture Alternative 3

The third architecture alternative has a metric A score of $(17 + 23) * 2 = 80$. The metric B score is $(34 * 89) + 2 = 3028$.

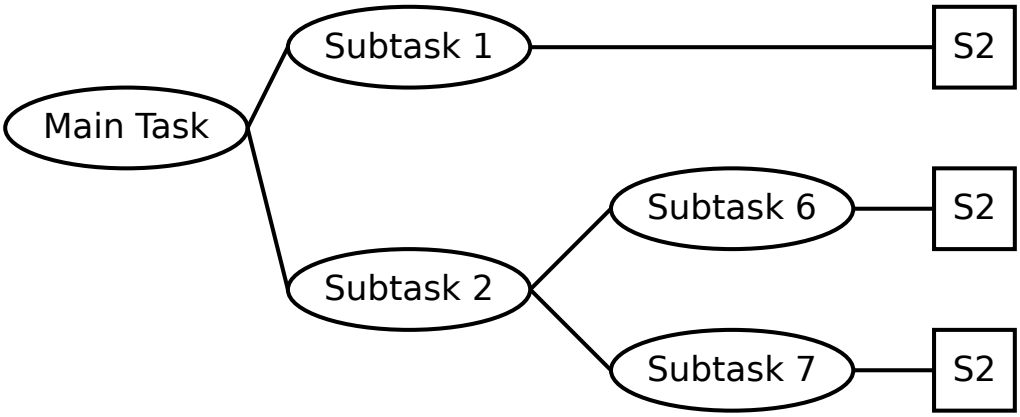


Figure 35: Canonical Example Architecture Alternative 4

The fourth architecture alternative has a metric A score of $(17 + 23) * 3 = 120$. The metric B score is $(34 * 89) + 3 = 3029$.

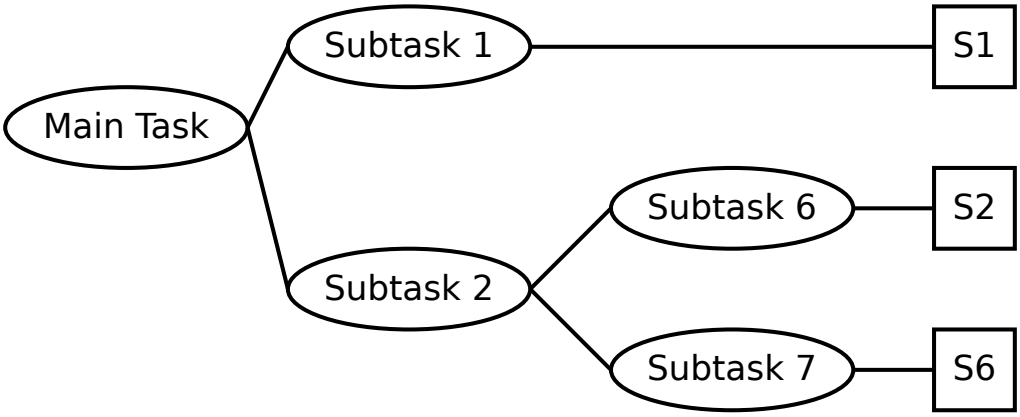


Figure 36: Canonical Example Architecture Alternative 5

The fifth architecture alternative has a metric A score of $(17 + 29) * 2 = 92$. The metric B score is $(34 * 144) + 2 = 4898$.

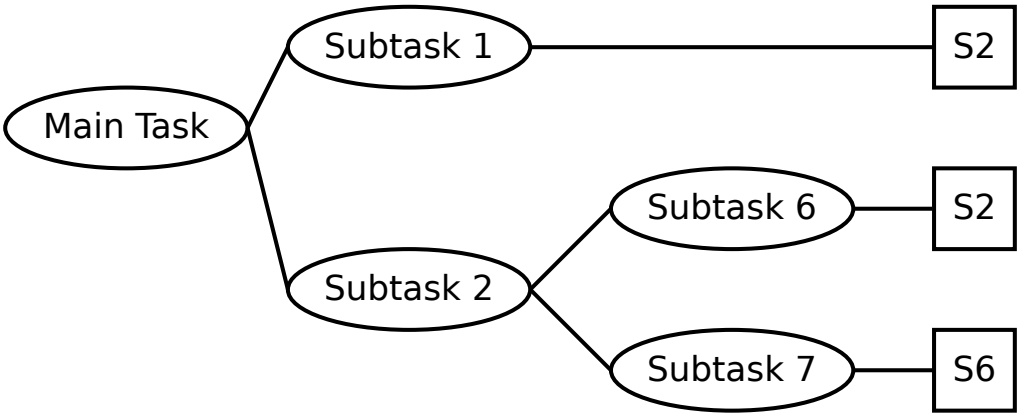


Figure 37: Canonical Example Architecture Alternative 6

The sixth architecture alternative has a metric A score of $(17+29)*3 = 138$. The metric B score is $(34 * 144) + 3 = 4899$.

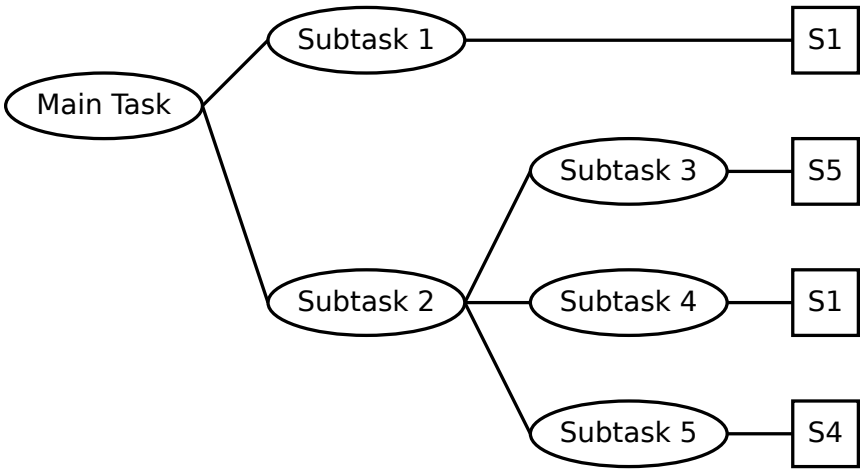


Figure 38: Canonical Example Architecture Alternative 7

The seventh architecture alternative has a metric A score of $(5 + 7 + 13) * 2 = 50$. The metric B score is $(5 * 8 * 21) + 2 = 842$.

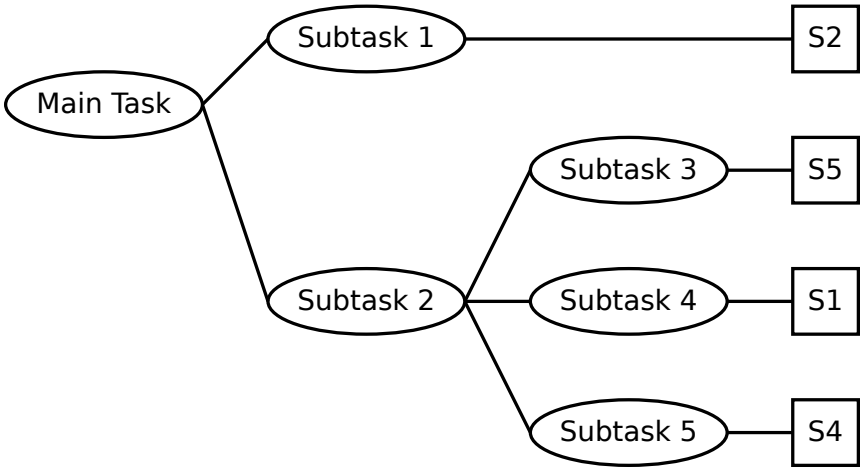


Figure 39: Canonical Example Architecture Alternative 8

The eighth architecture alternative has a metric A score of $(5 + 7 + 13) * 3 = 75$. The metric B score is $(5 * 8 * 21) + 3 = 843$.

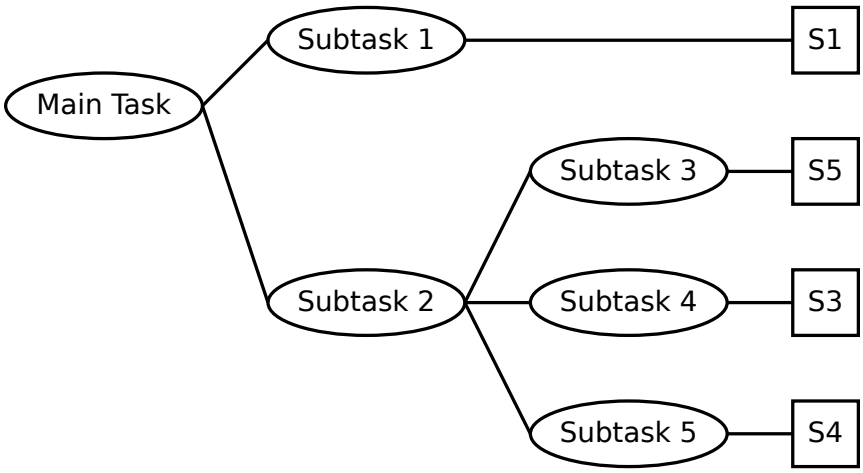


Figure 40: Canonical Example Architecture Alternative 9

The ninth architecture alternative has a metric A score of $(5 + 11 + 13) * 2 = 58$. The metric B score is $(5 * 13 * 21) + 2 = 1367$.

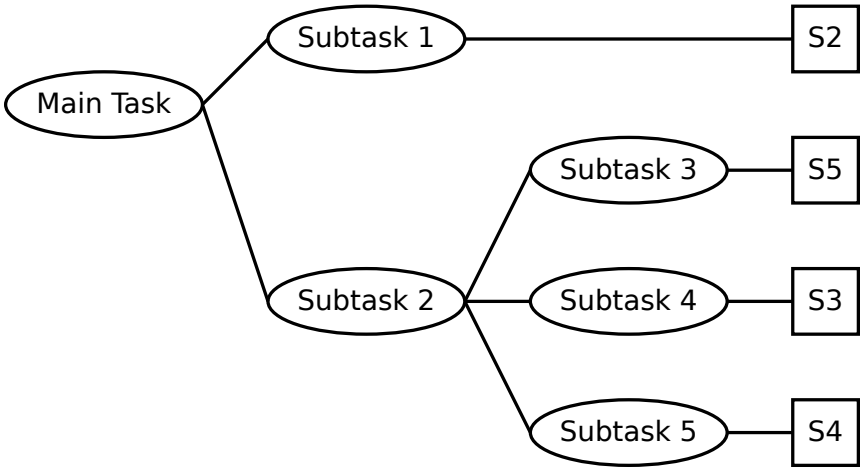


Figure 41: Canonical Example Architecture Alternative 10

The tenth architecture alternative has a metric A score of $(5 + 11 + 13) * 3 = 87$. The metric B score is $(5 * 13 * 21) + 3 = 1368$.

8.1.2 Portfolio View

Sometimes the analyst is interested in a different viewpoint on the architecture alternatives. Often there can be different systems allocated to the same task at different times while operating a systems of systems architecture. Another viewpoint is to look at the combined scores for a set of architecture alternatives with the same system portfolio. In the canonical example, there are eight different portfolios. The different portfolios are detailed in Table 4.

Table 4: Possible System Portfolios for the Canonical Example

S1	S2	S3	S4	S5	S6	# of Alts
x	x					3
	x					1
x	x				x	1
	x				x	1
x			x	x		1
x	x		x	x		1
x		x	x	x		1
	x	x	x	x		1

The canonical example was used to test RAAM concepts and the developed software. Since the canonical example has all of the characteristics of a larger and more realistic example, it can be used to add confidence to RAAM.

8.2 Proof of Concept: Suppression of Enemy Air Defenses

Once the method was developed with the canonical example, a more challenging and real world scenario was developed based on the suppression of enemy air defenses (SEAD). A SEAD scenario is used to test the method with different types of portfolios and task hierarchies. SEAD is defined as, "activity that neutralizes, destroys, or temporarily degrades surface-based enemy air defenses by destructive and/or disruptive means".[84] Surface-based enemy air defenses include engagement systems, sensor systems, and a command and control network. Example engagement systems are Surface to Air Missiles (SAM) and Anti-Aircraft Artillery (AAA). Early warning

and fire control radars are typically used as sensor systems. The enemy's portfolio of systems are often combined into an Integrated Air Defense System (IADS). The goal of the system of systems architecture is to provide a SEAD capability by nullifying the effectiveness of the enemy IADS.

The SEAD mission is growing in importance to the Department of Defense and approached 25% of the missions in Northern/Southern Watch. Bosnia was 32% of the missions. SEAD missions have varied efficiency depending on the force structure of the enemy. Simpler air defense networks are almost completely destroyed while more complex air defenses may only have twelve to twenty five percent of the assets destroyed. [12]

The SEAD mission is extremely challenging as it is designed to shape the battlefield. The difficulty translates into high cost in terms of both monetary and life. To properly model a SEAD capability, there must be attention to the complexity of a SEAD operation. Many different units must be coordinated to reduce the loss of life.

The Suppression of Enemy Air Defenses involves an architecture with the five properties of a system of systems.[126] The five properties are operational independence, managerial independence, geographical distribution, emergent behavior, and evolutionary development. Multiple services work together in a joint way to accomplish all elements of the Suppression of Enemy Air Defenses capability which provide managerial and operational independence. A Suppression of Enemy Air Defenses scenario by definition includes geographical distribution, which is only extended further by our communication abilities. The interactions between the different systems can create emergent behaviors. Due to the method of system procurement, any system of systems that was created for the Suppression of Enemy Air Defenses is evolutionary. In addition, the system of systems evolves with the changing threats.

Different portfolios of systems can be used to accomplish the SEAD mission. Examples of different types of portfolios for this scenario include portfolios based on:

carrier based aircraft (both manned and unmanned), special forces teams, submarine launched cruise missiles, and ballistic missiles as engagement systems. The decision maker is interested in choosing between these disparate portfolio types that may include novel systems. It is difficult to model all of these different portfolio types in one common framework. An agent based modeling method would require specialized models for each type of portfolio. Modeling such as Petri nets or other discrete event simulations run into a similar issue, although they can be used.

The capability to accomplish the suppression of enemy air defenses mission is described in a three level hierarchy. The SEAD example contains twenty two tasks and eleven different systems and is partially described in Figure 42. Not every system can do each task (i.e., some system combinations are infeasible), which reduces the number of architecture alternatives to 746,807,040. With four different mission dependent metrics of interest, 2,987,228,160 model evaluations are required to examine the entire decision space. The four mission dependent metrics are Probability of Success (P-success), Complexity, Time to Completion, and Maintainability. In addition, there were two mission independent metrics; cost and risk. The large number of independent model evaluations is suited to computation on cloud computing resources rather than thread based parallelization due to the number of processors required. As a demonstration, the software was enhanced to utilize the Amazon Elastic Compute Cloud [5], chosen for ease of use, quality of documentation, and software support.

The SEAD task hierarchy and system to task mapping was derived from the task hierarchy and system to task mapping defined in [56]. The model was adapted for the RAAM framework as part of the larger ARCHITECT project.

8.2.1 Development of the SEAD Task Hierarchy

The SEAD mission is described in a variety of military documents. Two documents are readily available: 1) JTTP (Joint Tactics, Techniques, and Procedures) for Joint

Suppression of Enemy Air Defenses (J-SEAD) [85] and 2) MCWP (Marine Corps Warfighting Publication) 3-22.2 Suppression of Enemy Air Defenses. [32] Neither document specifies a task hierarchy or engagement sequence for SEAD. MCWP 3-22.2 describes an engagement sequence for an Integrated Air Defense System (IADS) as comprised of:

1. Detect
2. Identify
3. Correlate/Track
4. Target Assignment
5. Weapons Control

The same engagement sequence can be used by a system of systems to accomplish the SEAD mission.

8.2.1.1 Detect

The **Detect** task is the first task in the task sequence that is used to provide the SEAD capability. The **Detect** task is primarily a command and control task. The **Detect** task has six different subtasks:

1. Reconcile Target Priorities
2. Determine Sensor Availability
3. Task Sensor
4. Wide Area Search
5. Fuse Sensor Data
6. Pass Warning/Location Data

The **Reconcile Target Priorities** task describes a command and control task. SEAD is used to allow for air superiority over the AOR/JOA (Area of Responsibility/Joint Operations Area). The target priorities from friendly operations are taken into account in this task.

The **Determine Sensor Availability** task describes a command and control task. The command and control node must determine which sensors are available for use.

Once the **Reconcile Target Priorities** task and the **Determine Sensor Availability** task are accomplished, the sensors must be given tasking in the **Task Sensor** task. The **Task Sensor** task is a command and control task.

The **Wide Area Search** task is accomplished by a sensor system. A search is conducted across the AOR/JOA (Area of Responsibility/Joint Operations Area).

The **Fuse Sensor Data** task is a command and control task. The sensor data from the **Wide Area Search** task is fused into the common operational picture.

The **Pass Warning/Location Data** task is a command and control task. The common operational picture is shared with the other participating systems.

8.2.1.2 Identify

The **Identify** task is the second task in the task sequence. This task is not broken down further. The targets of interest are identified in this task.

8.2.1.3 Correlate/Track

The **Correlate/Track** task is the third task in the task sequence. The opposing IADS system must be tracked. It is decomposed into three subtasks:

1. Manage Target Movement Data
2. Discriminate Launch/Support Systems from Decoys
3. Track Until Stopped

The **Manage Target Movement Data** task is a command and control task. The targets may be mobile launch platforms. Mobile launch platforms can have increased survivability if their movement is not detected. The target movement data is used in the following tasks.

The **Discriminate Launch/Support Systems from Decoys** task is accomplished by sensors. A common ruse in IADS is to have decoy systems that appear near identical to a variety of sensors. Actual launch systems or support systems are identified.

The **Track Until Stopped** task is a sensor task. Targets that have ceased movement may require less sensor time.

8.2.1.4 Target Assignment

The **Target Assignment** task is the fourth task in the task sequence. It is a command and control task. Different weapon systems must be assigned to the targets in the target list. The task consists of three subtasks:

1. Update Target List
2. Assess Engagement Capability
3. Assign Weapon/Target/Platform Selection

The **Update Target List** task is a command and control task. The new information from the **Correlate/Track** task is incorporated into the initial target list.

The **Assess Engagement Capability** task is a command and control task. The different engagement capabilities are assessed to provide guidance for the next task.

The **Assign Weapon/Target/Platform Selection** task is a command and control task. The different targets must be matched to the available engagement capability. A platform and weapon are assigned to a target.

8.2.1.5 Weapons Control

The **Weapons Control** task is the fifth task in the task sequence. The task is made of four different subtasks:

1. Engage to Destroy
2. Engage to Disrupt
3. Battle Damage Assessment
4. Remove from Target List

The **Engage to Destroy** task is one of two possible options for engaging the target. If the system is destroyed then it can not be used in an IADS. A destroyed target is hard to repair within the time frame of an attack.

The **Engage to Disrupt** task is one of two possible options for engaging the target. Depending on other friendly operations that are being supported by the SEAD mission, disrupting the IADS may be all that is required to accomplish the overall objectives.

The **Battle Damage Assessment** task is accomplished by sensor systems. Once a target is engaged, it is critical to determine the utility of the target to the enemy. The damage assessment attempts to measure the remaining capability of the target.

The **Remove from Target List** task is a command and control task. If the target is engaged in a satisfactory manner, the target is removed from the target list. The task sequence can be repeated once this task is accomplished.

The different tasks to accomplish a SEAD mission have been defined. In addition, the parent/child relationships between the tasks have been defined. The SEAD capability is provided by the five top level tasks, Detect, Identify, Correlate/Track, Target Assignment, and Weapons Control. Table 5 contains the mapping of task

Table 5: SEAD Task to Number Mapping

Task Number	Task Name
1.0	**Detect**
1.1	Reconcile Target Priorities
1.2	Determine Sensor Availability
1.3	Task Sensor
1.4	Wide Area Search
1.5	Fuse Sensor Data
1.6	Pass Warning/Location Data
2.0	**Identify**
3.0	**Correlate/Track**
3.1	Manage Target Movement Data
3.2	Discriminate Launch/Support Systems from Decoys
3.3	Track Until Stopped
4.0	**Target Assignment**
4.1	Update Target List
4.2	Assess Engagement Capability
4.3	Assign Weapon/Target/Platform Selection
5.0	**Weapons Control**
5.1	Engage to Destroy
5.2	Engage to Disrupt
5.3	Battle Damage Assessment
5.4	Remove from Target List

number to task name. The task numbers are used as a short hand in later tables in this manuscript.

8.3 Development of the SEAD System to Task Mappings

Before we can map systems to tasks, we must define which systems are available to accomplish the lowest level tasks. All of the lowest level tasks are not at the same level. Because the **Identify** task has not been decomposed and a single system can accomplish the task, it is a task at a higher level of decomposition than the rest of the following tasks.

Multiple systems can accomplish each task. The SEAD model has eleven different systems that are considered. The eleven systems are:

1. CVN

2. Central C2
3. Intel Satellite
4. X-47B
5. F/A-18
6. AH-64
7. EA-6B
8. E-2
9. M1
10. DDG
11. SOF

These systems and the tasks they can perform are subsequently described.

CVN Aircraft carrier ships are designated with CV and the N stands for nuclear powered. For the SEAD model, a nuclear powered aircraft carrier in the Nimitz class is used. The following information is drawn from [138]. Aircraft carriers provide power projection, forward presence, humanitarian assistance, deterrence, sea control, and maritime security. A Nimitz class carrier is designed for a fifty year service life which includes one refueling of the nuclear reactors. There are two nuclear reactors driving four propellers. The carrier is crewed with approximately 5000 people and over sixty aircraft. For the SEAD mission, the aircraft carrier is used for its command and control capabilities. The CVN system can be used for ten of the SEAD tasks:

- 1.1 Reconcile Target Priorities
- 1.2 Determine Sensor Availability

- 1.3 Task Sensor
- 1.5 Fuse Sensor Data
- 1.6 Pass Warning/Location Data
- 3.1 Manage Target Movement Data
- 4.1 Update Target List
- 4.2 Assess Engagement Capability
- 4.3 Assign Weapon/Target/Platform Selection
- 5.4 Remove from Target List

Central C2 The Central C^2 (Command and Control) system is designed to model a theater command and control node. The command and control node is located in a forward operating base or an Unified Combatant Command such as CENTCOM, EUCOM, or PACOM. The Central C^2 node communicates with the other systems using line of sight radio or satellite communications. An example of a component of a Command and Control system is holographic technology to visualize the battlespace. The Central C^2 system can be used for ten of the SEAD tasks:

- 1.1 Reconcile Target Priorities
- 1.2 Determine Sensor Availability
- 1.3 Task Sensor
- 1.5 Fuse Sensor Data
- 1.6 Pass Warning/Location Data
- 3.1 Manage Target Movement Data

- 4.1 Update Target List
- 4.2 Assess Engagement Capability
- 4.3 Assign Weapon/Target/Platform Selection
- 5.4 Remove from Target List

Intel Satellite An Intelligence Satellite is used to gather information about the battlefield. This system is representative of different national technical means[1] to gather intelligence using satellites. This would include Synthetic Aperture RADAR (SAR) imagery, visible spectrum imagery, measurement and signature intelligence (MASINT), and communications intelligence (SIGINT). An example photo of a recent intelligence satellite is not available, but rumors persist of an intelligence satellite series based on the same satellite bus as the Hubble Space Telescope. The Intelligence Satellite can be used for five of the SEAD tasks:

- 1.4 Wide Area Search
- 2.0 Identify
- 3.2 Discriminate Launch/Support Systems from Decoys
- 3.3 Track Until Stopped
- 5.3 Battle Damage Assessment

X-47B The X-47B is an Unmanned Combat Aerial Vehicle (UCAV) that is part of the Unmanned Combat Air System Demonstration (UCAS-D). The aircraft is tail less and designed to be carrier based. The X-47B is representative of future unmanned carrier based aviation. [112] The X-47B can be used for seven of the SEAD tasks:

[1]The term was first used in the SALT I treaty [145] and describes a variety of sensors, including satellites.

- 1.4 Wide Area Search
- 2.0 Identify
- 3.2 Discriminate Launch/Support Systems from Decoys
- 3.3 Track Until Stopped
- 5.1 Engage to Destroy
- 5.2 Engage to Disrupt
- 5.3 Battle Damage Assessment

F/A-18 The F/A-18 (Hornet) is an all weather fighter and attack aircraft. The F/A-18 can accomplish both attack (such as interdiction or close air support) and fighter missions. The new Super Hornet variant is capable in air superiority, fighter escort, reconnaissance, aerial refueling, close air support, air defense suppression, and day/night precision strike. It can carry precision guided munitions. [142] The F/A-18 can be used for five of the SEAD tasks:

- 1.4 Wide Area Search
- 2.0 Identify
- 3.2 Discriminate Launch/Support Systems from Decoys
- 5.1 Engage to Destroy
- 5.3 Battle Damage Assessment

AH-64 The AH-64 (Apache) is a four bladed attack helicopter with two engines. The helicopter is designed to conduct rear, close, and shaping missions. The AH-64 can provide deep precision strike against relocatable targets and reconnaissance in all

weather conditions. The AH-64 is attached to a heavy division or corps and crewed by two pilots. [144] The AH-64 can be used for five of the SEAD tasks:

- 1.4 Wide Area Search
- 2.0 Identify
- 3.2 Discriminate Launch/Support Systems from Decoys
- 5.1 Engage to Destroy
- 5.3 Battle Damage Assessment

EA-6B The EA-6B is designed for protecting strike aircraft, ground troops, and ships. The EA-6B can jam RADAR, electronic data links, and communications. The aircraft has a side by side cockpit arrangement with two engines. The aircraft is also capable of kinetic attacks using RADAR homing missiles. The EA-6B's primary mission is the suppression of enemy air defenses. [141] The EA-6B can be used for five of the SEAD tasks:

- 1.4 Wide Area Search
- 2.0 Identify
- 3.2 Discriminate Launch/Support Systems from Decoys
- 5.2 Engage to Disrupt
- 5.3 Battle Damage Assessment

E-2 The E-2 (Hawkeye) is a battle management aircraft with early warning and command and control in all weather conditions. The E-2 has two engines and a crew of five. A distinctive feature is the twenty four foot diameter radar rotodome on the top of the fuselage. The E-2 is capable of surface surveillance coordination, air

interdiction, counter air control, close air support coordination, time critical strike coordination, search and rescue coordination and communications, and a communications relay. [140] The E-2 can be used for six of the SEAD tasks:

- 1.5 Fuse Sensor Data
- 1.6 Pass Warning/Location Data
- 3.1 Manage Target Movement Data
- 3.2 Discriminate Launch/Support Systems from Decoys
- 3.3 Track Until Stopped
- 4.1 Update Target List

M1 The M1 Abrams is a heavily armored tank typically used against ground forces. It has a nuclear, biological, and chemical (NBC) protection system. The tank is also effective at night due to its thermal sensors. [143] The M1 can be used for three of the SEAD tasks:

- 2.0 Identify
- 5.1 Engage to Destroy
- 5.3 Battle Damage Assessment

DDG The DDG-1000 Zumwalt class destroyer is a ship that provides multi-mission offensive and defensive capabilities. The DDG-1000 can operate independently or as part of a carrier strike group. The ship provides anti-air, anti-submarine, and anti-surface capabilities. The DDG-1000 can be used for vertical take off vehicles such as manned or unmanned helicopters. For the SEAD model, the DDG-100 is providing surface fires to destroy air defense systems on land. [139] The DDG-100

is representative of a future destroyer. The DDG can be used for one of the SEAD tasks:

- 5.1 Engage to Destroy

SOF Special Operations Forces (SOF) are used in all environments but specialize in denied environments. Special Operations Forces are designed for eleven activities as follows: direct action, special reconnaissance, counterproliferation of weapons of mass destruction, counterterrorism, unconventional warfare, foreign internal defense, security force assistance, counterinsurgency, information operations, military information support operations, and civil affairs operations. All branches of the military have Special Operations Forces. [86] Kelley advocates using Special Operations Forces for sabotage and intelligence activities in the future. [89] The SEAD mission can require both of those activities. For the SEAD mission, Special Operation Forces are used for reconnaissance and engaging of enemy air defense systems. The SOF can be used for six of the SEAD tasks:

- 2.0 Identify
- 3.2 Discriminate Launch/Support Systems from Decoys
- 3.3 Track Until Stopped
- 5.1 Engage to Destroy
- 5.2 Engage to Disrupt
- 5.3 Battle Damage Assessment

The a summary of the above information concerning possible system to task mappings is shown in Table 6.

Table 6: SEAD Possible System to Task Mappings

System	1.1	1.2	1.3	1.4	1.5	1.6	2.0	3.1	3.2	3.3	4.1	4.2	4.3	5.1	5.2	5.3	5.4
CVN	X	X	X		X	X		X			X	X	X				X
Central C2	X	X	X		X	X		X			X	X	X				X
Intel Satellite				X			X		X	X						X	
X-47B				X			X		X	X				X	X	X	
F/A-18				X			X		X					X		X	
AH-64				X			X		X					X		X	
EA-6B				X			X		X						X	X	
E-2					X	X		X	X	X	X						
M1							X							X		X	
DDG														X			
SOF							X		X	X				X	X	X	

The task hierarchy and system to task mappings are summarized in Figure 42.

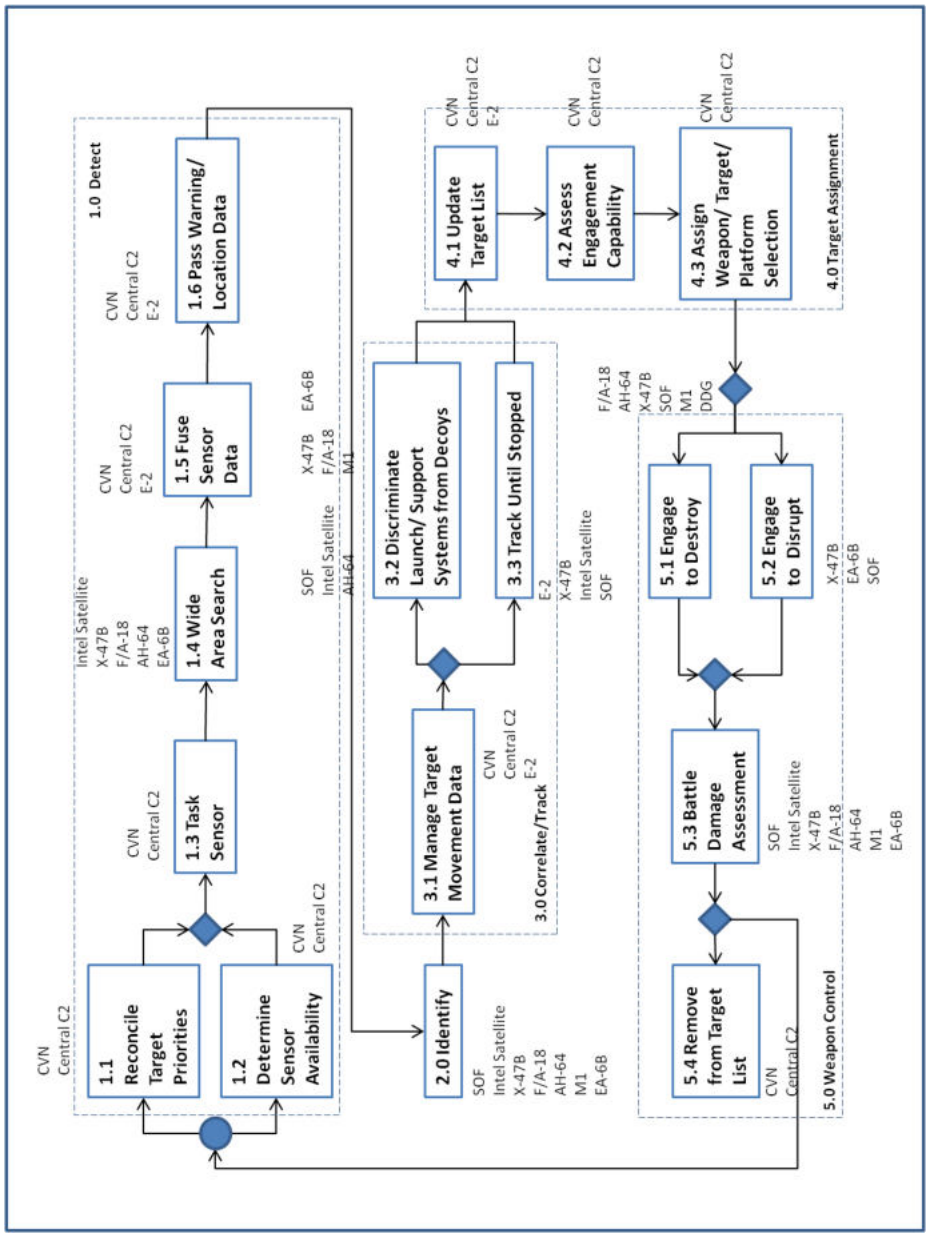


Figure 42: The SEAD example task hierarchy with possible systems [57]

8.4 Development of the Metric Scores for Each Task

8.4.1 Mission Independent Metrics

For the Suppression of Enemy Air Defenses mission, there are two mission independent metrics used in the model. The first mission independent metric is the cost of

Table 7: System Mission Independent Scores - Cost

System	**Cost**
CVN	10000.0
Central C2	15.0
Intel Satellite	3000.0
X-47B	80.0
F/A-18	68.0
AH-64	20.0
EA-6B	70.0
E-2	100.0
M1	0.25
DDG	2000.0
SOF	20.0

the system. In the case of this model, the cost includes acquisition costs and operational costs for the SEAD mission. The cost scores are summarized in Table 7. The second mission independent metric is the development risk of the system. Current systems such as the F/A-18, AH-64, or EA-6B have low risk while systems that are in development such as the DDG-1000 or X-47B are higher risk. The risk scores are summarized in Table 8. The RAAM model inputs for the mission independent metrics are included in the SEAD RAAM model (D).

Table 8: System Mission Independent Scores - Risk

System	Risk
CVN	0.10
Central C2	0.20
Intel Satellite	0.40
X-47B	0.80
F/A-18	0.05
AH-64	0.05
EA-6B	0.05
E-2	0.05
M1	0.05
DDG	0.60
SOF	0.20

8.4.2 Mission Dependent Metrics

The Suppression of Enemy Air Defenses mission has four mission dependent metrics. A mission dependent metric score for the entire SEAD mission is created from a set of system to task to metric scores. The four metrics are:

- Probability of Success (P-success)
- Complexity (Complexity)
- Time to completion (Time-to-completion)
- Maintainability (Maintainability)

8.4.3 Execution Models for Mission Dependent Metrics

The following paragraphs summarize the inputs to the SEAD model given in Appendix D. The SEAD mission does not have any transformation functions and is solely composed of aggregation functions.

The probability of success is the probability that the completed SEAD mission will be successful. Probability of success for a system to task pairing is the probability of success of the given system in accomplishing the task. The overall probability of success is calculated by using a product of all of the system to task pairing scores,

shown in Equation 8 where s_{ij} is the score (s) for the probability of success for the i^{th} system and j^{th} task.

$$\text{Probability of Success} = \prod s_{ij} \tag{8}$$

Complexity is an overall estimate of the complexity of accomplishing the required tasks for the SEAD mission. A complexity score for a system to task pairing is the contribution of complexity to the overall complexity score due to the system to task pairing. The complexity score calculation is more complex than the probability of success score calculation. Equation 9 describes the complexity score calculation.

$$\begin{aligned}
\text{Complexity} &= \prod (\text{Conduct SEAD subtasks}) \\
\text{Detect Complexity} &= \sum (\text{Detect subtasks}) \\
\text{Correlate and Track Complexity} &= \max (\text{Correlate and Track subtasks}) \\
\text{Target Assignment Complexity} &= \min (\text{Target Assignment subtasks}) \\
\text{Weapon Control Complexity} &= \sum (\text{Weapon Control subtasks})
\end{aligned} \tag{9}$$

Time to completion is the overall time to complete the SEAD mission objectives. The time to completion score for a system to task pairing is the time required to complete the task by the chosen system. In the case of this SEAD mission, the different tasks must be done in a serial fashion, so the overall time to completion is the sum of the times required to accomplish the tasks. Equation 10 shows the equation for time to completion. For time to completion, s_{ij} is the score (s) the i^{th} system and j^{th} task.

$$\text{Time to Completion} = \sum s_{ij} \tag{10}$$

Maintainability is the last metric that was modeled. The maintainability metric describes the overall maintainability of the SEAD mission. As assets are used, they

are degraded and require maintenance. The maintainability score for a for a system to task pairing is the maintainability of a system when accomplishing the task. The maintainability is the lowest maintainability of the set of systems accomplishing the mission. Equation 11 shows the equation for the maintainability. For maintainability, s_{ij} is the score (s) for the i^{th} system and j^{th} task.

$$\text{Maintainability} = \min s_{ij} \tag{11}$$

8.4.4 Mission Dependent Metric Scores for SEAD

8.4.4.1 CVN

The CVN metric scores are shown in Table 9.

Table 9: CVN Metric Scores

Tasks	Probability of success	Complexity	Time to completion	Maintainability
Reconcile Target Priorities	0.98	5	5	1
Determine Sensor Availability	0.99	5	12	4
Task Sensor	0.98	1	22	4
Fuse Sensor Data	0.99	3	16	3
Pass Warning and Location Data	0.97	3	13	2
Manage Target Movement Data	0.98	4	15	3
Update Target List	0.99	1	21	8
Assess Engagement Capability	0.99	2	16	5
Assign Weapon and Platform	0.995	5	18	4
Remove from Target List	0.96	2	14	8

8.4.4.2 Central C2

The metric scores are shown in Table 10.

Table 10: Central C^2 Metric Scores

Tasks	Probability of success	Complexity	Time to completion	Maintainability
Reconcile Target Priorities	0.98	4	9	10
Determine Sensor Availability	0.95	5	7	8
Task Sensor	0.99	2	15	2
Fuse Sensor Data	0.97	2	12	4
Pass Warning and Location Data	0.99	1	12	6
Manage Target Movement Data	0.95	5	8	6
Update Target List	0.98	1	19	4
Assess Engagement Capability	0.98	1	22	3
Assign Weapon and Platform	0.99	3	12	3
Remove from Target List	0.95	3	11	9

8.4.4.3 Intel Satellite

The metric scores are shown in Table 11.

Table 11: Intel Satellite Metric Scores

Tasks	Probability of success	Complexity	Time to completion	Maintainability
Wide Area Search	0.87	5	7	2
Discriminate Decoys	0.80	5	15	6
Track Until Stopped	0.95	2	15	4
Battle Damage Assessment	0.85	2	19	8
Identify	0.85	1	18	10

8.4.4.4 X-47B

The metric scores are shown in Table 12.

Table 12: X-47B Metric Scores

Tasks	Probability of success	Complexity	Time to completion	Maintainability
Wide Area Search	0.90	1	12	10
Identify	0.85	4	21	7
Discriminate Decoys	0.90	1	15	7
Track Until Stopped	0.95	3	9	1
Engage to Destroy	0.80	2	12	2
Engage to Disrupt	0.90	5	19	8
Battle Damage Assessment	0.99	1	20	4

8.4.4.5 F/A-18

The metric scores are shown in Table 13.

Table 13: F/A-18 Metric Scores

Tasks	Probability of success	Complexity	Time to completion	Maintainability
Wide Area Search	0.90	4	8	6
Discriminate Decoys	0.86	5	17	5
Engage to Destroy	0.97	2	12	10
Battle Damage Assessment	0.76	5	12	4
Identify	0.9	4	7	2

8.4.4.6 AH-64

The metric scores are shown in Table 14.

Table 14: AH-64 Metric Scores

Tasks	Probability of success	Complexity	Time to completion	Maintainability
Wide Area Search	0.80	2	4	5
Discriminate Decoys	0.96	2	10	10
Identify	0.79	5	15	3
Engage to Destroy	0.87	5	5	5
Battle Damage Assessment	0.98	1	16	1

8.4.4.7 EA-6B

The metric scores are shown in Table 15.

Table 15: EA-6B Metric Scores

Tasks	Probability of success	Complexity	Time to completion	Maintainability
Wide Area Search	0.90	4	3	10
Identify	0.999	1	14	4
Discriminate Decoys	0.80	5	8	6
Engage to Destroy	0.94	4	3	5
Engage to Disrupt	0.95	2	21	3
Battle Damage Assessment	0.98	1	19	2

8.4.4.8 E-2

The metric scores are shown in Table 16.

Table 16: E-2 Metric Scores

Tasks	Probability of success	Complexity	Time to completion	Maintainability
Fuse Sensor Data	0.98	1	8	10
Pass Warning and Location Data	0.99	2	17	3
Manage Target Movement Data	0.99	1	7	2
Track Until Stopped	0.98	2	5	10
Update Target List	0.97	5	17	4

8.4.4.9 M1

The metric scores are shown in Table 17.

Table 17: M1 Metric Scores

Tasks	Probability of success	Complexity	Time to completion	Maintainability
Discriminate Decoys	0.70	2	7	4
Engage to Destroy	0.80	3	14	7
Battle Damage Assessment	0.99	4	19	3
Identify	0.999	1	21	6

8.4.4.10 DDG

The metric scores are shown in Table 18.

Table 18: DDG Metric Scores

Tasks	Probability of success	Complexity	Time to completion	Maintainability
Engage to Destroy	0.99	1	12	3

8.4.4.11 SOF

The metric scores are shown in Table 19.

Table 19: SOF Metric Scores

Tasks	Probability of success	Complexity	Time to completion	Maintainability
Identify	0.98	1	4	8
Discriminate Decoys	0.76	4	11	4
Track Until Stopped	0.98	3	15	8
Engage to Destroy	0.99	5	10	5
Engage to Disrupt	0.99	5	19	6
Battle Damage Assessment	0.99	2	17	2

8.5 SEAD Problem Discussion

With a larger architecture alternative space to explore, it becomes difficult to manage the resulting data. In the Suppression of Enemy Air Defenses example, there are three quarters of a billion architecture portfolios with four metric scores per alternative. Common tools are not designed to handle that many data points. If the data can be loaded in the first place, it routinely crashes when attempting to explore the space. Two alternative methods for dealing with the large data sets were developed which are using a portfolio view of the problem and keeping the top 'n' best architecture alternatives.

The portfolio view of a system of systems architecture alternative was introduced with the canonical example. A unique portfolio is defined by a unique set of systems. The Suppression of Enemy Air Defenses example has 1266 different portfolios. The different architecture alternatives with the same set of constituent systems have their metric scores combined in some manner. For the purposes of demonstration, we are reporting the average value for each of the metrics for the portfolio. By storing the number of architecture alternatives that contributed to the average, we can update the average using the formula $Ave_{n+1} = Ave_n + \frac{score - Ave_n}{n+1}$. Also reported are the number of architecture alternatives that have been combined (n) and the number of systems in the portfolio. The approximately three fourths of a billion alternatives are still evaluated, but the data storage requirements drops to storing data on only 1266

different portfolios. Initial down selection can occur with the portfolio information. A subset of portfolios can be run and the full data set stored for the interesting portfolios. Other modeling methods often take a portfolio view of a system of systems architecture alternative and the resulting scores can be better compared.

The results of the portfolio view are shown in Appendix E.

8.6 SEAD Example Visualization

The following section will discuss two uses of a visualization environment to aid decision makers. The first use will cover an analysis that includes all of the portfolios in the Suppression of Enemy Air Defenses example. The second use will cover an analysis that is drilling down into one of the portfolios. JMP [82] is used to create the visualizations. The four mission dependent metrics of probability of success, time to completion, maintainability, and complexity were used for both visualization uses.

8.6.1 All Portfolios

The data generated by RAAM can be used to gain insight and understanding into system of systems architecture trades. The suppression of enemy air defenses example has 1266 different possible portfolios. Since RAAM calculates the scores for each architecture alternative in the set of architecture portfolios for a portfolio, the scores had to be combined in some way. In this case, each of the four metrics were combined by taking the average. The running average equation was used to combine the different scores. The following analysis is using the average scores for each of the metrics for each of the 1266 portfolios. The goal of the portfolio analysis is to downselect from the complete set of possible portfolios to a small subset of portfolios to carry forward for more detailed analysis. Figure 43 shows the distributions of the average portfolio scores for the four different metrics. In addition, Figure 43 shows the location of quantiles and moments of the distribution. The average probability of success metric score has a distribution that is approximately normal with a mean of

42.23% probability of success with a standard deviation of 3.9% probability of success. The best performing portfolio has a probability of success of 54.4%. The worst performing portfolio has a probability of success of 30.4%. Overall the chosen SEAD mission was challenging for all of the portfolios when probability of success was taken into account. The average complexity metric score has a skewed distribution. The average complexity metric score had a mean of 0.808 and a standard deviation of 0.279. The most complex portfolio had a complexity metric score of 1.913. The least complex portfolio had a complexity metric score of 0.210. The complexity scores are relative and are used for comparison. The average time to completion metric score is becoming a more multi-modal distribution. The mean time to completion metric score was 229.56 minutes and the standard deviation of the distribution was 10.76 minutes. The fastest portfolio has a time to completion score of 199 minute. The slowest portfolio has a time to completion score of 260 minutes. The average maintainability metric score had a roughly uniform distribution except the outlier at a maintainability score of 1.0. The mean of the average maintainability metric score was 1.33 and the standard deviation of the distribution was 0.31. The maximum possible score for maintainability is 2.0 with a variety of portfolios reaching the maximum score. The lowest possible maintainability score was 1.0, with over one quarter of the portfolios reaching the lowest possible maintainability score.

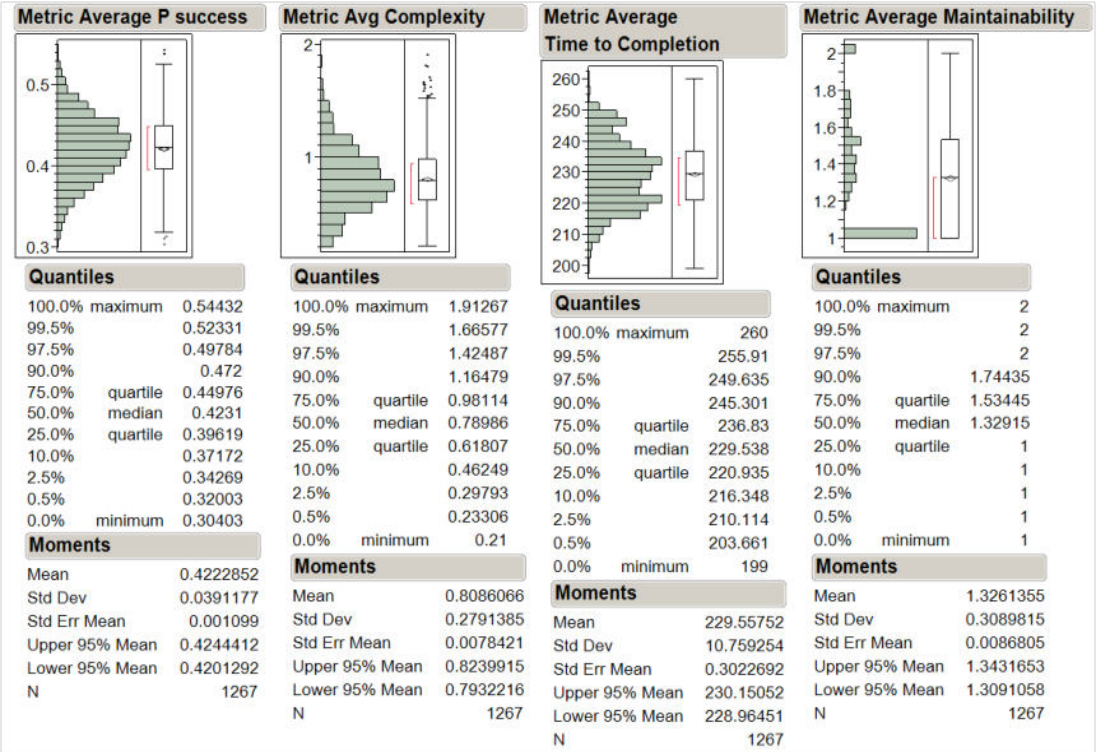


Figure 43: Distributions of the SEAD Portfolio Averages

The same data can be visualized in a scatter plot matrix. A scatter plot is "a powerful approach to exploratory data analysis" [18]. A scatter plot matrix is a collection of scatter plots that are arranged so that row or column of scatter plots share an axis. Each variable (in this case the average metric scores) is shown against the other variables. Each individual system of systems portfolio is represented by one point on each scatter plot in the scatter plot matrix shown in Figure 44. A scatter plot matrix can be used to quickly determine correlation between variables and also can be used to filter to a desired region.

The average maintainability metric score is non-linear and many portfolios collect in the minimum or maximum values. This is the horizontal lines in the bottom three scatter plots in Figure 44. The scatter plots show areas of high density of points, which corresponds to regions of the metric score space that have a large number of portfolios.

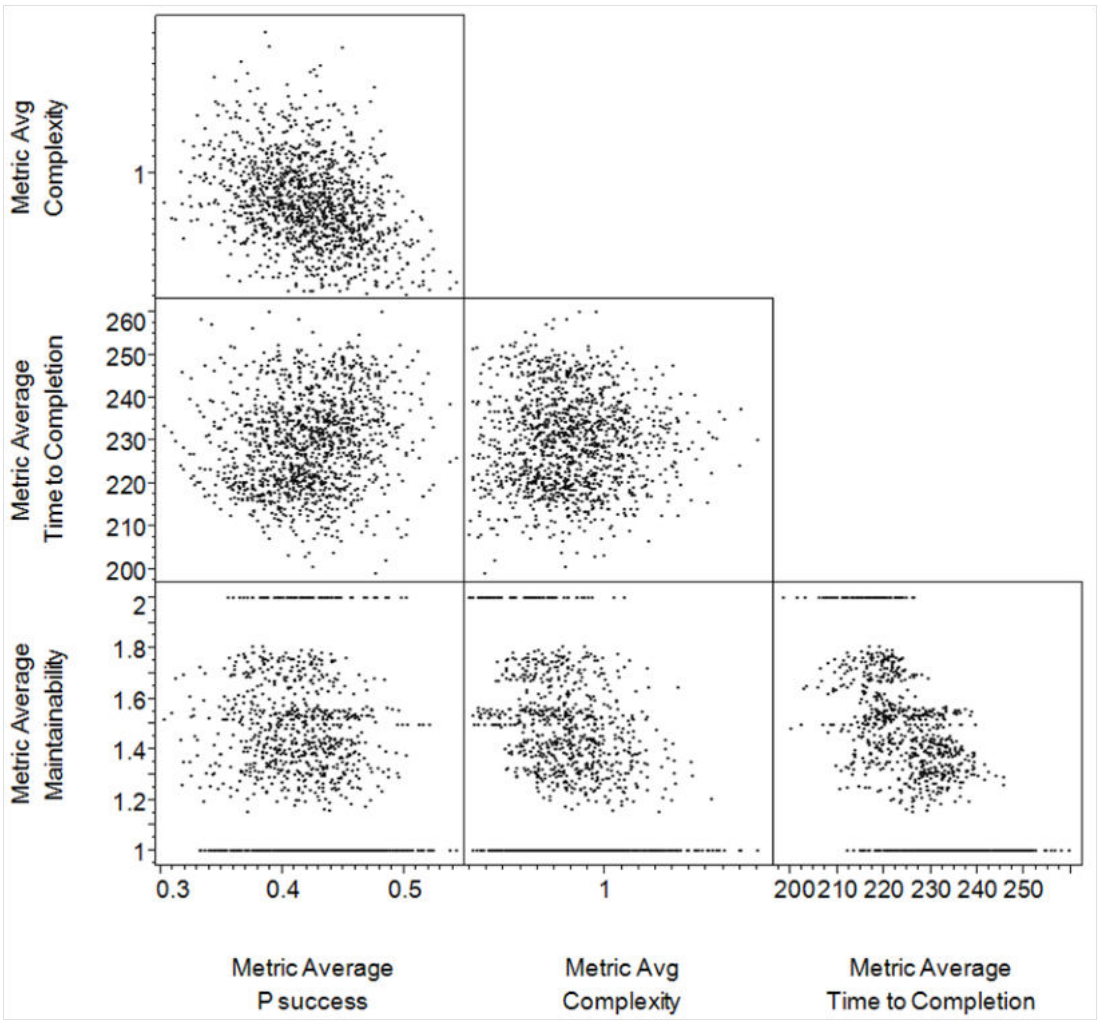


Figure 44: Scatter Plot Matrix of the Four Metrics for SEAD

In order to discover what is driving the maintainability clusters, the portfolios can be colored by their maintainability scores. Figure 45 shows the same scatter plot matrix as Figure 44 but with color. The colors are chosen such that highly maintainable portfolios are green, moderately maintainable portfolios are yellow, and hard to maintain portfolios are red. By using the colorization, the reader can see that portfolios with high time to completion scores are often harder to maintain. Easier to maintain systems often have low times to completion.

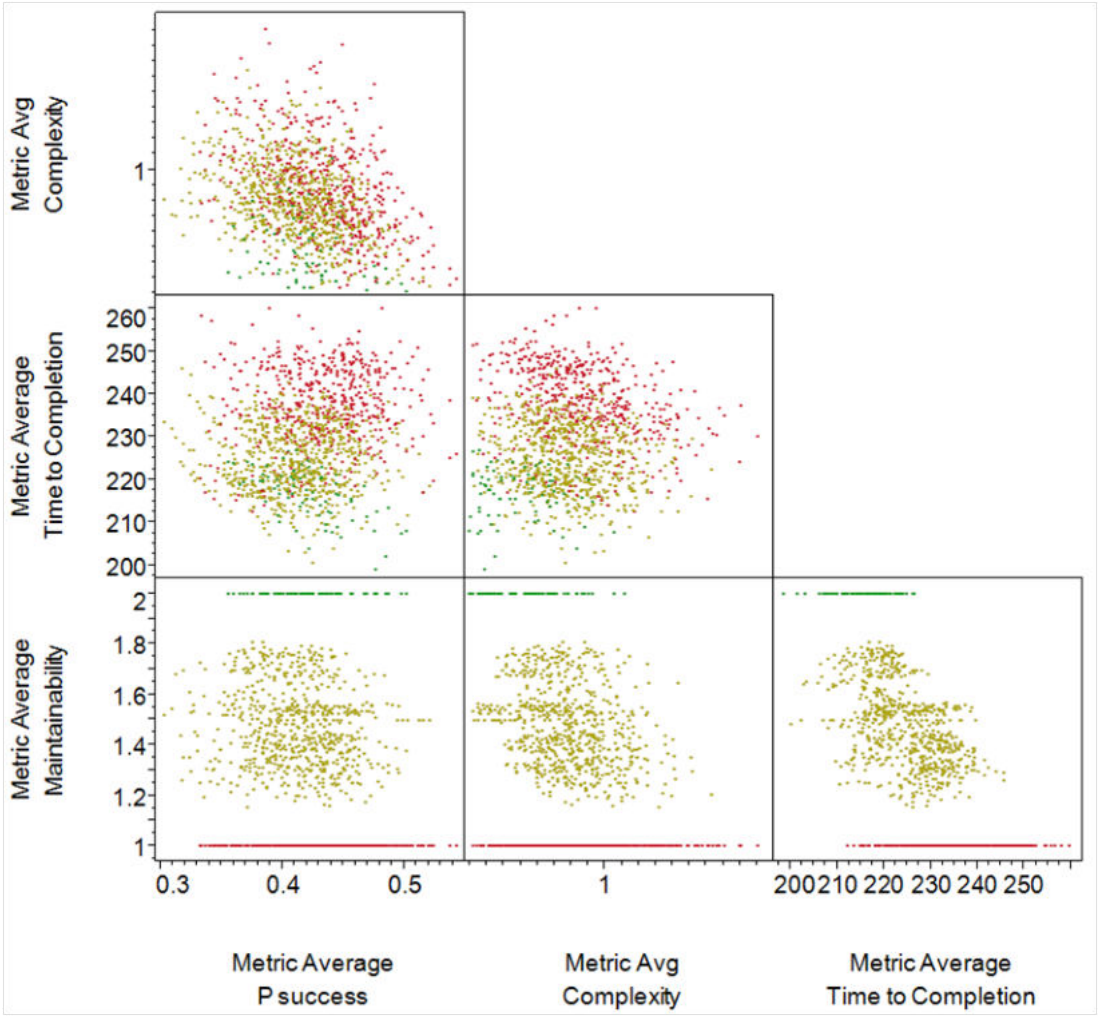


Figure 45: Colored Scatter Plot Matrix for the Four Metrics for SEAD

In order to explore the different locations of maintainability clusters, the portfolios with the lowest maintainability are highlighted in Figure 46. The red dots represent the portfolios with a maintainability score of 1.0. The portfolios that score low in maintainability are clustered toward higher time to completion scores but occur at all complexity levels.

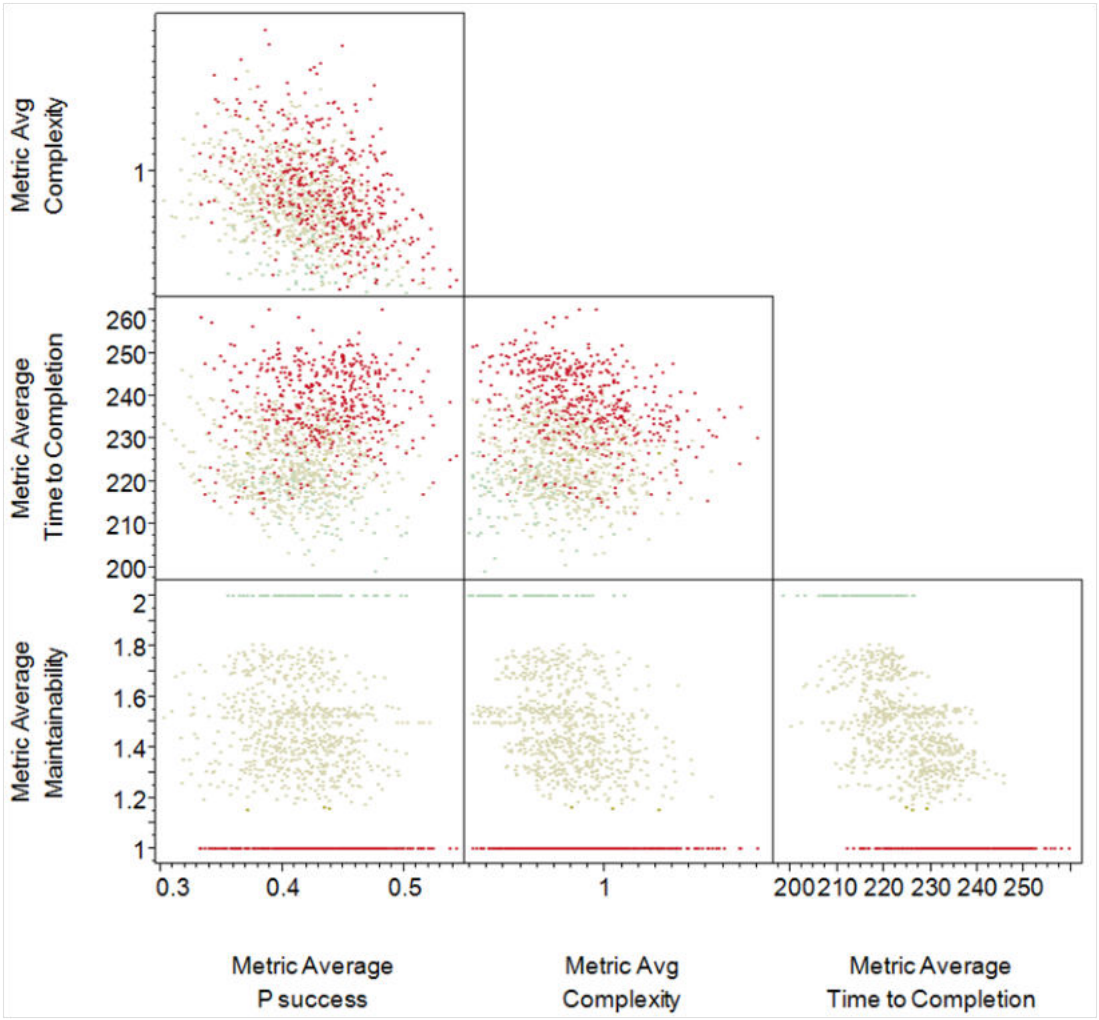


Figure 46: Scatter Plot Matrix of the SEAD Metrics with Low Maintainability Highlighted

The decision maker will be interested in which systems cause the low maintainability scores. In each portfolio there are systems that are either included or excluded. By looking at the frequency of the inclusion or exclusion of different systems, the decision maker can see which system is the culprit. In Figure 47 the frequencies for the Central C^2 and CVN system are shown in an allocation distribution. A level of 1 means that the system was included in that portfolio. A level of 2 means that the system was not included in that portfolio. Both the Central C^2 and CVN systems are both used approximately two thirds of the portfolios. They may have been used

together or separately in the portfolios. The length of the bars shows the relative number of portfolios that each type of system was included or excluded from. The dark green portions of the allocation distributions show portfolios with low maintainability scores of 1.0 that had the system either included (1), or excluded (0). It is observed that almost all of the portfolios with the lowest maintainability have the CVN system and very few use the Central C^2 system. For all other systems, there seems to be no relation between the inclusion of the system in the portfolio and the portfolio receiving a low score in maintainability. This matches intuition as the CVN is the most maintenance intensive system in the set of possible systems.

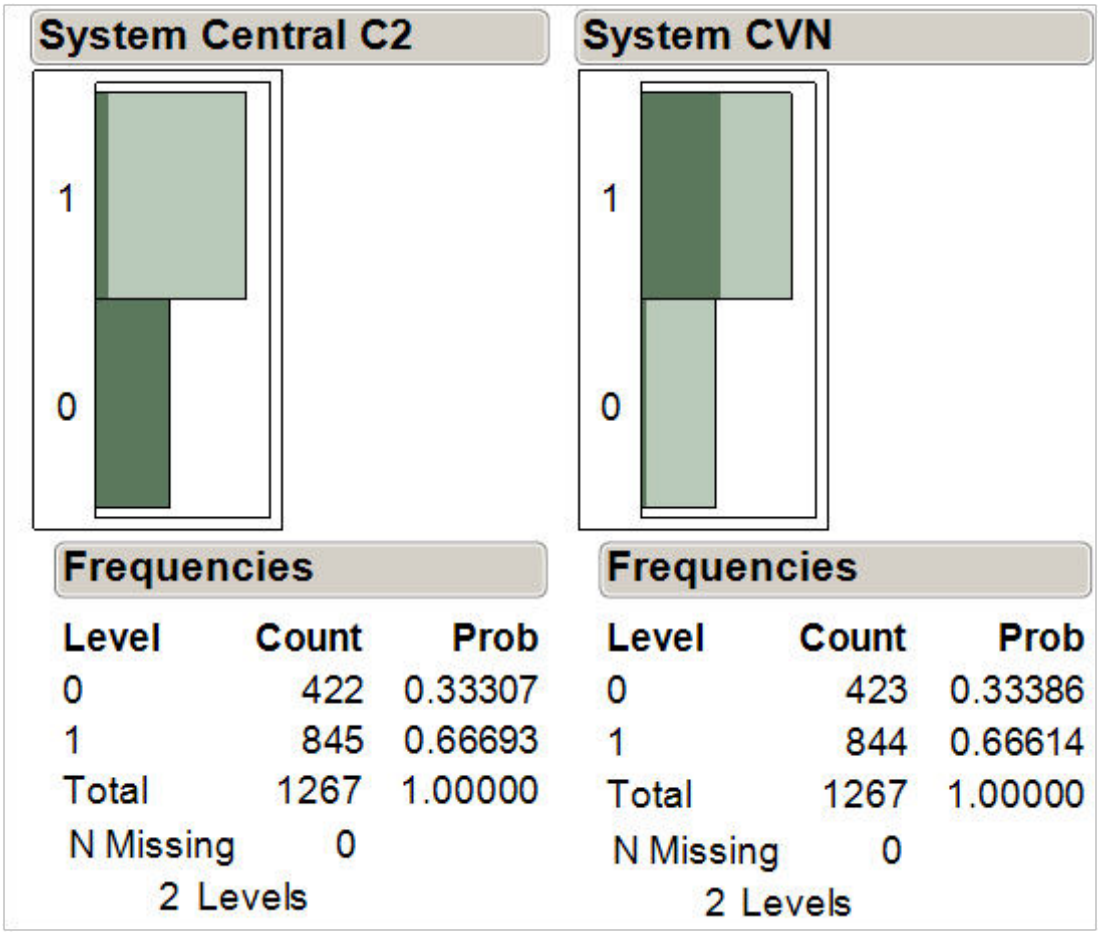


Figure 47: Distribution of Inclusion/Exclusion of Central C^2 and CVN Systems in Low Maintainability Portfolios

However, it still may be desirable to understand the source of the low maintainability score for those cases where the CVN system was not included in the portfolio. To do this, a double filter is applied. First, all of the lowest maintainability scoring portfolios are selected as the first filter stage. From that set, all of the portfolios that contain the CVN system are removed in the second filter stage. For the remaining subset of portfolios, the decision maker would look at the distributions of either including or excluding a system from the portfolio. It can be observed that these portfolios always include the Central C^2 system and the X-47B system. This is shown in Figure 48. The Central C^2 system is the only available alternative to the CVN so by default it must be included in viable portfolios. The other systems are not included in the subset 100% of the time. From this, we can conclude that the combination of Central C^2 and the X-47B lead to low maintainability scores. The results are expected as the X-47B is a new system that has more complex subsystems than poorer performing systems such as the F/A-18.

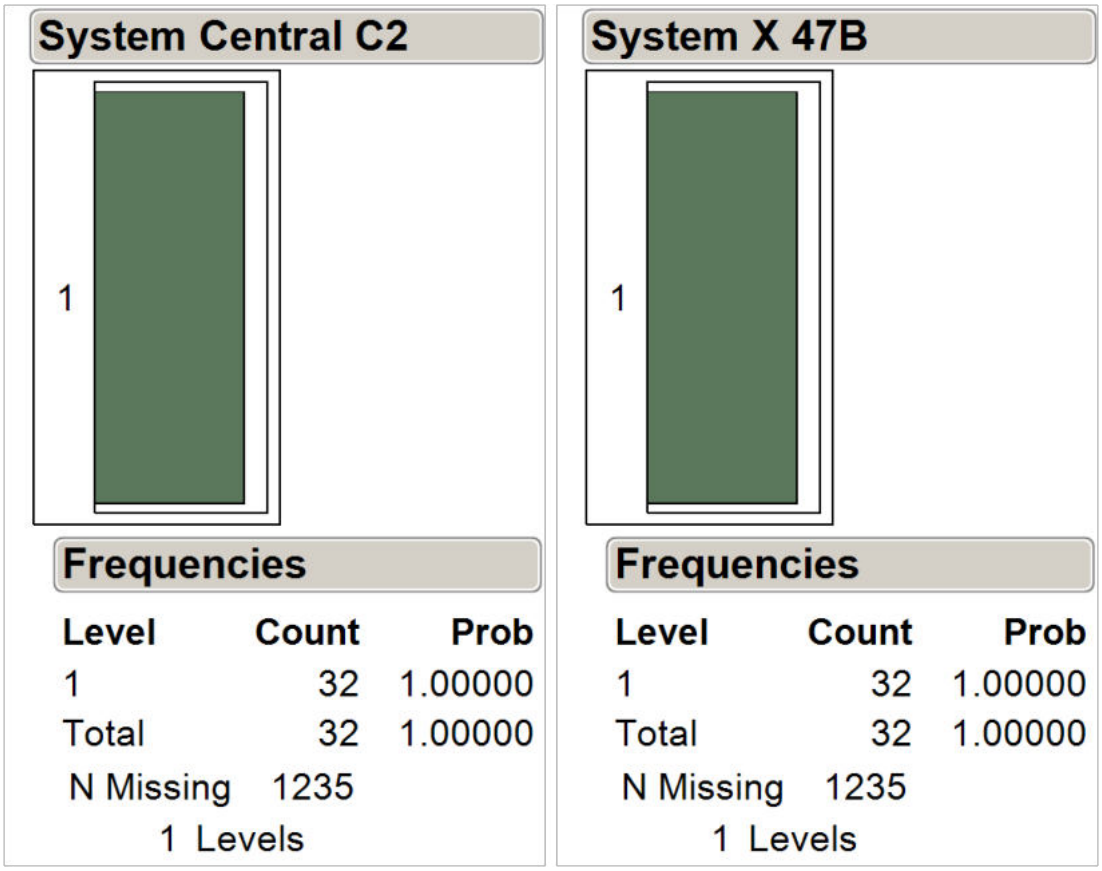


Figure 48: Distribution of Inclusion of Central C^2 and X-47B into Low Maintainability Portfolios without CVN

Going back to the scatter plot matrix of the metrics can show similar information for the portfolios with high maintainability. Similar analysis is possible for the highly maintainable portfolios. All of the portfolios with a value of 2.0 in maintainability are selected. They are highlighted in green in Figure 49.

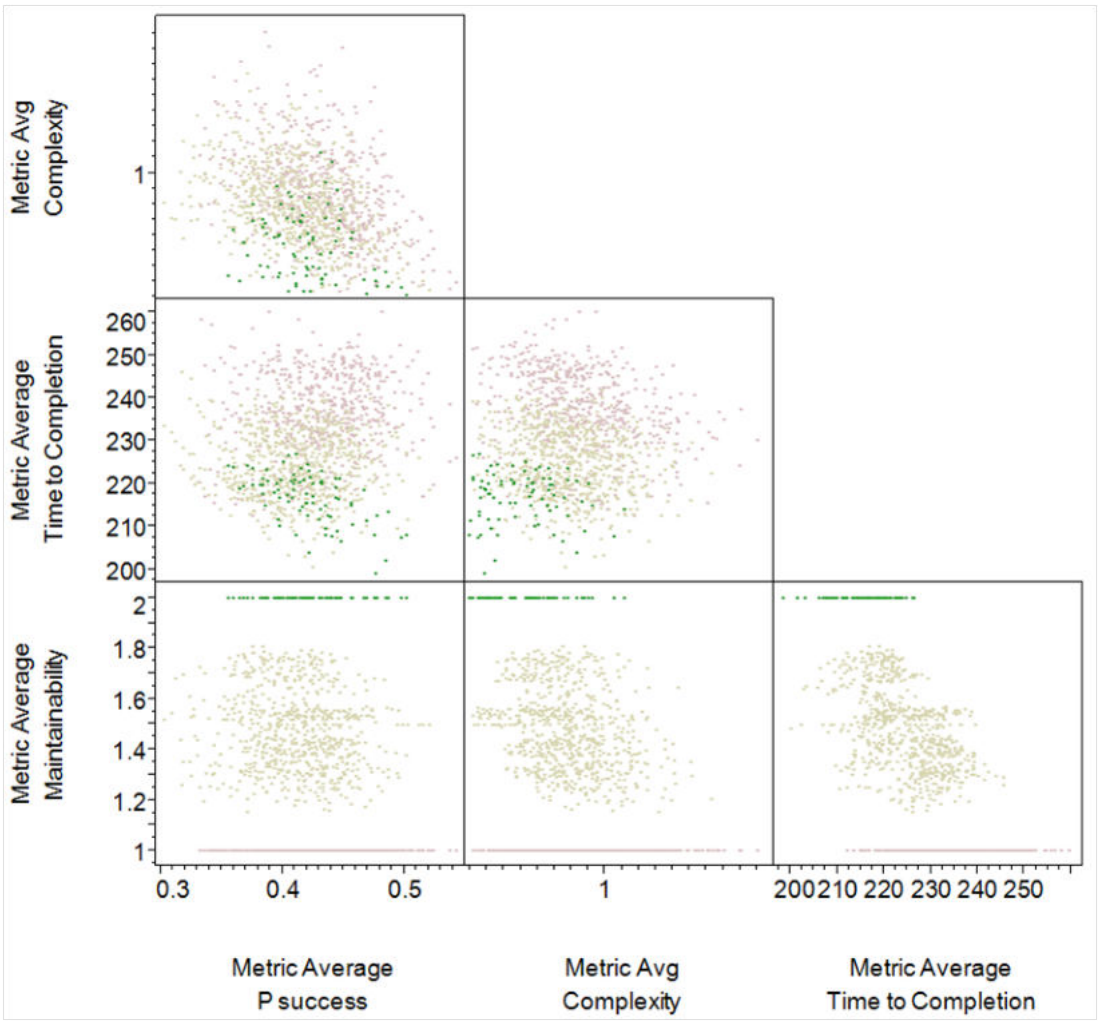


Figure 49: Scatter Plot Matrix of the SEAD Metrics with High Maintainability Highlighted

The system distributions are then examined with the high maintainability subset. Four system distributions are shown in Figure 50. The CVN and X-47B distributions are checked against the previous result from the low maintainability portfolios. As expected, the CVN and X-47B are never included in the portfolios with high maintainability. The Central C^2 system is included as expected. There are fewer portfolios that have high maintainability than have low maintainability. Unexpectedly, all high maintainability portfolios do not include the AH-64.

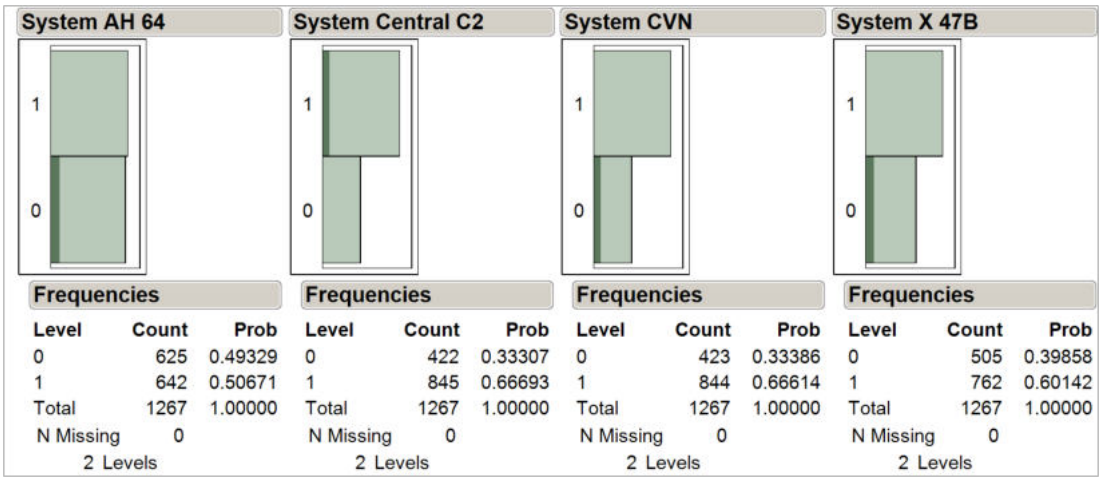


Figure 50: Distributions of Inclusion/Exclusion of AH-64, Central C^2, CVN, and X-47B for High Maintainability Cases

If maintainability is extremely important to the decision maker, they may decide to only consider those portfolios with the highest maintainability and filter out the rest of the portfolios. For the SEAD mission they will want to consider portfolios with the CVN system, but for the purposes of this example use of visualizations the assumption is that high maintainability is a requirement. The portfolios shown in Figure 51 are the portfolios that scored high in maintainability. JMP [82] has a feature where points that are close together are spread apart to make analysis possible. In this case, the average maintainability metric scores for all of the shown portfolios is 2.0. The JMP software has perturbed the points using the jitter feature.

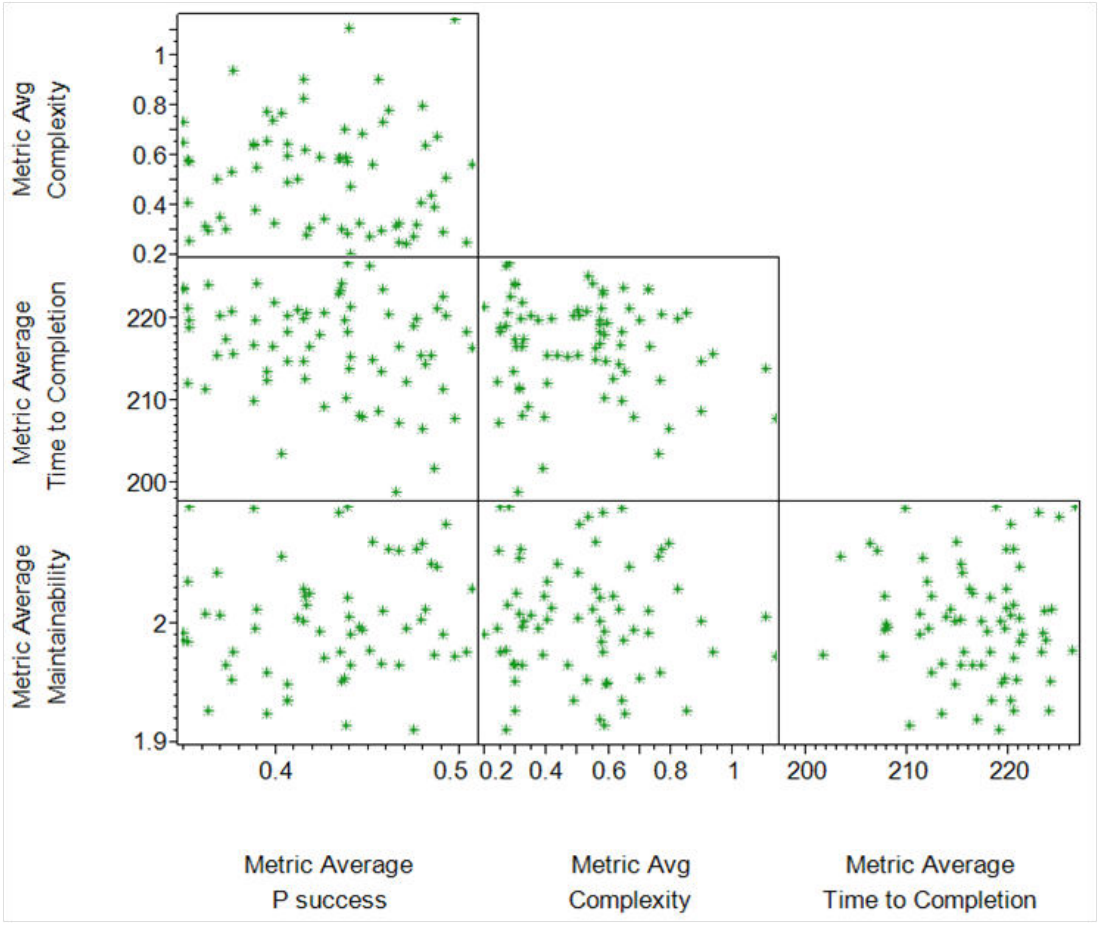


Figure 51: Scatter Plot Matrix of Highly Maintainable Portfolios

Figure 52 shows the distribution of portfolios in the complexity and time to completion metrics. Looking at the distribution of portfolios in the complexity metric shows that all of the highly maintainable portfolios have low complexity. In addition, the distribution is biased toward the lowest complexity portfolios. This is expected due to the correlation between maintainability and complexity in real world systems. The highly maintainable systems also have low time to completion scores with a less multi-modal distribution than all of the portfolios.

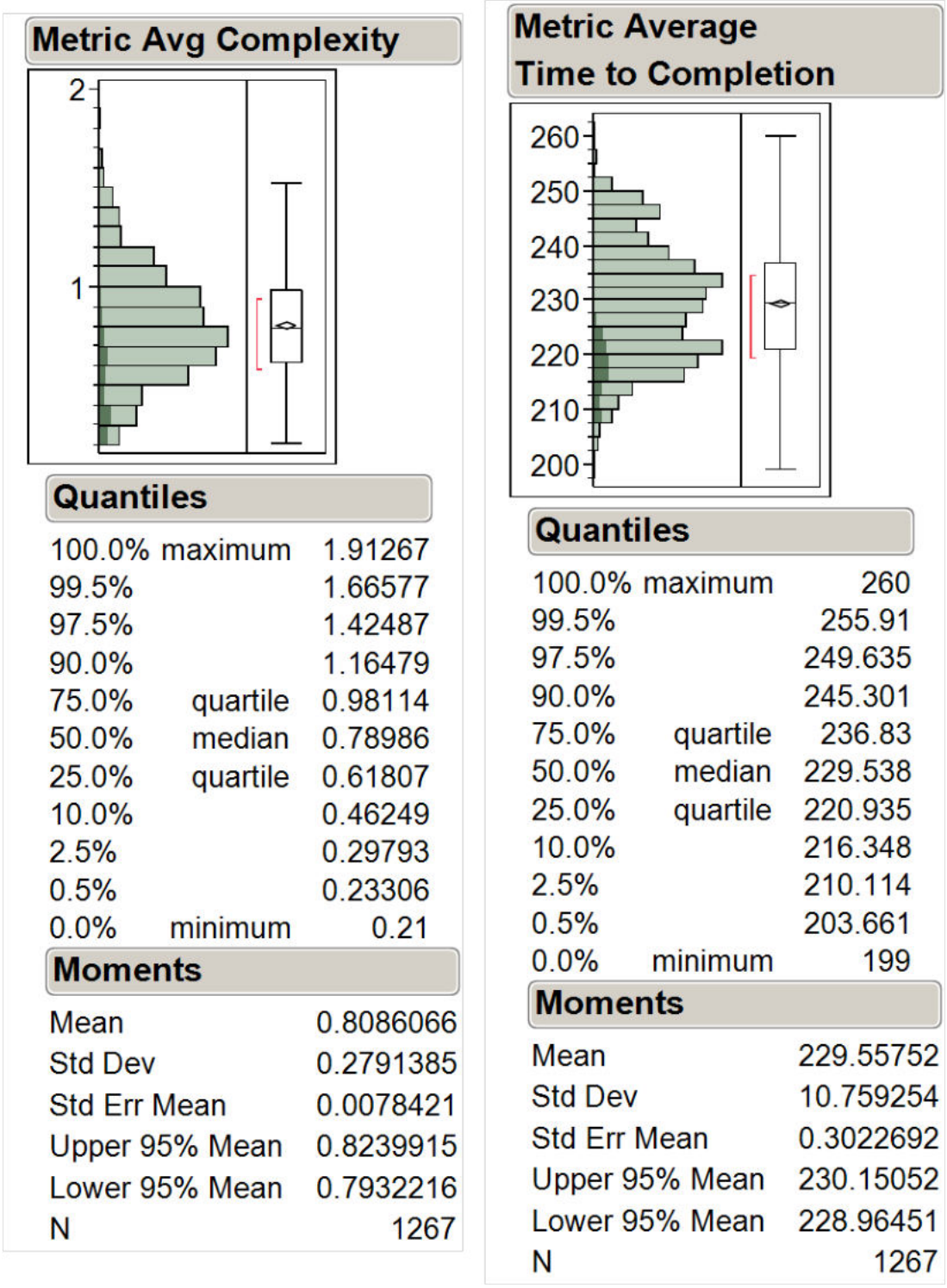


Figure 52: Distribution of Highly Maintainable Portfolios in Complexity and Time to Completion

Once the maintainability downselection has been made to include only highly maintainable systems, it may be necessary to apply further filtering based on mission requirements. Figure 53 shows the scatter plot matrix of the highly maintainable portfolios after filtering has been applied to time to completion. For example, perhaps the decision maker would choose to filter out those portfolios that have an average time to completion of greater than 215 minutes. 215 minutes is shown as a red line in the time to completion axis. The remaining portfolios are shown in Figure 53. The smaller subset of filtered portfolios is small enough to store the full data set for each possible system of systems architecture alternative from the selected portfolios.

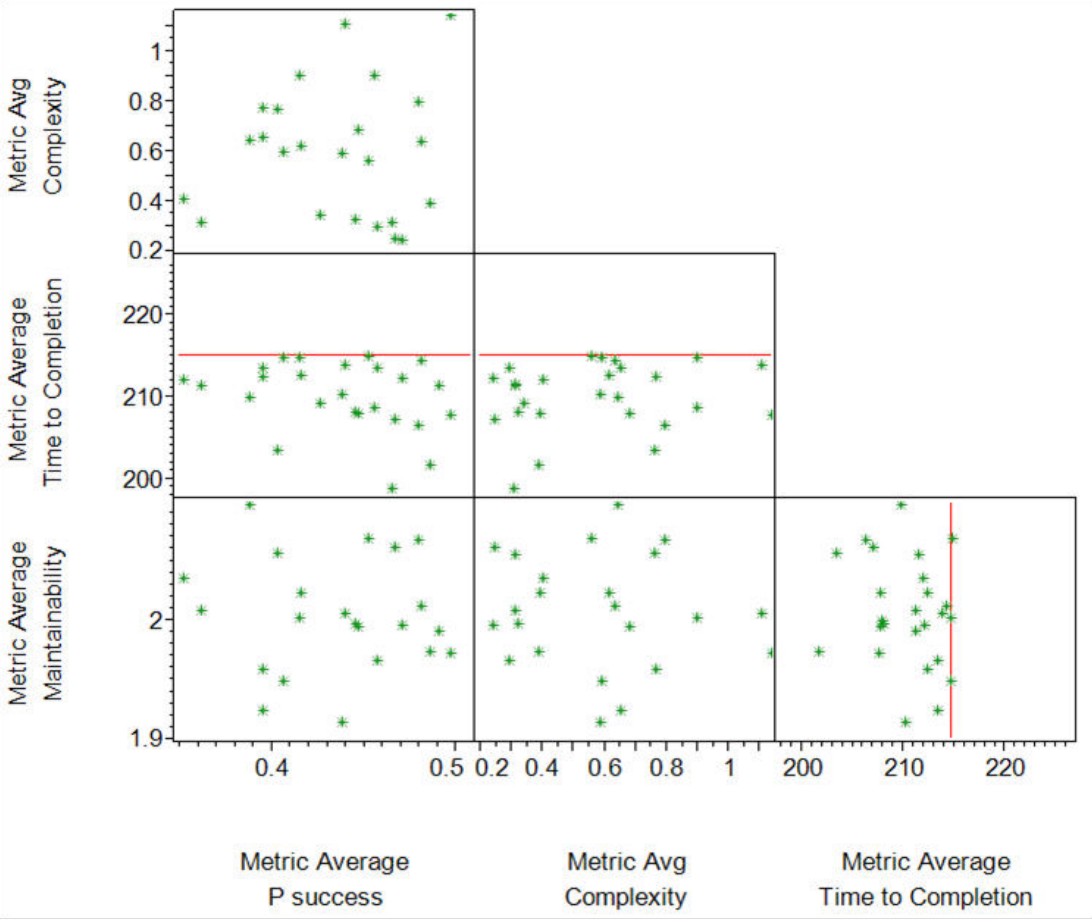


Figure 53: Filtered Scatter Plot Matrix of Highly Maintainable Portfolios

8.6.2 Portfolio Drilldown

The goal of this section of the analysis of the SEAD data is to explore a specific portfolio of systems. All of the different ways that the systems can be operationally allocated to tasks to complete the mission are explored. One goal of this analysis is to determine which system to task allocations are critical to achieving the desired mission performance. In addition, some tasks in the task hierarchy may not be as sensitive to which system is allocated to the task. The tasks that are less sensitive to system to task allocation allow for flexibility in the architecture. In this analysis, the four mission dependent metrics are used.

The portfolio selected for this analysis included four systems. The four included systems were:

- CVN
- Central C^2
- EA-6B
- E-2

Every possible system to task allocation was generated and executed for that set of four systems. There are 5152 different architecture alternatives that were combined to provide the original average portfolio scores. Figure 54 shows the distributions of the four mission dependent metrics. The probability of success metric score distribution is roughly normally distributed, with a mean of 49.6% and a standard deviation of 1.6%. The complexity metric score distribution is decidedly multi-modal. The majority of the architecture alternatives are in a cluster with scores between 700 and 1200. The complexity metric score distribution has a mean of 1052.57 and a standard deviation of 350. The time to completion metric score distribution is roughly normally distributed with a mean of 209.51 minutes and a standard deviation of 8.56 minutes.

The maintainability metric score is either 2.0 or 1.0 depending on the constituent systems. Due how JMP creates the distribution bins, the values of 2.0 are counted as between 2.0 and 3.0. Similarly, the scores of 1.0 is placed in a bin that is between 1.0 and 2.0.

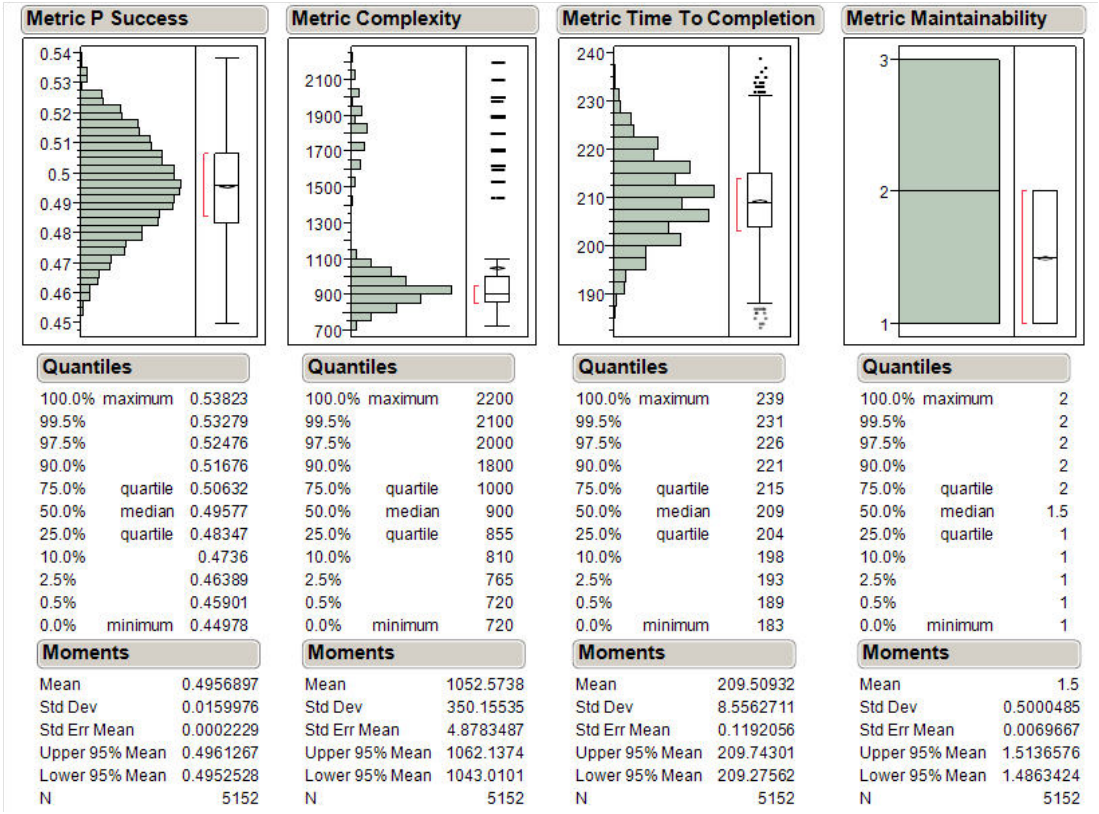


Figure 54: Distributions of Architecture Alternatives for One Portfolio

One notable observation that presents itself is the bimodal nature of the complexity distribution. To discover what is driving the high complexity architecture alternatives the architecture alternatives with high complexity are selected. They are highlighted in dark green and shown in Figure 55. The first thing to note is that these operational cases have a similar distribution in the other metrics to the full set of architecture alternatives for the portfolio. The probability of success metric score distribution has a similar mean and shape as the complete set. The time to completion metric score distribution has a slightly lower time to completion, but a similar

shape as compared to the complete set of architecture alternatives. For the highly complex architecture alternatives, the maintainability is distributed the same way as the complete set of architecture alternatives created from the portfolio. There is arguably a small decrease in the time to completion gained from the added complexity and this will be explored further.

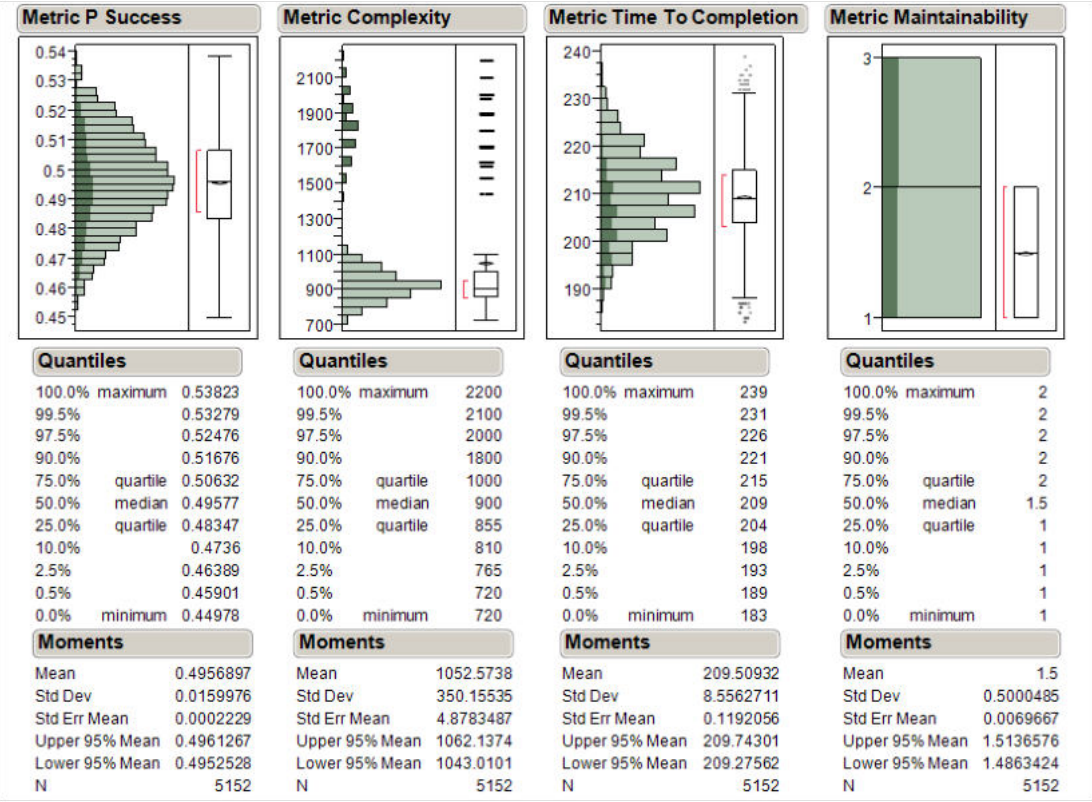


Figure 55: Distributions of High Complexity Architecture Alternatives for One Portfolio

One way to further explore the potential relationship between time to completion and complexity is to look at the scatter plot of time to completion vs complexity. This is shown in Figure 56 with the higher complexity cases colored red and the lower complexity cases colored green. The average time to completion of the red point cloud has a lower time to completion than the average of the green point cloud. However, the minimum time to completion in the red point cloud is not significantly less than the minimum time to completion for the green point cloud, so the added

operational performance is probably not worth the increase in complexity that may lead to integration challenges and added operational costs. Therefore, the higher complexity architecture alternatives can most likely be eliminated from consideration. However, before eliminating the higher complexity alternatives, it will be useful to understand what system to task pairings case the jump in complexity.

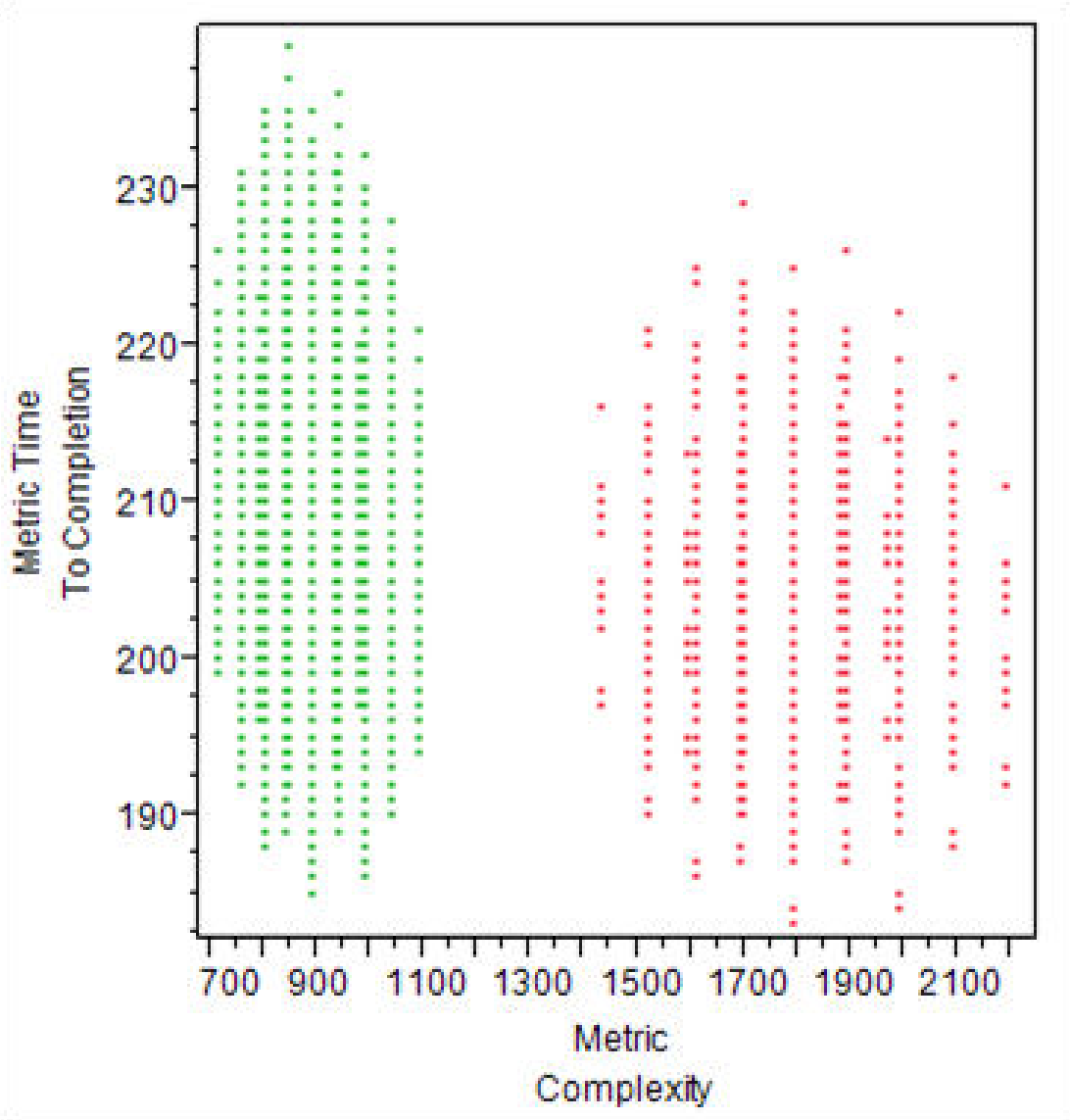


Figure 56: Time to Completion vs Complexity for All Alternatives in a Portfolio

In order to better determine what drives the complexity increase, distributions are created for each task based on how often a system is assigned to that task. This

is shown in Figure 57 for the Assess Engagement Capability and Update Target List tasks. By themselves, these distributions have little value to offer the decision maker. However, when these architecture alternatives with higher complexity are selected, we can see which system to task pairings are used. The dark green distribution in Figure 57 is from the higher complexity architecture alternatives while the light green distribution is all of the architecture alternatives in the portfolio. Based on this analysis, there are two system to task mappings that always occur in all of the highest complexity cases. When the CVN is used to Assess Engagement Capability and when the E-2 is used to Update the Target List. By including the CVN over Central C^2 we are bringing in a more complex system which should drive complexity higher. The inclusion of the E-2 adds communication complexity to either the CVN or Central C^2 as both will be used in other tasks in the portfolio.

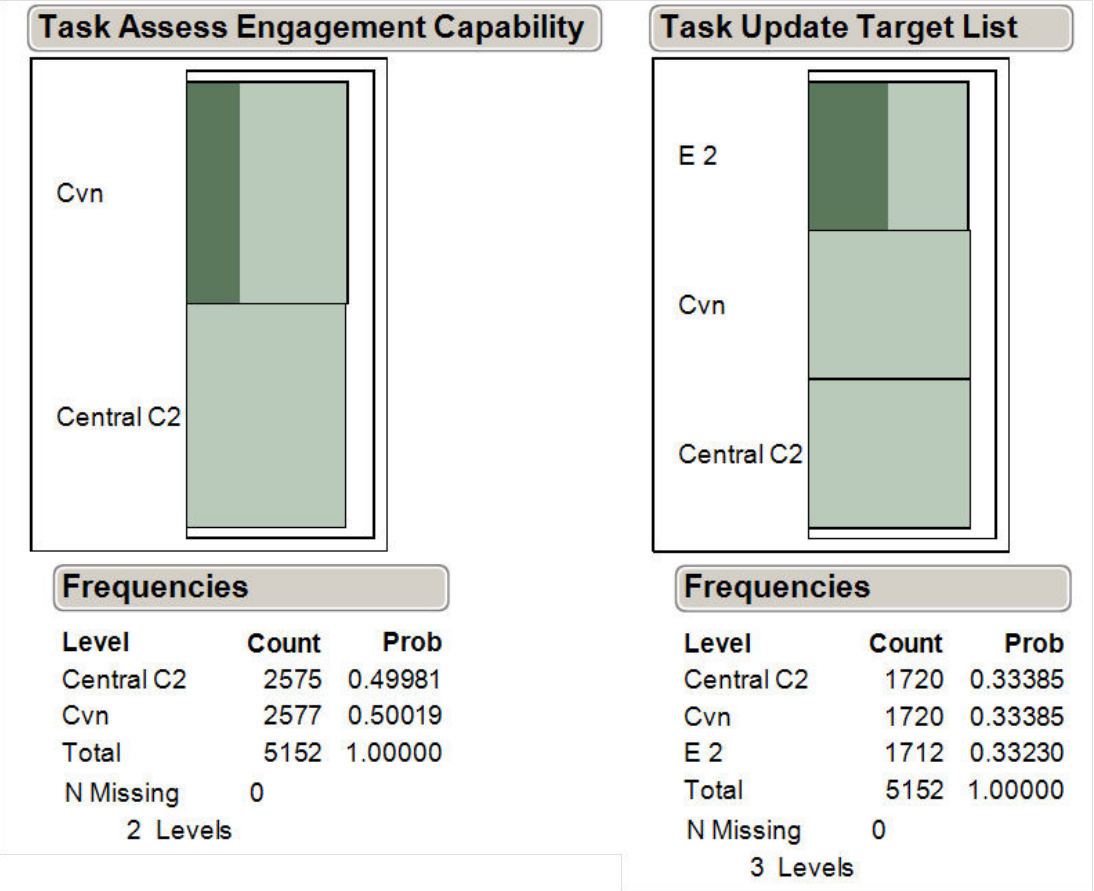


Figure 57: System to Task Distributions for Higher Complexity Alternatives

By looking at Figure 57 we have seen that there are architecture alternatives where the system to task mappings occur and the complexity is not in the high group. This begs the question of whether it is the combination of these two system to task mappings that drives complexity or not. To explore this idea further, all of the architecture alternatives where both of the previously identified system to task pairings occur are selected. The combination of Assess Engagement Capability is accomplished by the CVN and Update Target List is accomplished by E-2 does in fact correlate exactly to those cases in the higher complexity grouping. This suggests that these two system to task mappings are together the cause of the higher complexity. Figure 58 shows that when cases with the CVN accomplishing Assess Engagement Capability and the E-2 accomplishing Update Target List that only the

higher complexity cases are highlighted.

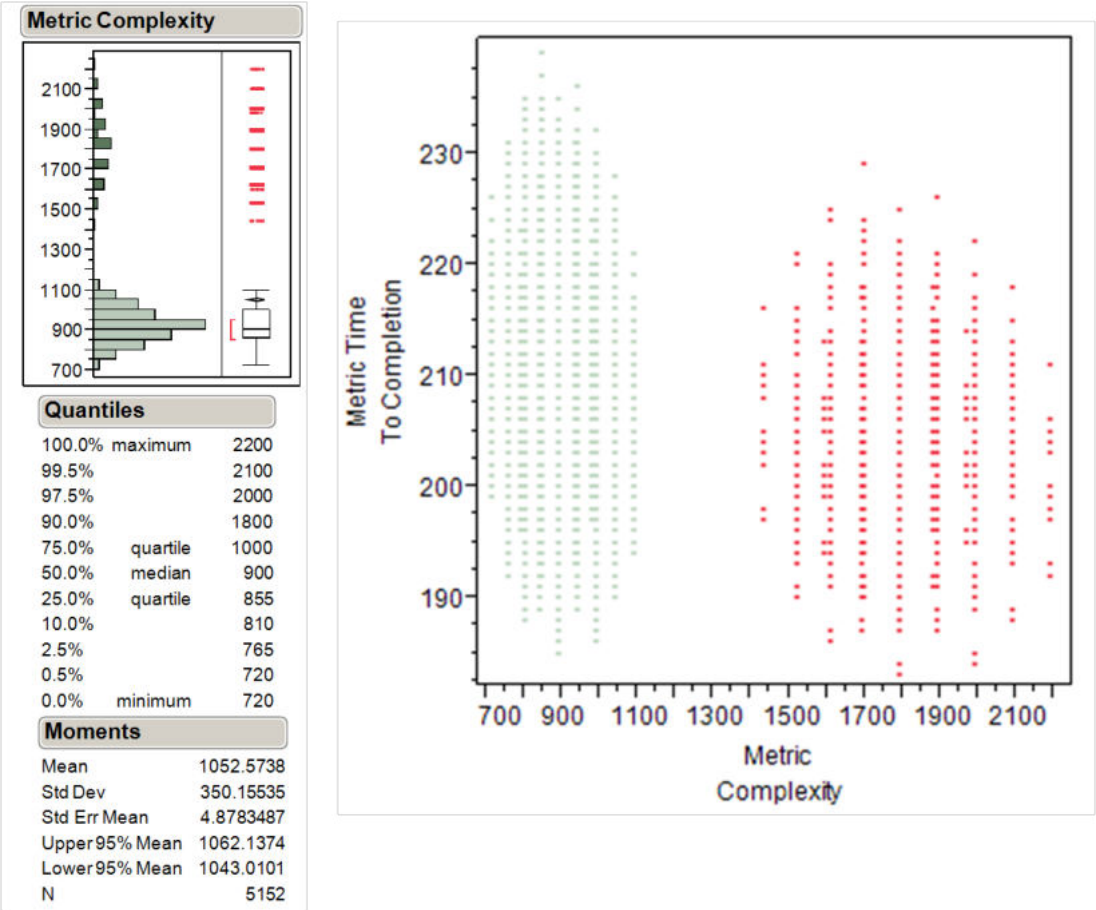


Figure 58: Scatter plot and Distribution of Selected System to Task Pairings

However, since it was noted that the increase in complexity does not provide significant potential for operational grain, it is further likely that a decision-maker would exclude those cases with higher complexity. After this down selection occurs, the remaining cases are explored using a scatter plot matrix. The green points from Figure 58 are retained and shown in a scatter plot matrix in Figure 59. The initial observation from this matrix is that the maintainability metric scores are also bimodal like the complexity scores. This is better shown with Figure 60.

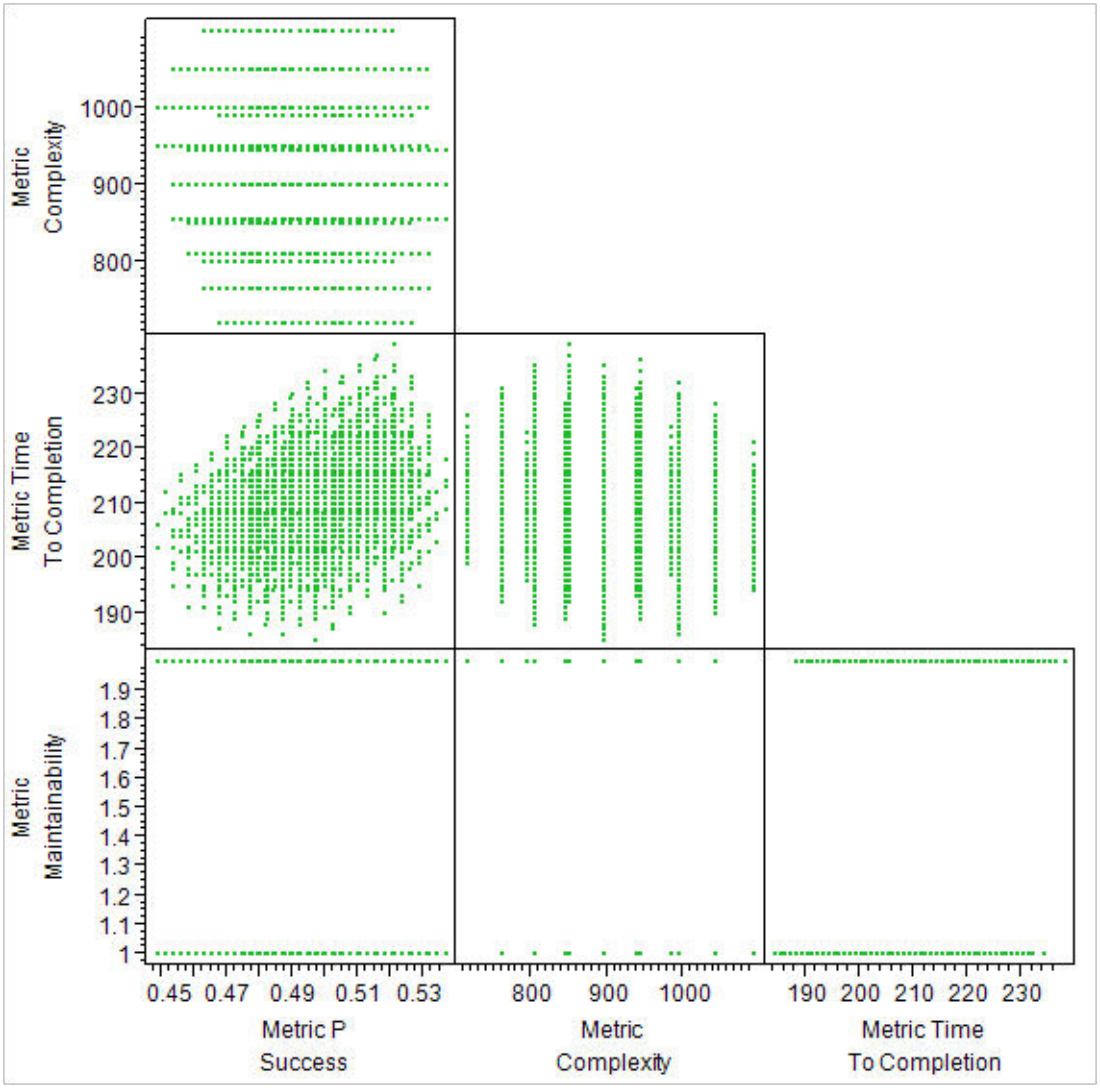


Figure 59: Scatter Plot Matrix of Low Complexity Alternatives

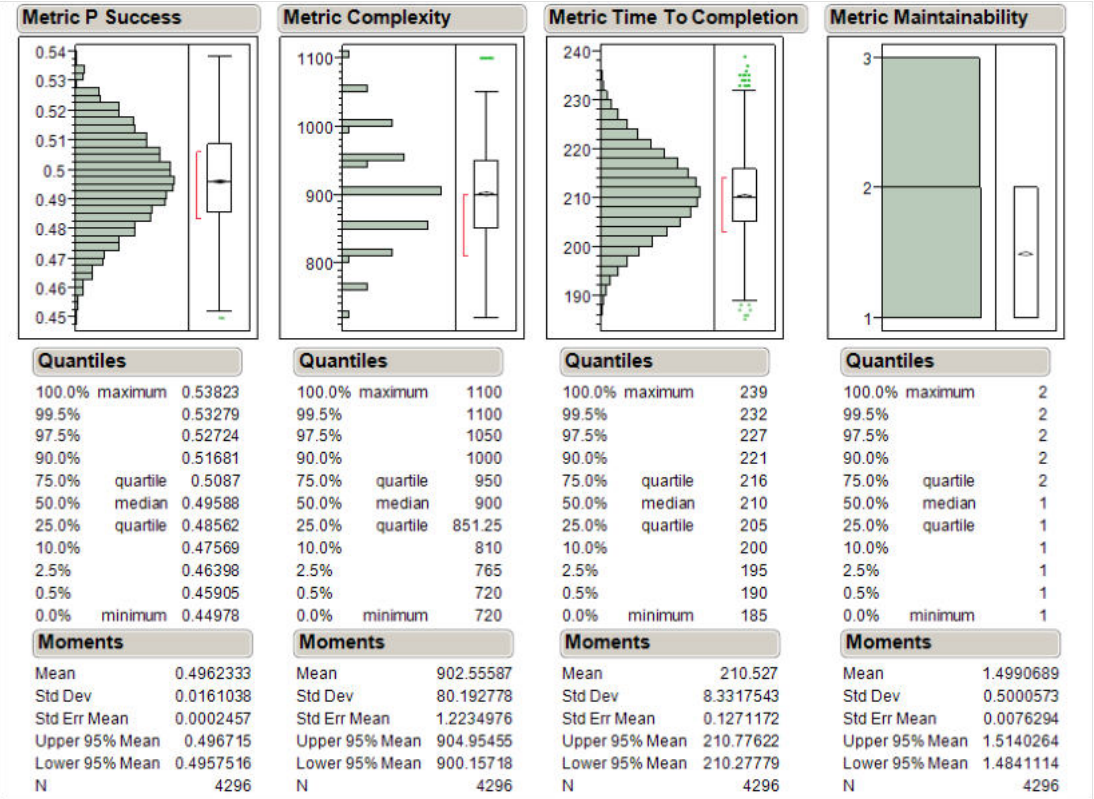


Figure 60: Distributions of Low Complexity Alternatives

In order to further explore the bimodal maintainability, the cases with lower maintainability are selected. This is shown in Figure 61. Performing a similar analysis to the complexity, it is immediately clear that the lower maintainability does not lead to an operational gain. The decision maker will likely be interested in removing the lower maintainability cases.

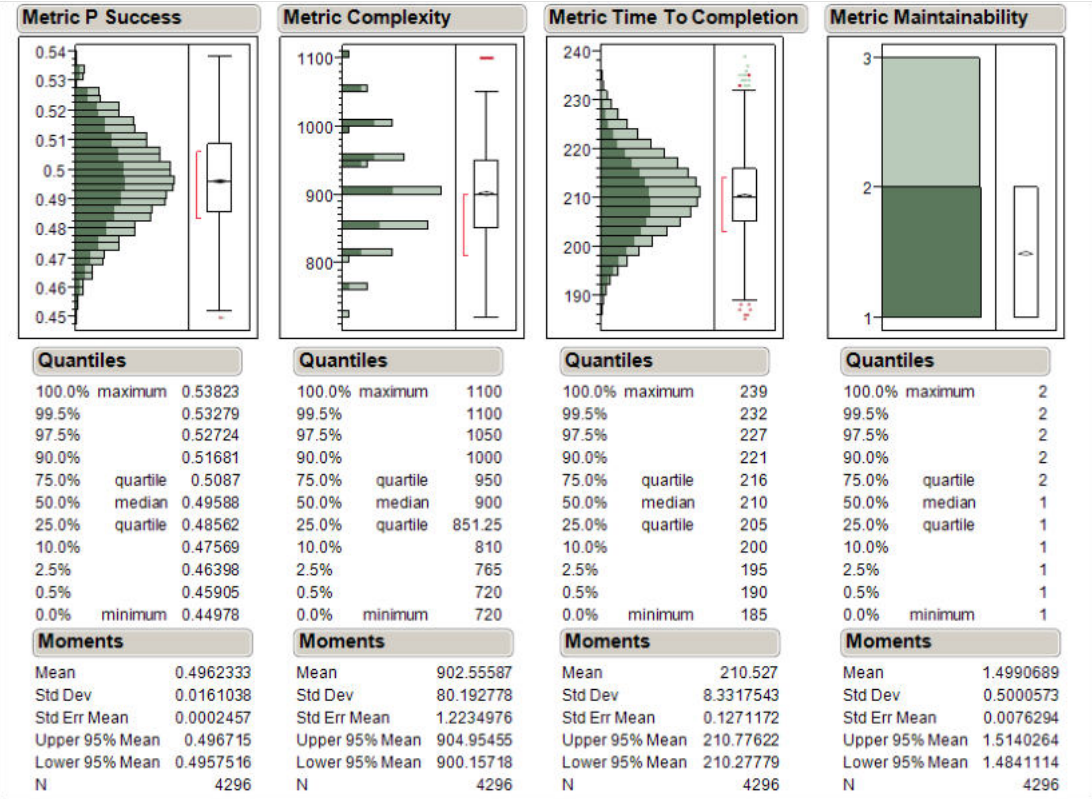


Figure 61: Distribution of Lower Complexity Cases with Low Maintainability Selected

Keeping only those cases with the higher maintainability scores, the resulting alternatives are shown in the scatter plot matrix in Figure 62. The Time to Completion vs Probability of Success scatter plot shows that there is still freedom in those dimensions to find useful alternatives.

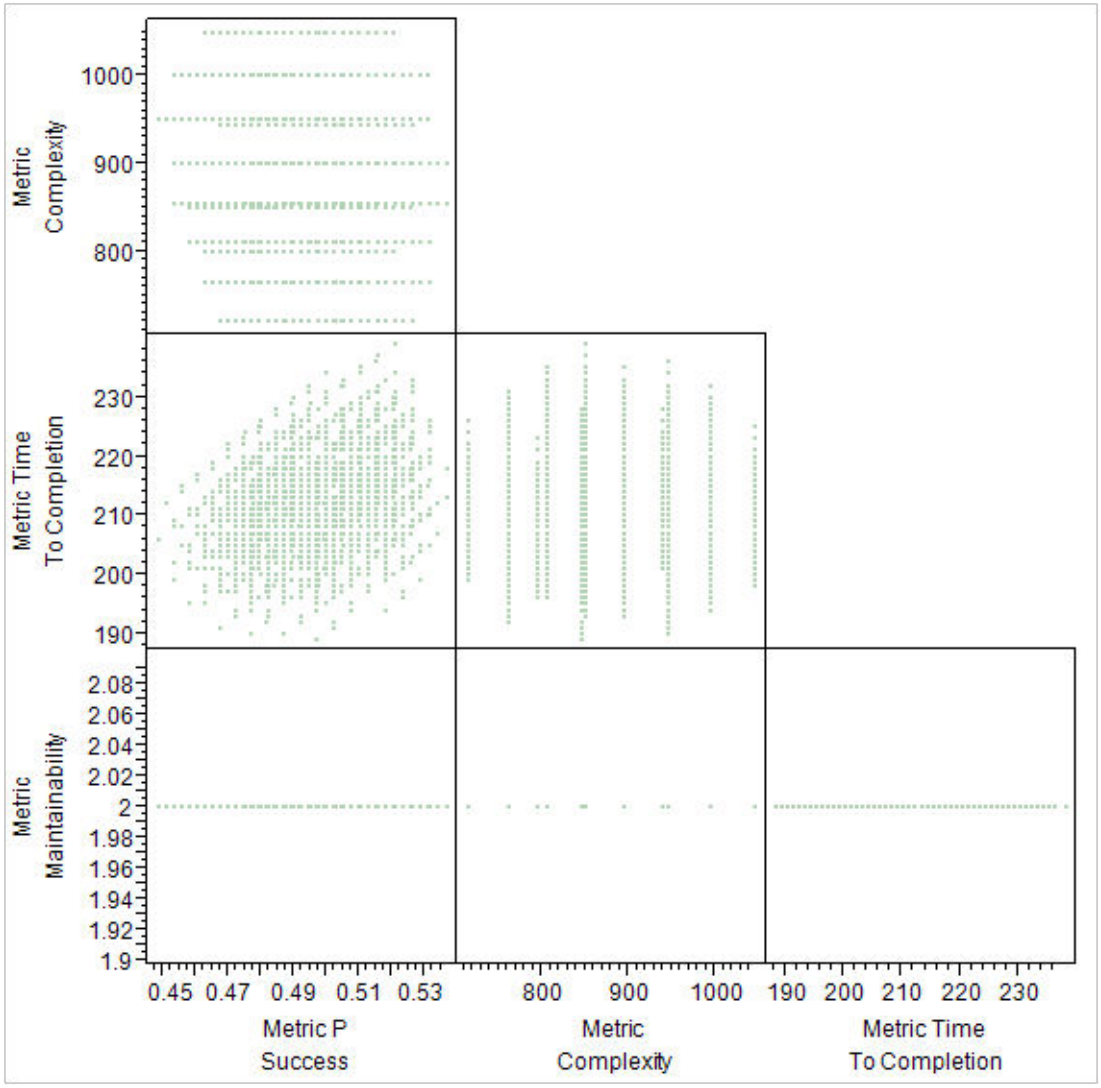


Figure 62: Scatter Plot Matrix with Low Complexity and High Maintainability

It is interesting to note that there is a trade between the Probability of Success and the time to completion as indicated by the red line in Figure 63. Since the decision maker is interested in those cases that have the highest probability of success with the shortest time to completion, the region designated by the orange oval overlay is desirable. The points that lie along the red Pareto Frontier are of most interest. The points outside of the desired region can be removed.

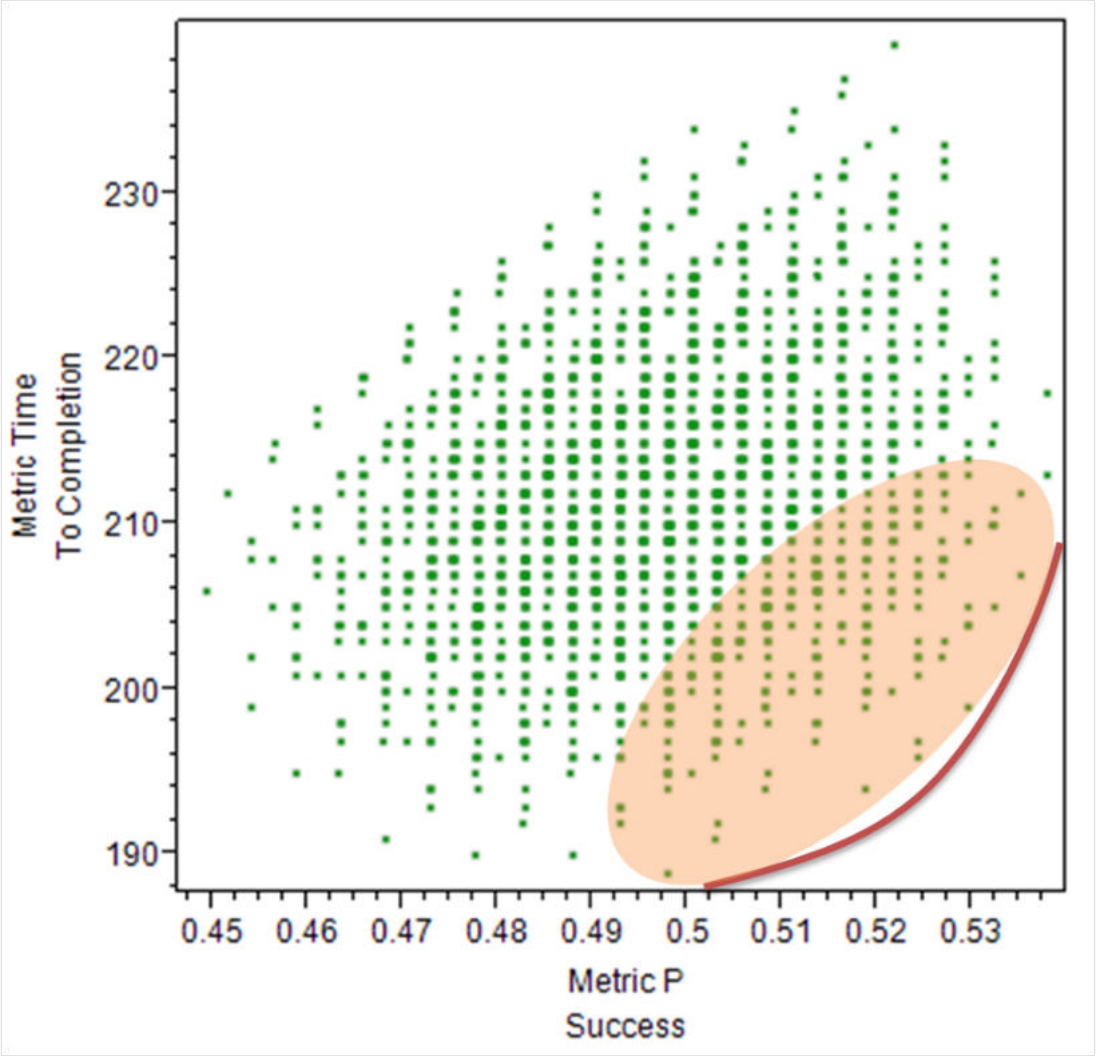


Figure 63: Scatter Plot of Probability of Success vs Time to Completion

The remaining architecture alternatives are shown in Figure 64. These architecture alternatives have low complexity, high maintainability, high probability of success and low time to completion.

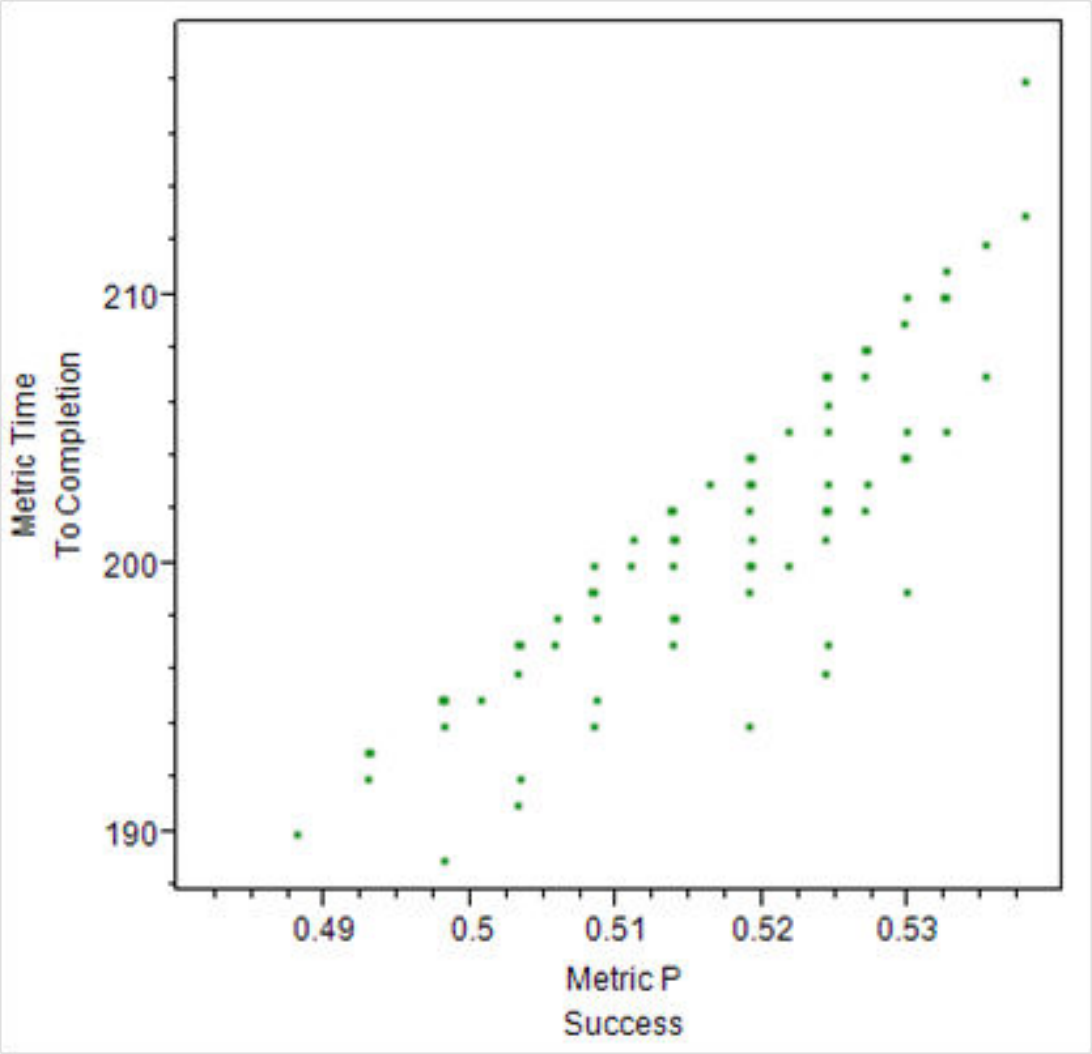


Figure 64: Pareto Architecture Alternatives on a Scatter Plot of Probability of Success vs Time to Completion

For this set of remaining cases, it is of interest to explore the system to task pairings. There are 77 remaining architecture alternatives on the Pareto Frontier. Sixteen of the most relevant system to task distributions are shown in Figures 65, 66, 67, and 68. The most important system to task pairings will be critical for operational prioritization when using the selected portfolio. The only engagement system in the portfolio was the EA-6B. It was used for the Engage to Destroy and Engage to Disrupt tasks. The CVN should be used for the Assess Engagement Capability and Determine

Sensor Availability tasks over Central C^2. The Central C^2 system should be used for the Assign Weapon And Platform task, the Pass Warning And Location Data task, and the Task Sensor task. The E-2 is most often used for the Fuse Sensor Data and Manage Target Movement Data tasks. Conclusions can not be drawn about the Remove From Target task as both the CVN and Central C^2 systems are used with approximately the same frequency. The same effect occurs with the Update Target List task.

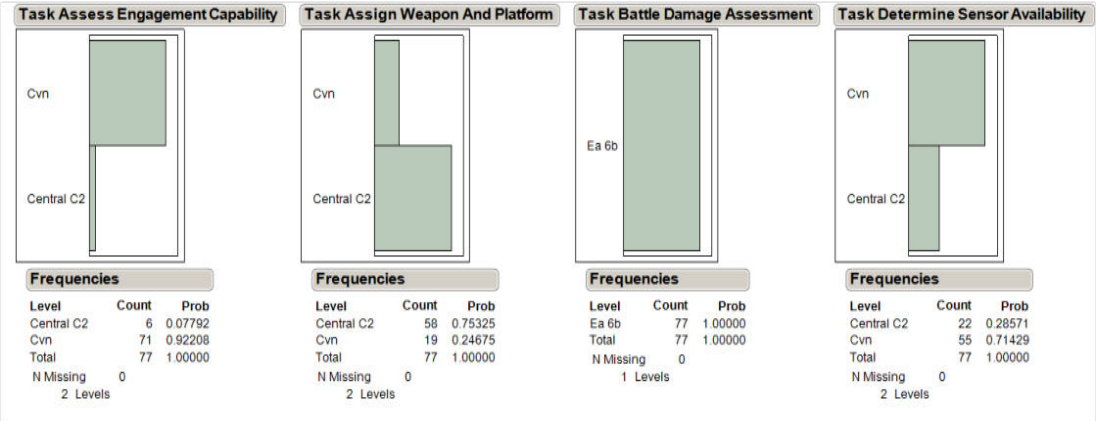


Figure 65: Distribution of System to Task Allocation for the Selected Portfolio 1

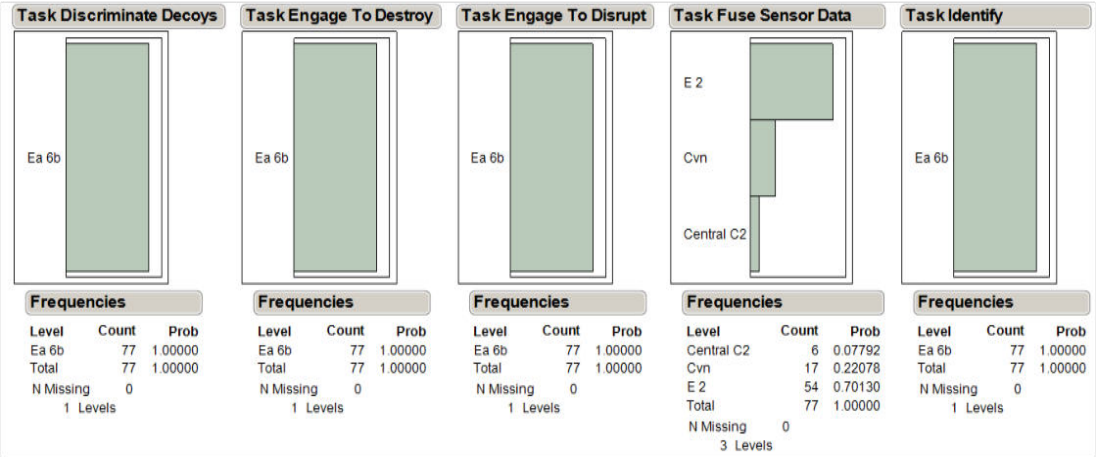


Figure 66: Distribution of System to Task Allocation for the Selected Portfolio 2

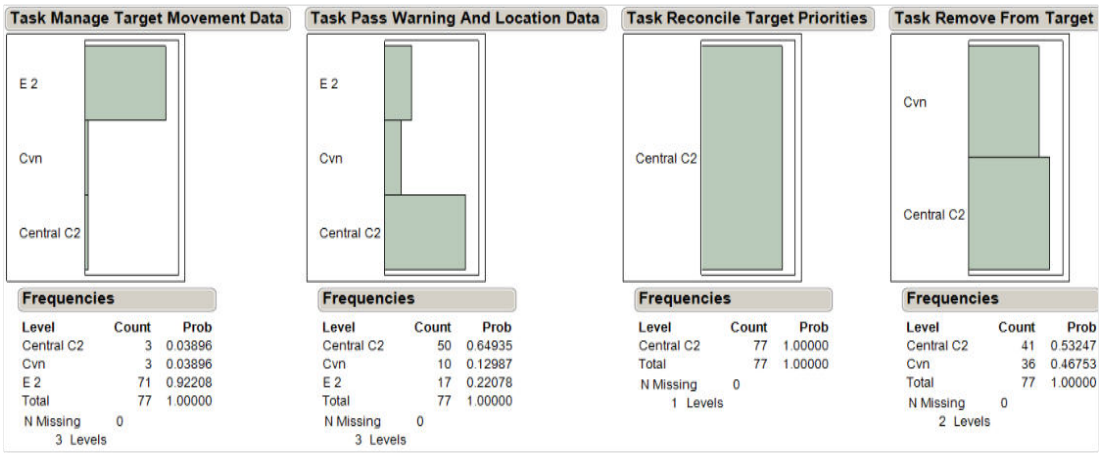


Figure 67: Distribution of System to Task Allocation for the Selected Portfolio 3

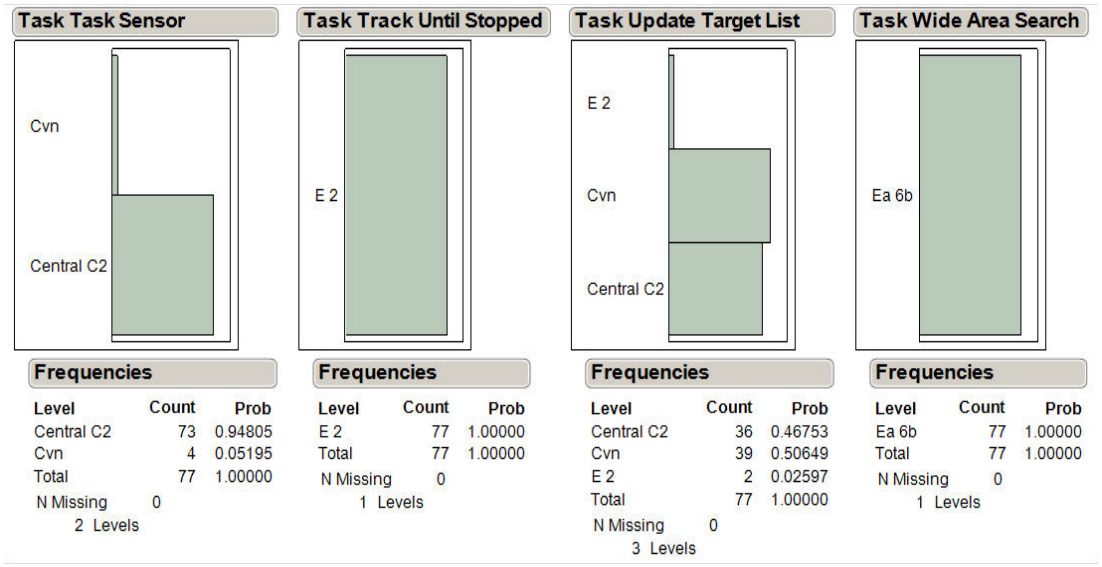


Figure 68: Distribution of System to Task Allocation for the Selected Portfolio 4

8.7 Experiments

The following series of experiments are designed to test their corresponding hypotheses. This section is a summary of the results as they pertain to the hypotheses from the canonical example and the proof of concept. A high level summary is shown in Figure 69.

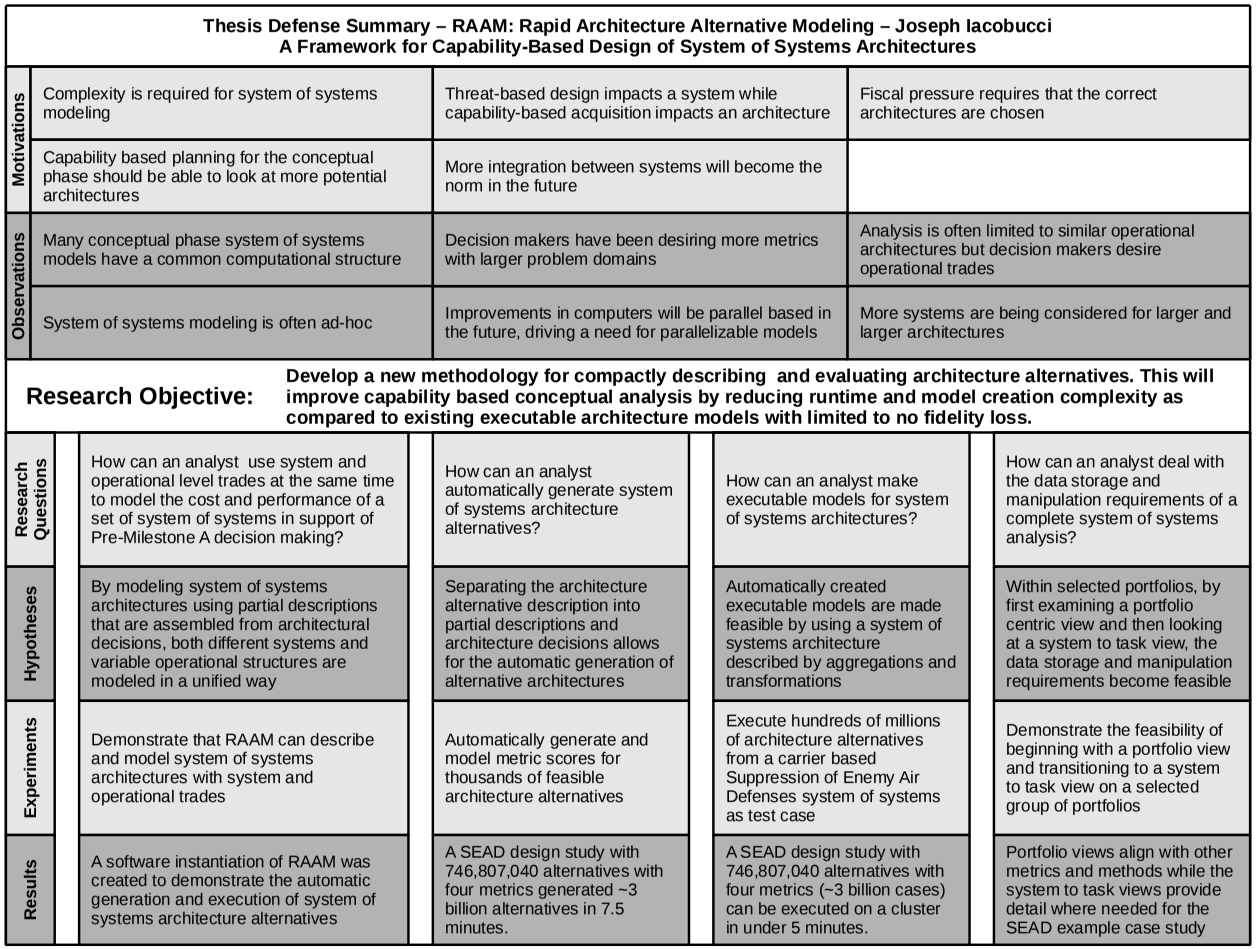


Figure 69: Summary of Experiments and Results

8.7.1 Experiment 1

The first experiment is to utilize a framework and methodology using the Rapid Architecture Alternative Modeling to analyze alternatives for a system of systems architecture portfolio. The chosen scenario is a military mission similar to the Suppression of Enemy Air Defenses (SEAD). The scenario currently has a strong Naval component.

The framework and methodology are documented in Chapter 6 (pg. 80). The Rapid Architecture Alternative Modeling implementation is documented Chapter 7 (pg. 91).

The Suppression of Enemy Air Defenses (SEAD) capability looked at four mission dependent and two mission independent metrics. These metrics of interest include metrics that measure performance, schedule, and cost. The four mission dependent metrics are Probability of Success, Maintainability, Complexity, and Time to Completion. The two mission independent metrics are Risk and Cost. The declarative description of the system of systems architecture portfolio analysis of alternatives provided by Rapid Architecture Alternative Modeling was tested in analyst time, computer time and cognitive load. All metrics except computer time are qualitative measures. Computer time is a quantitative measure.

The partial descriptions were used to describe both the canonical model and the Suppression of Enemy Air Defenses. The canonical example contained both operational and system level trades. By modifying the leaf task hierarchy nodes with a system selector function and converting the metric scores to a function, the unification of the operational and system level trades is achieved.

8.7.2 Experiment 2

The second experiment is designed to test Hypothesis 2. The methodology must be able to generate all feasible architecture alternatives from a declarative description.

Since thousands of candidate architecture must be generated, each alternative will not be able to be verified. A count of feasible alternatives is possible from information about the interoperability and possible system to task mappings. If the automatic generation of alternative architectures is successful, Hypothesis 2 will be shown to be true.

From the Suppression of Enemy Air Defenses design study, 746,807,040 alternatives with four mission dependent metrics generated $\tilde{3}$ billion cases in 7.5 minutes on a desktop machine. All of the expected architecture alternatives were created with minimal overhead. A streaming algorithm has been developed to generate the next alternative architecture on demand as is required. An efficient implementation of the generation algorithm allows the exploration of the entire design space.

8.7.3 Experiment 3

The third experiment builds off of experiment two. Using the generated alternatives from experiment two, a translator converts the declarative RAAM description to executable code for each alternative. The engineering models described using the RAAM method for performance, schedule, and cost are converted into computer programs automatically. If it is possible to automatically convert the descriptions in RAAM into executable models then Hypothesis 3 will be shown to be true.

A SEAD design study with 746,807,040 alternatives with four metrics ($\tilde{3}$ billion cases) can be executed on a cluster in under 5 minutes. The system of systems capability study for the SEAD case has been executed on a grid computing infrastructure in four minutes and forty two seconds. Previously it was not often possible to analyze such a large number of alternatives. Architecture alternative studies typically take six months to a year. Not all of the time is computational time, but analysis of the data is possible within two weeks with RAAM.

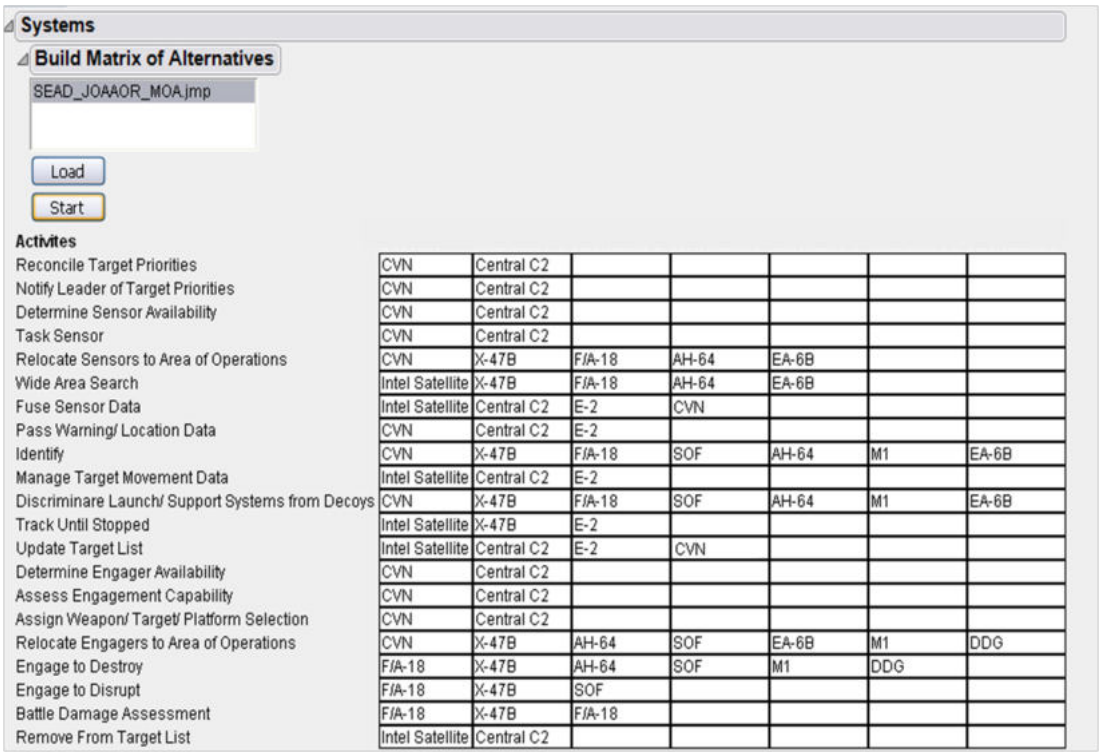
Since the modeling framework is easier to comprehend, the cognitive load is decreased. By using the abstraction of aggregation and transformation functions, it is possible to have a common description method for many types of relevant models. The input data is the same type across the different metrics, both of the scores for a system and for the aggregation and transformation functions. Decisions in the possible architecture space are able to be turned into a unique, executable model.

8.7.4 Experiment 4

Experiment four is the result of running into data storage and manipulation requirements from running the SEAD mission cases. It has been demonstrated that it is feasible to start with a portfolio view and 'zoom' into the system to task view as needed. The portfolio view used an average score for the portfolio when it was compared against the other portfolios.

The data requirements for storing the entire system of systems architecture alternative study can become prohibitive. Datasets can measure into the tens of gigabytes. Binary formats would be smaller but may require the use of specialized tools. Typical office and academic software packages such as Microsoft Excel and SAS's JMP can not load datasets of that magnitude. The portfolio view runs all of the architecture alternatives, but only stores the average values for the different metrics. In the SEAD example, the number of data rows changes from 746,807,040 to 1266 by grouping architecture alternatives by portfolios of systems.

Different systems can accomplish the tasks in the SEAD capability. Figure 70 shows the different possible systems for each task in the Suppression of Enemy Air Defenses capability. The linkages between system and task are not dependent on each other. Since this architecture design study does not have any operational trades, the system to task mapping can be represented in a two dimensional table.



Activites							
Reconcile Target Priorities	CVN	Central C2					
Notify Leader of Target Priorities	CVN	Central C2					
Determine Sensor Availability	CVN	Central C2					
Task Sensor	CVN	Central C2					
Relocate Sensors to Area of Operations	CVN	X-47B	F/A-18	AH-64	EA-6B		
Wide Area Search	Intel Satellite	X-47B	F/A-18	AH-64	EA-6B		
Fuse Sensor Data	Intel Satellite	Central C2	E-2	CVN			
Pass Warning/ Location Data	CVN	Central C2	E-2				
Identify	CVN	X-47B	F/A-18	SOF	AH-64	M1	EA-6B
Manage Target Movement Data	Intel Satellite	Central C2	E-2				
Discriminare Launch/ Support Systems from Decoys	CVN	X-47B	F/A-18	SOF	AH-64	M1	EA-6B
Track Until Stopped	Intel Satellite	X-47B	E-2				
Update Target List	Intel Satellite	Central C2	E-2	CVN			
Determine Engager Availability	CVN	Central C2					
Assess Engagement Capability	CVN	Central C2					
Assign Weapon/ Target/ Platform Selection	CVN	Central C2					
Relocate Engagers to Area of Operations	CVN	X-47B	AH-64	SOF	EA-6B	M1	DDG
Engage to Destroy	F/A-18	X-47B	AH-64	SOF	M1	DDG	
Engage to Disrupt	F/A-18	X-47B	SOF				
Battle Damage Assessment	F/A-18	X-47B	F/A-18				
Remove From Target List	Intel Satellite	Central C2					

Figure 70: Morphological Matrix for the SEAD Capability

First, the data from the portfolio viewpoint is plotted in a scatter plot matrix. The scatter plot matrix shows the performance of the different architecture alternatives for all of the mission dependent metrics. This is shown in Figure 71. There is a scatter plot for each possible pair in a scatter plot matrix. Patterns can be extracted from the scatter plot matrix for further use. For example, maintainability has three distinct concentrations of data points. There is a bottom row of points, a middle cluster of points, and a top row of points. By selecting the interesting points, the system that causes the trifurcation effect can be found.

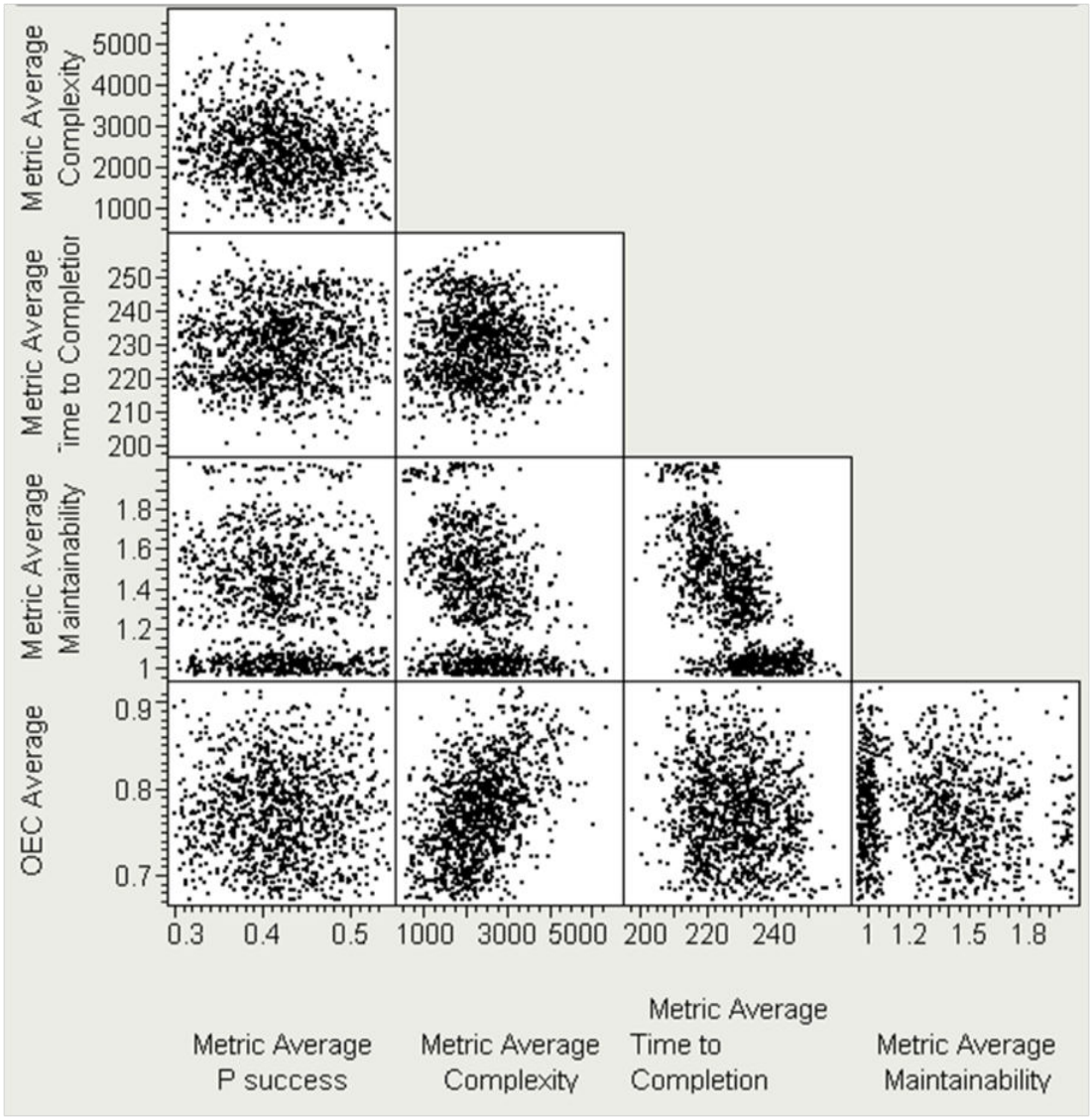


Figure 71: SEAD Scatter Plot Maxtrix

The scatter plot matrix can be used to find a subset of portfolios to carry forward. In Figure 72, there are sixteen portfolios chosen out of the original 1266. The decision maker filtered the data set by choosing the maximum values for Probability of Success, the minimum values for Complexity, the minimum values for Time to Completion, and a range of values for Maintainability.

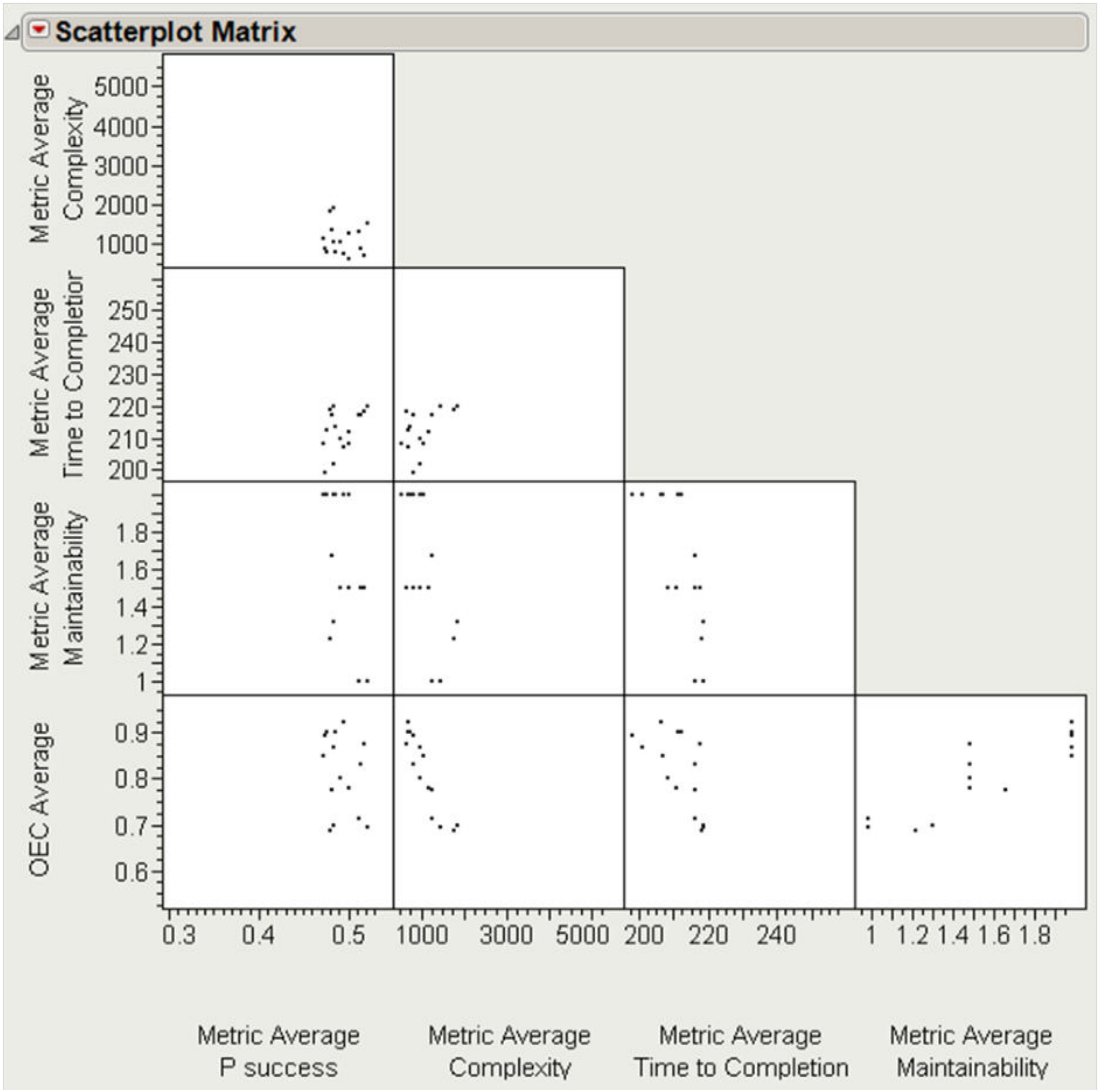


Figure 72: Portfolio Down Selection for SEAD

In addition, the performance of portfolios with or without a system can be plotted. Figure 73 shows the impact of using a DDG, the Carrier (CVN), and disallowing the use of an intelligence satellite.

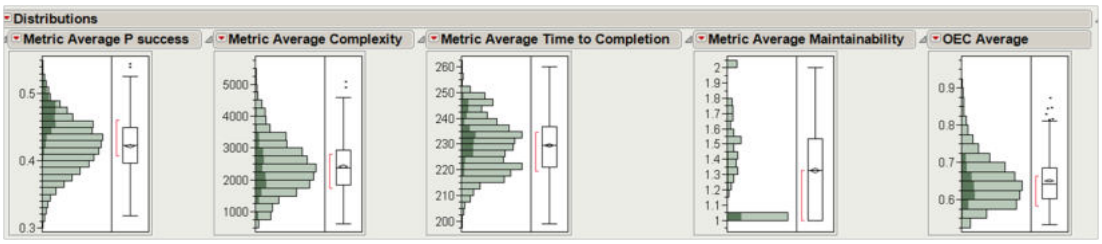


Figure 73: Example impacts of system selections

The selected sixteen portfolios were analyzed using RAAM to produce the full dataset for each of those portfolios. The full factorial result from a portfolio can also be very large, into the millions or tens of millions of cases. Specialized tools may be required at that point to fully analyze the data. The data set was loaded back into the same visualization environment to allow for further filtering.

8.8 Experiment Summary

The experiments were tested against a RAAM implementation that implemented the algorithms described in this manuscript. The RAAM algorithms required certain characteristics to be able to meet the research objective. This is summarized in Table 20.

Table 20: RAAM Algorithm Characteristics

Desired Characteristics	RAAM Algorithm has Characteristic
System Level Trades	yes
Operational Level Trades	yes
Reduced Runtime	yes
Linear Scaling	yes
Full Decision Space Exploration	yes
Only Runs Feasible Cases	yes
Automatic Generation of Alternatives	yes
Automatic Evaluation of Alternatives	yes
Portfolio Computation	yes
Streaming Metric Score Updating	yes

In addition, there were two test problems that demonstrated the ability to meet the research objective. The test problems and their characteristics are summarized

in Table 21.

Table 21: Test Problem Characteristics

Desired Characteristics	Canonical Example	SEAD Example
System Choices	yes	yes
Operational Choices	yes	no
Multi-level Hierarchy	yes	yes
Multiple Metrics	yes	yes
Tested on the Cloud?	no	yes
Attributes		
# of Portfolios	8	1266
# of Alternatives	10	746807040
# of Metrics	2	4
# of Cases	20	2987228160

8.9 Scaling performance

It is tough to make predictions, especially about the future.

Yogi Berra

Two different architecture alternative spaces are not sufficient to characterize the performance of the ideas in this dissertation and the created implementation. The ability to generate valid architecture alternative spaces was created to further explore the ability of RAAM to apply to large system of systems architecture alternative problems. Randomly generated architecture alternative spaces were used to analyze how RAAM scales with the number of tasks in the architecture alternative space. The maximum number of subtasks per task, the maximum number of systems per task, and the number of systems had much smaller effects on the performance of the RAAM implementation than the number of tasks in an architecture alternative space.

The randomly generated architecture alternative space had a specified number of tasks. The maximum number of subtasks per task was specified and the maximum number of systems. A task hierarchy was created by starting at the given root and choosing a randomly distributed system from the possible systems list. The compute

function was chosen randomly from a list of compute functions for the purpose of this testing. The four aggregation functions used were the sum, product, min, and max functions. There are liner and non-linear options.

8.9.1 Generation of Architecture Alternative Spaces

The generation of an architecture alternative space for testing the RAAM implementation must create valid architecture alternative spaces. For these experiments, all tasks can only have one set of valid subtasks. The maximum number of potential systems is an input to the process. The maximum number of subtasks per task must all be specified. The number of systems is also specified at the beginning of the algorithm.

First, all of the tasks are made. The tasks are made by starting with a root task called TASKN, where N is the number of tasks. A random number of subtasks up to the specified maximum number of tasks is created to be the subtasks of TASKN. The created tasks are added to a stack and popped off of the stack to have their subtasks created. The process continues until a task is created where more creating the subtasks would create more tasks than specified. When that occurs, the remaining tasks on the stack are used as the subtasks. All of the created tasks that do not have any subtasks are marked as leaf tasks.

The systems need to be created next. The total number of systems is specified and they are created with random scores for a mission independent metric. This score is not currently used but is available for later experimentation. Each leaf task has a random number of systems up to the specified maximum number of systems that are chosen from the set of available systems. A score is created for each task and system pair using a random number generator.

Finally, each non-leaf task must have an aggregation and transformation function chosen for it. The transformation function is currently the identity function. The

aggregation function was chosen from a set of four aggregation functions, summation, a product, the minimum, and the maximum. Other aggregation functions are possible but were not tested in this case. The compute option for each non-leaf task was randomly chosen from the set of available aggregation functions.

8.9.2 Performance Scaling

As previously discussed, the main driver of the run time of an architecture alternative is the number of tasks in the architecture. The tests were run on an Intel Core 2 Duo E7200 CPU running at 2.53GHz. Only a single core was used.

A desirable property of the generation and execution algorithm is that the algorithm has favorable scaling with regard to input size changes. Big O notation is a way of describing program growth rates. Ideally an algorithm scales linearly or sublinearly with the growth of the input size ($O(n)$ or less).

Figure 74 shows how the run time of an alternative scales with the number of tasks in the architecture. For comparison, a linear fit was created from the data points where the number of tasks is 800 or lower. The algorithms are roughly linear until the number of tasks exceeds 800. This effect is shown in Figure 75. Both the highest and lowest times are shown to demonstrate the variance as the number of tasks is increased. In the linear portion of the graph the variance between the fastest and slowest test cases is small. Large DoD architectures tend to have a maximum of around one thousand tasks. Since the generated architectures were randomly created, there is a variance in the runtimes. Below eight hundred tasks, the runtime of any created architecture alternative is roughly the same.

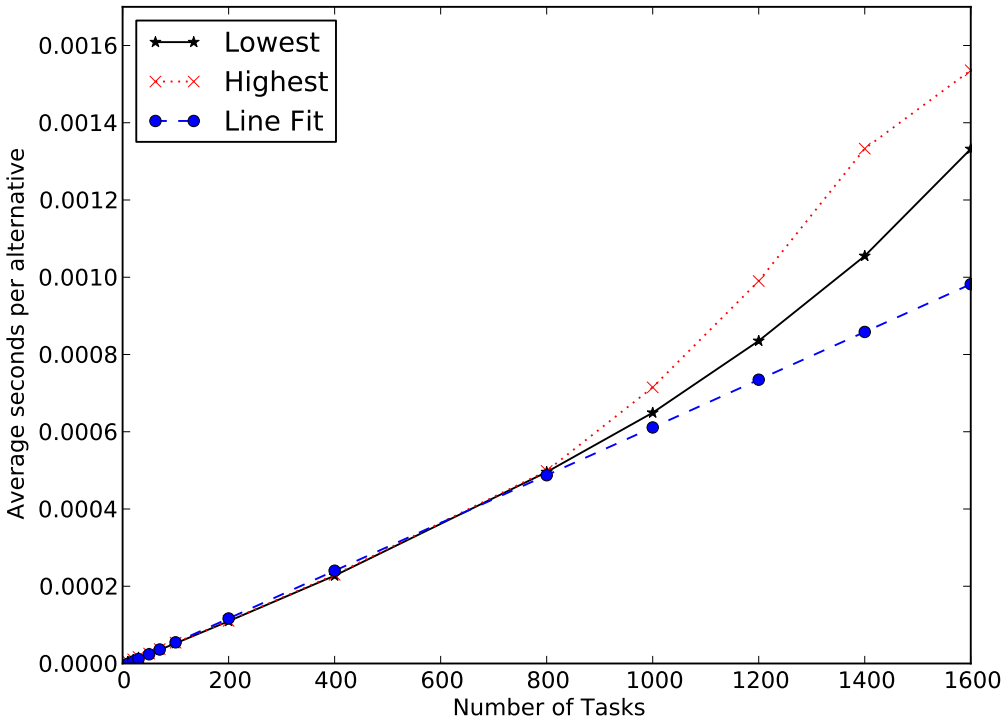


Figure 74: Seconds per Alternative as a Function of the Number of Tasks

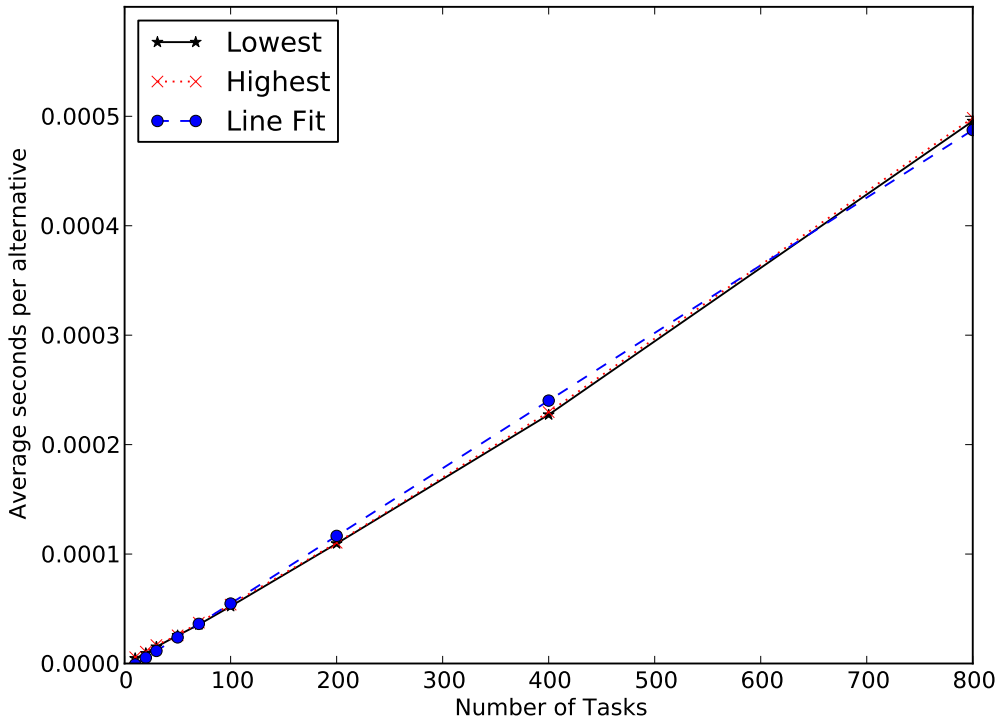


Figure 75: Seconds per Alternative as a Function of the Number of Tasks (Zoom)

Figures 74 and 75 show that predicting performance accurately is possible for architectures that are not excessive in size. An architecture with 800 tasks only requires half of a millisecond per architecture alternative. As a point of comparison, Davis et. al. [118] report that calculations of an architecture alternative require seconds to complete which results in over 94 years of computation if each SEAD alternative is evaluated for a second for each metric. Using RAAM on the cloud the computational time for evaluating the complete SEAD alternatives is under five minutes.

Investigating the reason for the divergence from the linear growth led to Figure 76 which shows the average seconds per task as a function of the number of tasks. The average time per task is below 0.6 microseconds until the number of tasks exceeds 800. When evaluating architecture alternatives with a small number of tasks, there is more measurement uncertainty. When varying between 100 to 800 tasks in an architecture alternative they differ by under 16%. As the number of tasks increases past 800, there is more variance in the average seconds per task. The variance is most likely attributable to cache issues as the problem size increases. The experiment shows that it is feasible to estimate the runtime of the architecture design study and that predicted problem sizes are handled well.

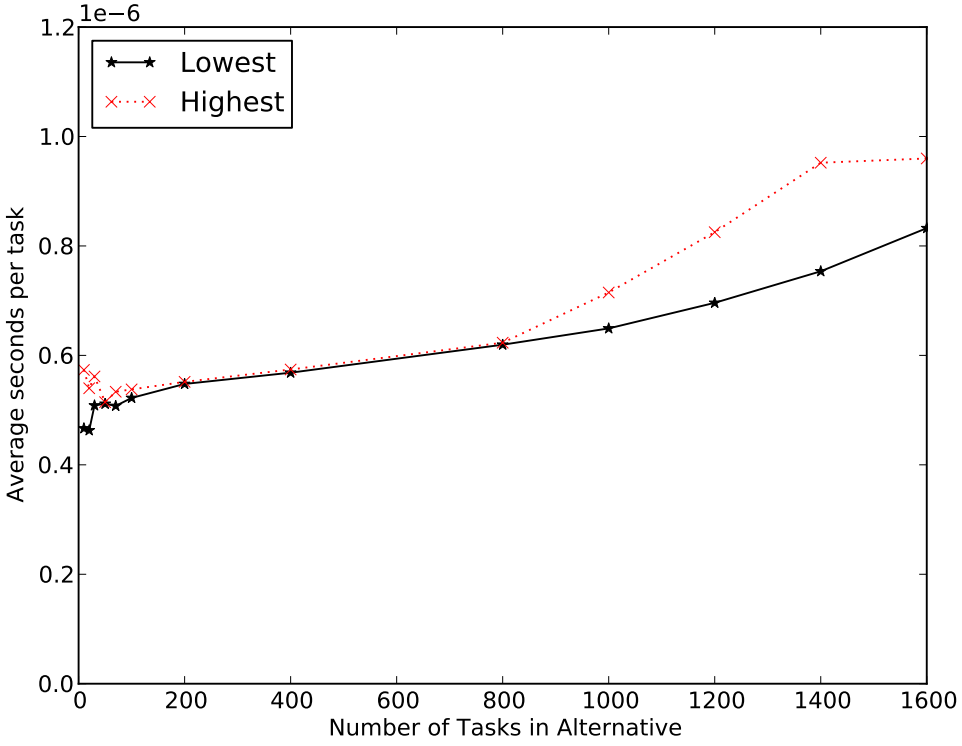


Figure 76: Average Seconds per Task as a Function of the Number of Tasks

8.10 Model Creation Complexity

Model creation complexity is a difficult topic to rigorously address. For the purposes of this manuscript, model creation complexity is based on the number of semantic concepts, the time required to create the RAAM input and the required supporting infrastructure for creating a model. DoDAF will be used to compare semantic concepts.

The RAAM methodology aimed to reduce the model creation complexity for the analyst. The semantic constructs should be simple to understand and simple to use. The DoDAF conceptual model has twelve different key concepts while RAAM has six key concepts. The twelve DoDAF conceptual models are:

1. Activity

2. Resource
3. Capability
4. Condition
5. Desired Effect
6. Measure
7. Measure Type
8. Location
9. Guidance
10. Project
11. Vision
12. Skill

Admittedly, DoDAF's scope is much larger than RAAM. The six key concepts for RAAM are:

1. Capability
2. Task
3. System
4. Metric
5. Compute
6. Portfolio Compute

Fewer key concepts means that the analyst has less to learn and understand to be able to do a RAAM enabled analysis. The analyst can focus on only the required elements for the initial design concept. Future analysis will entail more complicated mental models, but the concepts in RAAM are sufficient for early conceptual design of system of systems architectures.

The time required to create a model is an unmeasurable quantity. Different people, different organizations, or different analysis goals can all change the time to create input models for RAAM. However, qualitative assessments are possible. Current DoDAF modeling takes on the order of weeks to create system of systems models. RAAM models can be created with times on the order of days. The reduction in model creation time points to lower model complexity.

The supporting infrastructure for DoDAF is large. Often the data must be stored in a database system that is DoDAF standard compliant but based on proprietary tools. The different DoDAF products can use a variety of diagramming techniques including tables, IDEF, UML, and SysML. Seven different viewpoints are used with multiple artifacts per viewpoint. Specialized software is used to manipulate the artifacts in DoDAF. Multiple software packages may be required to create a DoDAF model. In addition, DoDAF does not have a computational element, so executable models must also be created. RAAM requires minimal infrastructure. The input files are plain text and can be edited with text editors. Future work would undoubtedly create specialized tools for manipulating a RAAM input file. In RAAM, the conceptual model, computational model, and metric scores are stored in the same file and s-expression based format. Executable models are automatically generated in RAAM.

The model creation complexity improvements were addressed qualitatively based on the SEAD example. The low number of concepts allows for the ability for analysts to keep the entire analysis structure in their head. The SEAD example was created

in a short amount of time. Minimal software infrastructure was used to create the SEAD example.

CHAPTER IX

CONCLUSIONS

> I am glad you have a Cat, but I do not believe it So remarkable a cat as My Cat.
>
> T.S. Eliot

The Department of Defense is attempting to transition to a capability based mindset for future acquisition programs. Early phase Pre-Milestone A analysis is being asked to provide studies with larger scopes than were analyzed before. The fiscal pressure on the Department of Defense requires more and more traceability into the decisions that are made that define an architecture. Warfare systems are becoming more complex and require system of systems thinking. The system of systems architecture alternative space is discrete and the computational models are often non-linear.

Surrogate models are difficult to fit across such a large design space with discrete elements. Broad sampling may miss important non-linear interactions. The manuscript describes Rapid Architecture Analysis Modeling, a method and framework which allows the exploration of the entire system of systems architecture design space. By exploring the entire design space, the decision maker can be confident that they have found the optimum in the multidimensional, multimodal space.

Comparing across multiple metrics can complicate optimization algorithms. Multimetric tools are difficult to compose from the different conceptual models associated with each tool. The different conceptual models may need different inputs or slightly different inputs. Rapid Architecture Analysis Modeling provides a unified conceptual model which reduces the required information gathering and models the different metrics simultaneously. Since only the information that is directly involved with

the analysis is required, the method is appropriate for conceptual design where the analyst does not have much information about the end product.

The dissertation started with discussing the motivations for pursuing the research. Next a subset of relevant topics was discussed that was used to quantify the current status of system of systems architecture portfolios analysis of alternatives. The research objective was to:

> **Develop a new methodology for compactly describing architecture alternatives that improves capability based conceptual analysis by reducing runtime and model creation complexity as compared to existing executable architecture models with limited to no fidelity loss.**

A new methodology has been developed that compactly describes architecture alternatives. The models execute quickly and are simple to define. The fidelity of the models is limited only by the effort from the analyst.

The methodology was used with a canonical example, a Suppression of Enemy Air Defenses example, and randomly generated architectures of varying sizes. The methodology proved to be a good way to organize the analysis of a system of systems architecture with the side benefit of low runtime speed. The core algorithms for the generation and evaluation of alternatives have good scaling performance for the predicted problem sizes.

RAAM does not exclude the use of existing state of the art tools for multi-objective optimization. The existing tools are not required for system of systems with fewer than a few trillion alternatives. Problems with more than forty binary decisions may require optimization algorithms. RAAM is designed in a hierarchical manner, which facilitates a 'divide and conquer' approach to large problem sizes. If the problem size is too big, RAAM can be executed on lower level tasks, or instructed to drop the lower level tasks until a different phase of the design effort. The architecture decomposition

can grow as the understanding of the problem progresses.

9.0.1 Integration with Existing Tools

The ideas represented in Rapid Architecture Alternative Modeling that describe an architecture are designed to be compatible with industry standards. Mappings are possible that convert RAAM inputs into other architectural modeling languages such as SysML or DoDAF. The architectures created for the down selection can be used in later phases of design. The architectures described with the RAAM should be a good starting point for further architecture development.

9.1 Contributions

9.1.1 RAAM

The RAAM methodology was created to enable more thorough and traceable system of systems architecture analysis. The methodology is as simple as possible to gain insight for system of systems analysis. The shown semantics impose a way of thinking onto the methodology which enables the automatic creation of executable models. The semantics (concepts) must be communicated, which requires a syntax (the structure and the symbols that represent the concepts). A domain specific language was tailored to the system of systems architecture portfolio conceptual phase analysis. Elements of RAAM should be useful for other phases of design but this was not proved in this thesis. The modeling language and its outputs have fed easily into decision support frameworks.

As compared to other architecture evaluation methods, RAAM is orders of magnitude faster for reasonable amounts of fidelity as shown in Figure 77, with information from [74].

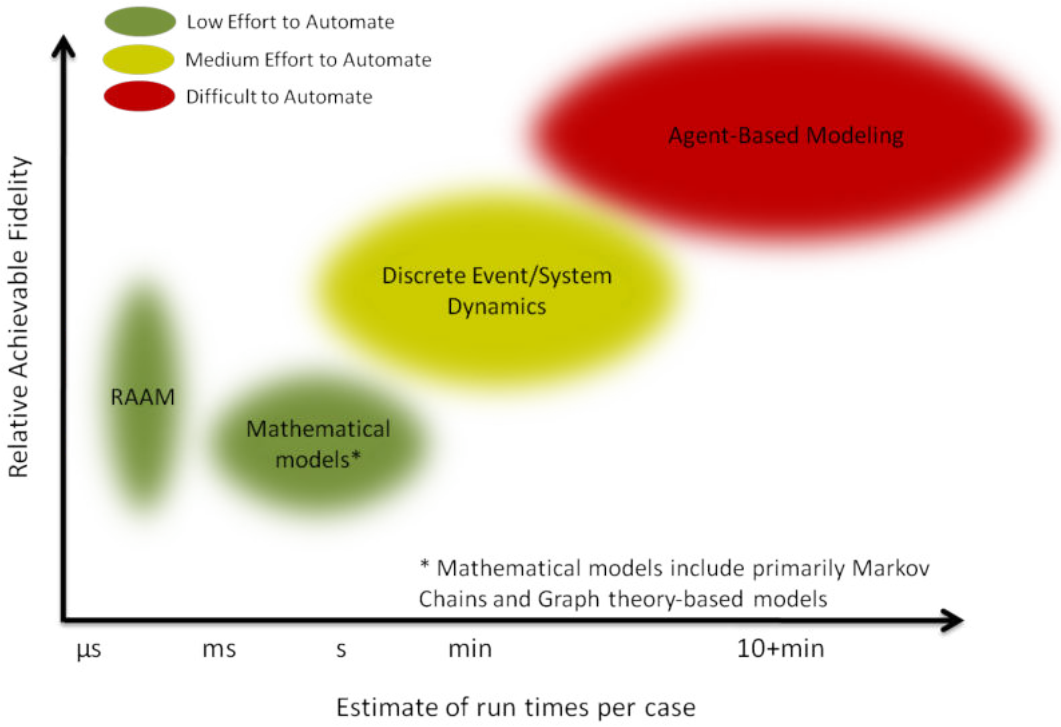


Figure 77: Fidelity vs Time per Case

9.1.2 Automatic Generation of Architecture Alternatives

The automatic generation of architecture alternatives is something that has been possible before. In this case more alternatives are possible because the structure of the architecture is varied in addition to the composition of the portfolio. Both operational and system level trades are possible in the same framework.

9.1.3 Automatic Creation of Executable Models

The automatic generation of architecture alternatives has also been possible before, but it is not often utilized and was often more specialized than as defined in this manuscript. The work defines a meta-model or a model of how to build models. The analyst can then build the model that they are familiar with and trust but in the RAAM framework. The language is designed to be executed very quickly on computers after it has been translated. It is possible to create many different models

for many different metrics of interest in the same tool. Previously, multiple tools and modeling methods were often required to model all of the metrics of interest.

9.1.4 Techniques for Managing Data Output

The RAAM methodology includes methods to make the analysis of the large amounts of output data tractable. Streaming algorithms for data reduction and visualizations are used to manage large amounts of data. Using portfolio views can provide a way to reduce the data storage requirements by providing a two step process to decision makers. The first step downselects to a set of promising portfolios and then the second step downselects to an architecture alternative.

9.1.5 Framework and Methodology

The framework and methodology tie the previously mentioned contributions together. By having in mind from the beginning of the research effort all of the eventual uses of the RAAM framework it is possible to design the method to have highly attractive properties. The combination of a new way to describe the problem (RAAM), automatic generation of architecture alternatives, and automatic creation of executable models of architecture alternatives into a framework that works together allows the faster runtime and an easier job for the analyst.

9.2 Limitations

The RAAM methodology is not perfect and has limitations. Due to the fact that RAAM was designed for the front end of early design phases, compromises have been made to enable the full exploration of the design space.

9.2.1 Problem Size

The problem size that RAAM is capable of handling is eventually limited by computing resources. As the number of alternatives grows exponentially with the number of decisions, the number of possible decisions is limited. The following discussion will

use binary decisions were the choice is between two options. The choice is either between two different systems that will accomplish a task, or between two sets of subtasks that will accomplish a task. This results in a best case analysis of the number of decisions possible, but does not take into account removing infeasible cases. Thirty binary decisions is approximately one billion alternatives. Forty binary decisions is approximately one trillion alternatives. Every addition of ten choices results in approximately one thousand ($2^{10} = 1024$ times more alternatives to consider. This results in a requirement for either one thousand times more computers or one thousand times more time for ten more decisions. Large organizations with 1000 times the computing power and the ability to compute for three and a half days can consider approximately fifty decisions. To counter the limitation caused by the combinatorial explosion, the decision makers should analyze the system of systems in a recursive manner. The architecture alternatives will initially begin with less detail and then be refined when decisions are known. This also reduces wasted effort in collecting detailed data on the architecture.

9.2.2 Possible Computational Models

Another limitation of RAAM is that not all computational models for system of systems architectures fit into the aggregation and transformation framework. Many computational models that do not at first glance fit into the aggregation and transformation framework can be adapted to the framework, while other computational models can not. The aggregation and transformation framework can not provide time series data that is sometimes required in system of systems analysis. For that reason, RAAM will not be able to replace discrete event simulations or petri-net simulations.

9.2.3 Optimization Limitations

Associativity is when the order of operations does not matter as long as the sequence of operands it not changed, e.g: $(1 + 2) + 3 = 1 + (2 + 3)$). Communativity is when

the order of the operands is allowed to change and does not change the result, e.g: $1 + 2 = 2 + 1$. Exploitation of properties such as associativity or communativity is not done in the current RAAM implementation. Due to the general nature of the aggregation and transformation functions, which are defined by the analyst and can be any function, optimization can not take advantage of associativity or communativity. By limiting the possible set of aggregation and transformation functions, finding the optimum architecture can be found directly without evaluating all of the architecture alternatives. In general, with multiple metrics, and unknown decision maker preferences at the time of computational modeling, it is not possible to directly solve for the optimum architecture alternative.

9.3 Future Work

The RAAM methodology has a variety of improvements or additions that will make the methodology more useful.

A library of reusable tasks and systems should be developed for different communities. For the military community, task lists such as the Universal Joint Task List (UJTL) or other service specific task lists can be translated into the RAAM syntax. Vetted computational models can be associated with the tasks from the UJTL for common military metrics of interest. With a known set of tasks, the collection of system to task scores can be done in a distributed manner. The system to task scores can be reused once collected and shared between different analyses.

The current RAAM implementation was tested with a canonical example, a Suppression of Enemy Air Defenses example, and randomly created examples. Larger models the size of the randomly created models with realistic data are an obvious next step. Innovative methods for gathering system to task metric scores from subject matter experts can be explored. Bringing together subject matter experts to do scoring for systems to tasks can be difficult. Alternatives such as using internet based

data collection will allow more data gathering. The internet based data collection can be more targeted which will reduce overall time requirements to gather the data.

The current RAAM implementation is correct, but is a prototype implementation. Optimization of the RAAM implementation may result in a 10-1000x speed up. This estimation is based on theoretical CPU throughput. In addition, the RAAM framework is amenable to being adapted to Graphics Processing Units (GPUs). Fang et. al. have accelerated MapReduce with GPUs. [63] Many current and planned supercomputer and cluster computing systems are utilizing GPU based computation for large speedups in computational time. The independence of the evaluation of an architecture alternative allows for the use of GPUs. In addition, evaluation of task hierarchies with the same operational decision can take advantage of the Single Instruction Multiple Data (SIMD) based instruction sets, as used on GPUs and CPUs. Another way of saying the same thing is that the RAAM algorithms can be vectorized.

This research represents a preliminary attempt at evaluating the full decision space of a system of systems architecture alternative study. Further research is required to determine the utility of RAAM outside of academic uses. Real world data on system of systems analyses is often proprietary, but the methodology should be compared against other methodologies. It will be useful to understand the sensitivities of the final architecture alternative choice to different choices of computational models and differing opinions of subject matter experts.

APPENDIX A

RAAM INTERNALS

Important pieces of the source code for RAAM are detailed in this chapter. The main file is organized as follows:

⟨*raam.lisp*⟩≡
  ⟨*Package Information*⟩
  ⟨*Global Variables*⟩
  ⟨*Task Definitions*⟩
  ⟨*Capability Definitions*⟩
  ⟨*Metric Definitions*⟩
  ⟨*System Definitions*⟩
  ⟨*Compute Definitions*⟩
  ⟨*Finalize the Architecture*⟩
  ⟨*Top N Alternatives*⟩
  ⟨*Making Decisions*⟩
  ⟨*Portfolio Functions*⟩
  ⟨*Utility Functions*⟩
  ⟨*Aggregation Functions*⟩
  ⟨*Transformation Functions*⟩
  ⟨*Main Program*⟩

The functions and macros are defined in the asdl.raam package.

⟨*Package Information*⟩≡
```
(defpackage :asdl.raam
  (:use :common-lisp))
(in-package :asdl.raam)
```

The global variables are used to keep global state and simplify certain functions. ***systems***, ***tasks***, ***capabilities***, ***decisions***, and ***dep-metrics*** are used to keep a list of the respective objects. While executing a given architecture for a given metric ***current-metric*** stores the current metric. The main task is stored in ***mtsk***. ***top-n***, ***top-n-alternatives**, ***oec-function***, ***portfolio-scores***, and ***portfolio-isodata*** are used when storing the data from runs. ***oec-function*** is designed to be redefined in a raam model file. ***top-n-alternatives*** is initialized with zero as the minimum value. The minimum value should be changed from zero if the ***oec-function*** can have negative values.

⟨*Global Variables*⟩≡

```
(defparameter *systems* '())
(defparameter *tasks* '())
(defparameter *capabilities* '())
(defparameter *decisions* '())
(defparameter *dep-metrics* '())
(defparameter *indep-metrics* '())

(defparameter *current-metric* 'none)
(defparameter *mtsk* '())

(defparameter *top-n* 100)
(defparameter *top-n-alternatives*
  (make-list *top-n* :initial-element '(0.0 '() '())))
  ;0.0 is assumed to be OEC minimum.
(defparameter *oec-function* #'+)
(defparameter *portfolio-scores*
  (make-hash-table :test #'equal))
(defparameter *portfolio-isodata*
  (make-hash-table :test #'equal))

(defparameter *sorted-system-symbols* '())
(defparameter *sorted-task-symbols* '())
(defparameter *allowed-systems* '())
  ;set to nil for all systems
```

The task definitions section first defines a class that stores information about a task. A task is comprised of two parts, a task object that is an instantiation of the task class and a task method that specializes on a metric. Currently, the slot for *parent* is not used, but may be useful for future algorithms. To make the printing of a task better looking, the *print-object* function was defined to print the name of the task. The task macro first checks if the task has been defined or not. If not, then a task object is created with the name, description and subtasks. A symbol with the task name is bound to the task object. If the task does not have any subtasks, it is a leaf node and is marked appropriately. The task object is added to the list of tasks stored in ***tasks***. Finally, a generic mmthod that dispatches based on a metric is defined.

⟨*Task Definitions*⟩≡

```
;; (task NAME DESCRIPTION LIST-OF-SUBTASKS)
(defclass task ()
  ((name
    :accessor name
    :initarg :name)
   (description
    :accessor description
    :initarg :description)
   (subtasks
    :accessor subtasks
    :initarg :subtasks)
;;   (parent ; This is filled in later
;;    :accessor parent)
   (possible-systems
    :accessor possible-systems)
   (score
    :accessor score)
   (leaf?
    :accessor leaf?
    :initform nil)
   (current-decision
    :accessor current-decision
    :initform (lambda (x) (nth 0 x)))))
```

```
(defmethod print-object ((tsk task) stream)
  (princ (name tsk) stream))

(defmacro task (nam description &rest subtasks)
  `(if (boundp ',nam)
       (format t "Already added that task: ~a~%" ',nam)
       (let ((task-object
              (make-instance 'task
                             :name ',nam
                             :description ,description
                             :subtasks ',subtasks)))
         (defparameter ,nam task-object)
         (if (equal '(()) (subtasks task-object))
             (setf (leaf? task-object) t)
             '())
         (setf *tasks* (cons task-object *tasks*))
         (defgeneric ,nam (metrikk)))))
```

Next, the capability keyword is defined. First a class is created that holds the information about the capability. The name of the capability is used for printing to the screen in *print-object*. The capability macro first checks if the capability is defined, if not, a new capability object is created. The created capability object is bound to the name of the capability and added to the list of capabilities. The mission dependent metrics are added to the ***dep-metrics*** list. For each mission dependent metric, a new metric class is created and a variable is created that contains an instance of that metric class. Finally, the main task is stored in ***mtsk***.

⟨*Capability Definitions*⟩≡

```
;; (capability NAME DESCRIPTION MAIN-TASK METRICS
;;   PORTFOLIO-METRICS)
(defclass capability ()
  ((name
    :accessor name
    :initarg :name)
   (description
    :accessor description
    :initarg :description)
   (main-task
    :accessor main-task
    :initarg :main-task)
   (metrics
    :accessor metrics
    :initarg :metrics)
   (portfolio-metrics
    :accessor portfolio-metrics
    :initarg :portfolio-metrics)))

(defmethod print-object ((capabilit capability) stream)
  (princ (name capabilit) stream))

(defmacro capability (nam description main-task metrics
                      portfolio-metrics)
  `(if (boundp ',nam)
       (format t "Already added that capabilitity: ~a~%"
               ',nam)
       (let ((capability-object
              (make-instance 'capability
```

```
              :name ',nam
              :description ,description
              :main-task ,main-task
              :metrics ',metrics
              :portfolio-metrics
                ',portfolio-metrics)))
(defparameter ,nam capability-object)
(setf *capabilities*
      (cons capability-object
            *capabilities*))
(defparameter *dep-metrics* ',metrics)
(defparameter *indep-metrics* ',portfolio-metrics)
,@(loop for metrik in metrics
     collect `(defclass ,metrik () ()))
,@(loop for metrik in metrics
     collect
       `(defvar ,metrik
          (make-instance ',metrik)))
(defparameter *mtsk* ',main-task))))
```

The metric keyword is defined below. If the system is in ***allowed-systems*** then a method is created for the task to system pairing for each metric. The method is named task-system, specialized on the metric type, and converts the score to a float. The coercion to a float would need to be changed if other types of metric scores are used, for example if the scores were distributions. The method is added to the current task's list of subtasks and a symbol with the method name is bound to the system associated with it.

⟨*Metric Definitions*⟩≡
  ⟨*Metric Defintion Helpers*⟩

```
(defmacro metric (task sys &rest metric-score-pairs)
  (if (or (null *allowed-systems*)
          (find sys *allowed-systems*))
      (let ((fun-name
             (intern
              (concatenate 'string (string task) "-"
                           (string sys)))))
        `(progn
           (defgeneric ,fun-name (metrikk))
           ,@(loop for (metri score) in metric-score-pairs
                collect `(defmethod ,fun-name
                             ((metrikk ,metri))
                           ,(coerce `,score
                                    'single-float)))
           (add-score-to-task ,task ',fun-name)
           (defparameter ,fun-name ',sys)))))
```

The system definitions are defined below. The information for a system is only required when using mission independent metrics or for meta data about the system. First, a class is defined to hold the information about a system including its name, description, and mission independent metrics (or system-attributes). The system macro tests if the system is defined, and then creates a system object which is added to ***system***. A variable is created with the system object bound to the name of the system.

⟨*System Definitions*⟩≡

```
;; (system NAME DESCRIPTION SYSTEM-ATTRIBUTES)
(defclass system ()
  ((name
    :accessor name
    :initarg :name)
   (description
    :accessor description
    :initarg :description)
   (system-attributes
    :accessor system-attributes
    :initarg :system-attributes)
   (possible-tasks
    :accessor possible-tasks)))

(defmacro system (nam description &rest system-attributes)
  `(if (boundp ',nam)
       (format t  "Already added that system: ~a~%" ',nam)
       (let ((system-object
              (make-instance 'system
                             :name ',nam
                             :description ,description
                             :system-attributes
                               ',system-attributes)))
         (defparameter ,nam system-object)
         (setf *systems*
               (cons system-object *systems*)))))
```

The compute definitions are defined below. A function to get the currently selected children is defined in *get-selected-children*. The compute macro associates the defined aggregation and transformation functions to the task method. The transformation function scales the result of the aggregation function operating on the currently selected children.

The mission independent metrics are stored in an association list and are extracted with the function *get-metric*. Mission dependent metrics are created with the *portfolio-compute* macro.

⟨*Compute Definitions*⟩≡

```
(defun get-selected-children (taskk)
  (funcall (current-decision taskk) (subtasks taskk)))

(defmacro compute (task metric aggregation transformation)
  `(defmethod ,task ((metrikk ,metric))
     (let ((selected-children
            (get-selected-children ,task)))
       (declare (optimize (safety 0) (speed 3)))
       (,transformation
        (apply #',aggregation
               (mapcar #'(lambda (x)
                           (funcall x *current-metric*))
                       selected-children))))))

(defun get-metric (metric sys)
  (cadr (assoc metric (system-attributes sys))))

(defmacro portfolio-compute (nam aggregation
                             transformation)
  `(defun ,nam ()
     (,transformation
      (apply #',aggregation
             (mapcar #'(lambda (sys)
                         (get-metric ',nam (symbol-value sys)))
                     (get-decided-systems))))))
```

After all of the information is read into the computer, we must finalize the structures and add a function to choose between the possible systems. *add-system-selectors-to-leafs* is used to add a method specializer for each system, task, and metric triplet.

⟨*Finalize the Architecture*⟩≡

```
(defmacro add-selector-to-leaf (tas metri)
 `(defmethod ,tas ((metrikk ,metri))
    (funcall (car (get-selected-children ,tas))
             *current-metric*)))

(defmacro add-system-selectors-to-leafs ()
  `(progn ,@(loop for taskk in
                  (loop for task in *tasks*
                     when (leaf? task)
                     collect (name task))
                append (loop for metrik in *dep-metrics*
                             collect
                               `(add-selector-to-leaf
                                 ,taskk ,metrik)))))
```

⟨*Top N Alternatives*⟩≡

```
(defun insert-into-top (oec-val-alt)
  (when (> (car oec-val-alt)
           (car (elt *top-n-alternatives* 0)))
    (setf (elt *top-n-alternatives* 0) oec-val-alt)
    ;replace the min
    ;put the min in the front, the sort is destructive...
    (setf *top-n-alternatives*
          (sort *top-n-alternatives* #'< :key #'car))))

(defun update-top-n (oec-function metrics)
  (let* ((oec-val (apply oec-function metrics))
         (oec-val-alt
          (list oec-val
                (list-current-decisions) metrics)))
    (insert-into-top oec-val-alt)))

(defun output-top-n-to-file (n)
  (with-open-file
      (stream
       (concatenate 'string (write-to-string n) ".data")
       :direction :output :if-exists :supersede
       :if-does-not-exist :create)
    (format stream "'~a~%" *top-n-alternatives*)))
```

⟨*Making Decisions*⟩≡

```
(defun update-decision (taskk n)
  (setf (current-decision taskk)
        (lambda (x) (nth (- n 1) x))))

(defun symbol-to-task-resolver (decisions)
  (loop for (task-symbol n) in decisions
       collect (list `,(symbol-value task-symbol) n)))

(defun make-listed-decisions (decisions)
  (loop for (tsk n) in decisions
       do (update-decision tsk n)))

(defun make-wasteful-decisions
    (made-decisions not-made-decisions)
  (cond ((null not-made-decisions)
         (make-listed-decisions made-decisions)
         (output-decisions-and-metric-scores))
        ((equal (cadar not-made-decisions) 1)
         (make-wasteful-decisions
          (cons (car not-made-decisions) made-decisions)
          (cdr not-made-decisions)))
        (t (make-wasteful-decisions
            (cons (car not-made-decisions) made-decisions)
            (cdr not-made-decisions))
           (make-wasteful-decisions
            made-decisions
            (cons (list (caar not-made-decisions)
                        (- (cadar not-made-decisions) 1))
                  (cdr not-made-decisions))))))

(defun get-decided-tasks (mtsk)
  (let ((childrens (get-selected-children mtsk)))
    (if (null childrens)
        '()
        (list (name mtsk)
              (loop for child in childrens
                 collect
                   (if (leaf?
                        (symbol-value child))
                       child
                       (get-decided-tasks
```

```
                        (symbol-value child))))))))

(defun get-decided-leafs (mtsk)
  (let ((childrens (get-selected-children mtsk)))
    (if (null childrens)
        '()
        (loop for child in childrens
           collect (if (leaf? (symbol-value child))
                       (car (get-selected-children
                             (symbol-value child)))
                       (get-decided-leafs
                        (symbol-value child)))))))
```

The permutate decisions (p-d) algorithm starts with a list of the task nodes with more than one decision called *not-made-decisions*. The other input is an empty list called *made-decisions*. The algorithm makes decisions and places them into the *made-decisions* list. At each iteration of the algorithm one of two things happens, based on if the *not-made-decisions* list is empty or not. When the *not-made-decisions* list is empty and there are no decisions on the next level, the main task function is called for each metric which returns the scores for a given alternative. If *not-made-decisions* is empty and there are decisions on the next level, then the permutate decisions algorithm is called on the next level of decisions with an empty *made-decisions* list. Otherwise, then the permutate decisions algorithm moves the first decision in the *not-made-decision* list to the *made-decisions* list. If there are not any more decision options, then the algorithm is done. If there are decision options left in the first decision of the *not-made-decisions* list, then the first decision of the *not-made-decisions* list is changed to a different set of subtasks for that decision and the permutate decisions algorithm is run again.

⟨*Portfolio Functions*⟩≡

```
(defun decisions-to-portfolio (mtsk)
  (let ((task-system-mappings
         (flatten (get-decided-leafs mtsk))))
    (mapcar #'symbol-value task-system-mappings)))

(defun get-decided-systems ()
  (sort (delete-duplicates
         (decisions-to-portfolio
          (symbol-value *mtsk*)))
                      #'string<))

(defun add-to-average (new-val old-val n)
  (+ old-val (/ (- new-val old-val) (1+ n))))

(defstruct portfolio-data
  metric-scores
  oec-score
  n)

(defstruct portfolio-isodata p-systems p-tasks)

(defun update-portfolio-score-for-all-metrics
    (systems oec-val metric-vals)
  (let ((pd (gethash systems *portfolio-scores*)))
    (cond ((null pd) (setf
                      (gethash systems *portfolio-scores*)
```

```
                    (make-portfolio-data
                     :metric-scores metric-vals
                     :oec-score oec-val
                     :n 1.0))
             (setf (gethash systems *portfolio-isodata*)
                   (make-portfolio-isodata
                    :p-systems systems
                    :p-tasks (flatten ;;fix w/task hierarchy
                              (get-decided-tasks
                               (symbol-value *mtsk*))))))
            (t (let ((old-val (portfolio-data-oec-score pd))
                     (n (portfolio-data-n pd))
                     (old-metric-vals
                      (portfolio-data-metric-scores pd)))
                 (setf (portfolio-data-metric-scores pd)
                       (loop for metric-score in metric-vals
                          for old-metric-score in
                          old-metric-vals
                          collect
                            (add-to-average
                             metric-score
                             old-metric-score n)))
                 (setf (portfolio-data-oec-score pd)
                       (add-to-average oec-val old-val n))
                 (incf (portfolio-data-n pd)))))))

(defun update-running-average-for-all-metrics
    (oec-function metric-vals)
  (let ((oec-val (apply oec-function metric-vals))
        (systems (sort (delete-duplicates
                        (decisions-to-portfolio
                         (symbol-value *mtsk*)))
                       #'string<)))
    (update-portfolio-score-for-all-metrics
     systems oec-val metric-vals)))


(defun output-portfolio-decisions-and-metric-scores ()
;;(format t "~%~{~a, ~}~{~a, ~}" (list-current-decisions)
  (update-running-average-for-all-metrics *oec-function*
                          (loop for mtric in *dep-metrics*
                             do (setf
                                 *current-metric*
                                 (symbol-value mtric))
                             collect
```

```
                                (funcall
                                 (symbol-function *mtsk*)
                                 *current-metric*))))

(defun make-wasteful-portfolio-decisions
    (made-decisions not-made-decisions)
  (make-wasteful-decisions-with-output-function
   made-decisions not-made-decisions
   #'output-portfolio-decisions-and-metric-scores))


(defun make-wasteful-decisions-with-output-function
    (made-decisions not-made-decisions out-function)
  (cond ((null not-made-decisions)
         (make-listed-decisions made-decisions)
         (funcall out-function))
        ((equal (cadar not-made-decisions) 1)
         (make-wasteful-decisions-with-output-function
          (cons (car not-made-decisions) made-decisions)
          (cdr not-made-decisions) out-function))
        (t (make-wasteful-decisions-with-output-function
            (cons (car not-made-decisions) made-decisions)
            (cdr not-made-decisions) out-function)
           (make-wasteful-decisions-with-output-function
            made-decisions
            (cons (list (caar not-made-decisions)
                        (- (cadar not-made-decisions) 1))
                  (cdr not-made-decisions))
            out-function))))

(defparameter *limit* 2)
(defun
    make-wasteful-decisions-wof-and-limit
    (made-decisions not-made-decisions out-function)
  (cond
    ((equal *limit* 0) ())
    ((null not-made-decisions)
     (make-listed-decisions made-decisions)
     (funcall out-function)
     (setf *limit* (- *limit* 1)))
    ((equal (cadar not-made-decisions) 1)
     (make-wasteful-decisions-wof-and-limit
      (cons (car not-made-decisions) made-decisions)
      (cdr not-made-decisions) out-function))
    (t (make-wasteful-decisions-wof-and-limit
```

```
        (cons (car not-made-decisions) made-decisions)
        (cdr not-made-decisions) out-function)
       (make-wasteful-decisions-wof-and-limit
        made-decisions
        (cons (list (caar not-made-decisions)
                    (- (cadar not-made-decisions) 1))
              (cdr not-made-decisions)) out-function))))

(defun get-list-of-leaves ()
  (loop for (tsk n) in (list-decisions)
               when (leaf? tsk)
               collect tsk))

(defun get-list-of-non-leaves ()
  (loop for (tsk n) in (list-decisions)
                   when (not (leaf? tsk))
                   collect tsk))


(defun list-of-x-to-bits (current-list full-list)
  (loop for item in full-list
       collect (if (member item current-list) 1 0)))

(defun add-prefix-and-fix-for-jmp (prefix item)
  (concatenate 'string prefix
               (string-capitalize
                (substitute #\  #\- (symbol-name item)))))

(defun add-prefixs (prefix lst)
  (loop for item in lst
       collect (add-prefix-and-fix-for-jmp prefix item)))

(defun output-for-jmp-portfolio-data-for-all-metrics ()
  ;Horrible let* due to code printing width...
  (let* ((fmt-st "~&~{~a,~}~{~a,~}~{~a,~}OEC Average,")
         (fmt-sr "Number of Alternatives,")
         (fmt-tr "Number of Systems")
         (fmt-str (concatenate 'string
                               fmt-st fmt-sr fmt-tr)))
    (format
     t fmt-str
     (add-prefixs "Task " *sorted-task-symbols*)
     (add-prefixs "System " *sorted-system-symbols*)
     (add-prefixs "Metric Average " *dep-metrics*)))
  (loop for k being the hash-keys in *portfolio-isodata*
```

```
     using (hash-value v)
     do (let ((pd (gethash k *portfolio-scores*)))
          (format t "~&~{~a,~}~{~a,~}~{~a,~}~a,~a,~a,"
                  (list-of-x-to-bits
                   (portfolio-isodata-p-tasks v)
                   *sorted-task-symbols*)
                  (list-of-x-to-bits
                   (portfolio-isodata-p-systems v)
                   *sorted-system-symbols*)
                  (portfolio-data-metric-scores pd)
                  (portfolio-data-oec-score pd)
                  (portfolio-data-n pd)
                  (apply #'+
                         (list-of-x-to-bits
                          (portfolio-isodata-p-systems v)
                          *sorted-system-symbols* ))))))

    ;collect (list v (gethash v *portfolio-scores*))))

(defun output-decisions-and-metric-scores-2 ()
  (format t "~%~{~a, ~}~{~a, ~}" (list-current-decisions)
          (loop for mtric in *dep-metrics*
             do (setf *current-metric*
                      (symbol-value mtric))
             collect (funcall (symbol-function *mtsk*)
                              *current-metric*))))

(defun dont-output-just-run ()
  (loop for mtric in *dep-metrics*
     do (setf *current-metric* (symbol-value mtric))
     collect (funcall (symbol-function *mtsk*)
                      *current-metric*)))

(defun dont-run-just-make-decisions ()
  '())

(defun symbol-to-task (tsklist)
  (loop for task-symbol in tsklist
       collect `,(symbol-value task-symbol)))

(defun get-selected-system-from-task (tsk)
  (symbol-value
   (car
    (funcall (current-decision tsk) (subtasks tsk)))))
```

```
(defun get-children (tsk)
  (if (leaf? tsk)
      '()
      (funcall (current-decision tsk) (subtasks tsk))))

(defun get-next-level-decided-children (tasklist)
  (loop for tsk in tasklist
     append (symbol-to-task (get-children tsk))))

(defun get-next-level-decided-children2 (tasklist)
  (loop for (tsk n) in tasklist
     append (symbol-to-task (get-children tsk))))

(defun get-all-levels-decided-children (tsklist)
  (if tsklist
      (append tsklist (get-all-levels-decided-children
                           (get-next-level-decided-children tsklist)))
      '()))

;(defun make-decision-list (decisions)
;  (loop for (task-symbol n) in decisions
;       collect (list task-symbol n)))

(defun make-decision-list (tsklist)
  (loop for tsk in tsklist
       collect (list tsk (length (subtasks tsk)))))

(defun p-d (m-d n-m-d outfunc) ;Permutate Decisions
  (cond ((null n-m-d)
         (make-listed-decisions m-d)
         (let ((next-level-decided-children
                (get-next-level-decided-children2 m-d)))
           (if (null next-level-decided-children)
               (funcall outfunc)
               (p-d '() (make-decision-list next-level-decided-children)
                      outfunc))))
        (t (p-d (cons (car n-m-d) m-d)
                  (cdr n-m-d) outfunc)
           (unless (equal (cadar n-m-d) 1)
             (p-d m-d
                    (cons (list (caar n-m-d)
                                (- (cadar n-m-d) 1))
                          (cdr n-m-d))
                    outfunc)))))
```

```
(defun make-decisions-with-output-function (outfunc)
  (p-d '()
       (make-decision-list
        (get-next-level-decided-children
          (symbol-to-task (list *mtsk*))))
       outfunc))

(defun p-d-limit (m-d n-m-d outfunc) ;Permutate Decisions
  (cond ((equal *limit* 0) ())
        ((null n-m-d)
         (make-listed-decisions m-d)
         (decf *limit*)
         (let ((next-level-decided-children
                (get-next-level-decided-children2 m-d)))
           (if (null next-level-decided-children)
               (progn (funcall outfunc))
               (p-d-limit
                '()
                (make-decision-list next-level-decided-children)
                outfunc))))
        ((equal (cadar n-m-d) 1)
         (p-d-limit (cons (car n-m-d) m-d)
                    (cdr n-m-d) outfunc))
        (t (p-d-limit (cons (car n-m-d) m-d)
                      (cdr n-m-d) outfunc)
           (p-d-limit m-d
                      (cons (list (caar n-m-d)
                                  (- (cadar n-m-d) 1))
                            (cdr n-m-d))
                      outfunc))))

(defun make-decisions-with-output-function-and-limit (outfunc)
  (p-d-limit '()
             (make-decision-list
              (get-next-level-decided-children
                (symbol-to-task (list *mtsk*))))
             outfunc))
```

⟨*Metric Defintion Helpers*⟩≡

```
(defun add-score-to-task (taskk fun-nam)
  "Adds scores to all leaf tasks."
  (if (equal '(()) (subtasks taskk))
      (setf (subtasks taskk) (list (list fun-nam)))
      (push (list fun-nam) (subtasks taskk))))
```

⟨*Utility Functions*⟩≡
  ⟨*Decision Functions*⟩
  ⟨*Determine Number of Alternatives*⟩
  ⟨*Execution Functions*⟩

```
(defun remove-dashes-from-symbol-to-string (sym)
  "Removes dashes in a symbol and converts it to a string.
   Used to interoperate with software such as JMP."
  (substitute-if #\_ (complement #'alphanumericp)
                 (prin1-to-string sym)))

(defun flatten (lst)
 "Removes nesting in a list."
 (cond ((null lst) '())
       ((atom lst) (list lst))
       (t (nconc (flatten (car lst))
                 (flatten (cdr lst))))))
```

⟨*Decision Functions*⟩≡

```
(defun list-number-of-decisions (taskks)
  "Creates a list of the number of decisions for each task.
   It is sorted in the same order as the taskks arguement."
  (loop for taskk in taskks
       collect (list-length (subtasks taskk))))

(defun list-decisions ()
  "Creates a list of all tasks with more than one decision."
  (loop for tsk in *tasks*
        for decision in (list-number-of-decisions *tasks*)
       when (> decision 1)
     collect (list tsk decision)))

(defun list-all-decisions ()
  "Creates a list of all tasks with decisions."
  (loop for tsk in *tasks*
        for decision in (list-number-of-decisions *tasks*)
        collect (list tsk decision)))

(defun list-current-decisions ()
  "Returns a list of the currently made decisions as a set of
   decision positions."
  (loop for tsk in *tasks*
     collect
       (position (funcall (current-decision tsk) (subtasks tsk))
                 (subtasks tsk))))

(defun list-current-decisions-2 ()
  "Returns a list of the current decisions with either the subtasks
   or the system that is chosen"
  (loop for tsk in *tasks*
     collect
       (if (leaf? tsk)
           (symbol-value (car (funcall (current-decision tsk) (subtasks tsk))))
           (funcall (current-decision tsk) (subtasks tsk)))))

(defun list-operational-decisions ()
  "Returns a list of only the operational decisions"
  (loop for tsk in *tasks*
        for decision in (list-number-of-decisions *tasks*)
       when (and (> decision 1) (not (leaf? tsk)))
```

```
      collect (list tsk decision)))
```

⟨*Determine Number of Alternatives*⟩≡

```
(defun count-number-of-alternatives (capabil)
  "Count the number of alternatives given the capability."
  (count-from-task-tree (make-task-tree (main-task capabil))))

(defun make-task-tree (taskk)
  (loop for taskkks in (subtasks taskk)
     collect (loop for tas in taskkks
                collect (if (eql 'symbol (type-of (symbol-value tas)))
                            tas
                            (make-task-tree (symbol-value tas))))))

;for style warning for mutually recursive functions
(declaim (ftype function count-from-task-tree))
(defun count-from-task-tree-helper (task-decision-tree)
      (apply #'* (mapcar #'count-from-task-tree task-decision-tree)))

(defun count-from-task-tree (task-decision-tree)
  (if (listp task-decision-tree)
      (apply #'+ (mapcar #'count-from-task-tree-helper task-decision-tree))
      1))

(defun count-alternatives (tsk)
  "Count the number of alternatives given any task."
  (count-from-task-tree (make-task-tree tsk)))
```

⟨*Execution Functions*⟩≡

```
(defun output-decisions-and-metric-scores-old (decisions)
;  (setf *decisions* decisions)
;  (format t "~%~{~a, ~}~{~a, ~}" decisions
  (loop for mtric in *dep-metrics*
     do (setf *current-metric* (symbol-value mtric))
     collect (funcall (symbol-function *mtsk*) *current-metric*)))
;)

(defun output-decisions-and-metric-scores ();(decisions)
;;(format t "~%~{~a, ~}~{~a, ~}" (list-current-decisions)
  (update-top-n *oec-function*
                (loop for mtric in *dep-metrics*
                   do (setf *current-metric* (symbol-value mtric))
                   collect (funcall (symbol-function *mtsk*) *current-metric*)))
)
;;)
```

⟨*Aggregation Functions*⟩≡

```
(defun sum (&rest lst)
  (the single-float (apply #'+ lst)))

(defun product (&rest lst)
  (the single-float (apply #'* lst)))
```

⟨*Transformation Functions*⟩≡

```
(defun ident (val)
  "The identity transformation function. Does nothing to the input."
  val)
```

The current common lisp implementation used in this work is sbcl. When an executable is created from this file, the executable will load the file passed to it. It is expected that the **load-file-name** is a raam model.

⟨*Main Program*⟩≡

```
;for now, to interface with the rest of the world, going
;to use sbcl specific stuff. If I don't use sbcl,
;it should fail
#+sbcl
(defun main-prog ()
  (let ((load-file-name (second sb-ext:*posix-argv*)))
    (if load-file-name
        (load load-file-name) ;; error handling goes here.
        '())))

(main-prog)
```

APPENDIX B

PARALLELIZATION CODE

Listing B.1: run-on-cloud.lisp

```
1
2  #-sbcl
3  (error "Not running sbcl, no other common lisp implementations are
       supported yet.")
4
5  ;(load "cfml.lisp")
6  (load "sead_seam.lisp")
7  ;(load "simp.lisp")
8
9  (in-package :asdl.raam)
10
11 (defparameter *ec2-base-dir* "/usr/bin/")
12 (defparameter *cert-file*
       "../amazonec2/cert-F5GXKT3IDOWGVUTJGWKC3KHFX3GNI4XT.pem")
13 (defparameter *pk-file*
       "../amazonec2/pk-F5GXKT3IDOWGVUTJGWKC3KHFX3GNI4XT.pem")
14 (defparameter *auth* (concatenate 'string " -K " *pk-file* " -C "
       *cert-file*))
15 ;(defparameter *cur-ami-id* "ami-ccf405a5") ;micro
16 ;(defparameter *ami-type* "t1.micro") ;micro
17 (defparameter *cur-ami-id* "ami-a6f504cf") ;small
18 (defparameter *ami-type* "m1.small") ;small
19 (defparameter *keypair-auth* " --key firstkeypair022311")
20 (defparameter *login-auth* "-i ../amazonec2/firstkeypair022311.pem")
21 (defparameter *ec2-instances* '())
22 (defparameter *image-name* "raam.exe")
23
24 (defparameter *start-wait* 15) ; seconds
25 (defparameter *data-wait* 15) ; seconds
26 (defparameter *image-wait* 5) ; seconds
27 (defparameter *let-boot-for-seconds* 45)
28
29 (defparameter *simulate-cloud* t)
30 (if *simulate-cloud*
31     (progn (setf *start-wait* 0)
32            (setf *data-wait* 0)
33            (setf *image-wait* 5)
34            (setf *let-boot-for-seconds* 0))
35   '())
36
37 (defparameter *pastec-p* t)
38 (if *pastec-p*
39     (progn (setf *simulate-cloud* nil)
```

```
40              (setf *start-wait* 0)
41              (setf *data-wait* 5)
42              (setf *image-wait* 2)
43              (setf *let-boot-for-seconds* 0))
44    '())
45
46  ;;Make an image that can be used to run the model on the cloud
        (executable)
47  (require 'sb-posix)
48
49  (defun make-deployable-image ()
50    (let ((pid (sb-posix:fork)))
51      (cond
52        ;((zerop pid) (print "saving teh image! [fake]") (sb-ext:quit))
53        ((zerop pid) (sb-ext:save-lisp-and-die *image-name* :executable t
              :toplevel (function main-prog))) ;main-prog is in cfml.lisp,
              it just loads a lisp file.
54        ((plusp pid) (sleep *image-wait*) (format t "~%~%~%~%") '())
55        (t (error "Didn't quite fork :-("))))
56
57  (make-deployable-image)
58
59  ;(defun get-column-at (describe-output n)
60  ;  (let* ((command-str (concatenate 'string "/bin/echo \""
        describe-output "\" | sed -n 0~2p" (write-to-string n)))
61  ;         (data-str (string-right-trim '(#\newline)
        (with-output-to-string (stream)
62  ;
        (sb-ext:run-program "/bin/sh" `("-c" ,command-str) :output
        stream)))))
63  ;     (loop for i = 0 then (1+ j)
64  ;           as j  = (position #\newline data-str :start i)
65  ;           collect (subseq data-str i j)
66  ;           while j)
67  ;
68  ;))
69
70  (defun run-command-get-string (command-str)
71    (let ((sim (not *simulate-cloud*)))
72          (if sim
73              (with-output-to-string (stream)
74                                              ;I went with this method so
                                                   that I could use pipes.
                                                   C'est la vie.
75                                              ;Update: I am not sure that I
                                                   want a sed dependancy.
                                                   Might remove the use of
                                                   pipes.
76                ;(format t "~&command-str: ~a" command-str)
77                (sb-ext:run-program "/bin/sh" `("-c" ,command-str)
                      :output stream))
78              (format t "~&**SIM**~a" command-str))))
79
80  (defun status-of-command-str (command-str)
```

```
81    (let ((sim (not *simulate-cloud*)))
82      (if sim
83          (progn
84            ;(format t "~&For status:~a" command-str)
85            (let ((exit-code (sb-ext:process-exit-code
                  (sb-ext:run-program "/bin/sh" '("-c" ,command-str)))))
86              ;(format t "~&statuscode: ~a" exit-code)
87              exit-code))
88          (progn
89            (format t "~&**SIM**~a" command-str)
90            0))))
91
92  (defun split-at-char (in-str charr)
93    (loop for i = 0 then (1+ j)
94          as j = (position charr in-str :start i)
95          collect (subseq in-str i j)
96          while j))
97
98  (defun remove-internal-whitespace (str) ;I also get rid of some problem
        chars like ':' (colon).
99    (setf *print-pretty* nil) ;turn off pretty printing to get rid of
        those damned newlines.
100   (let ((outstr (format nil "~a" (read-from-string (substitute #\. #\:
        (concatenate 'string "'(" str ")"))))))
101     (setf *print-pretty* t)
102     outstr))
103
104 (defun get-instance-id-ip-and-status (instance-str)
105   (when (not (string= "" instance-str))
106     ;(format t "~%instance-str:~A" (remove-internal-whitespace
          instance-str))
107     (loop for i = 0 then (1+ i)
108        for column in (split-at-char (remove-internal-whitespace
             instance-str) #\Space)
109        when (or (eql i 2) (eql i 4) (eql i 6))
110        collect (string-downcase column))))
111
112 (defun get-describe-output ()
113   (let ((command-str (concatenate 'string *ec2-base-dir*
        "ec2-describe-instances" *auth* " | sed -n /INSTANCE/p"))) ;here
        I can remove sed (todo)
114     (run-command-get-string command-str))) ;very safe to leave as 't'
115
116 (defun get-instance-ids-ips-and-status ()
117   (setf *ec2-instances*
118         (if *simulate-cloud*
119             '(("i-c9e2e7a5" "ec2-50-17-41-163.compute-1.amazonaws.com"
                  "running"))
120             (let ((describe-output (get-describe-output)))
121               (loop for instance-str in (split-at-char
                      (string-right-trim '(#\Newline) describe-output)
                      #\newline)
122                     collect (get-instance-id-ip-and-status
                        instance-str)))))))
```

```
123
124 ;(get-instance-ids-ips-and-status)
125 ;(print *ec2-instances*)
126
127 ;; Example command-output with newlines.
128 ;;"RESERVATION     r-3626a65b      860663546571    default
129 ;;INSTANCE        i-ddc7f9b1      ami-aa7083c3
      pending firstkeypair022311      0               t1.micro
      2011-03-02T18:12:18+0000        us-east-1c      aki-407d9529
                              monitoring-disabled
                                                      ebs
                                                    paravirtual
130 ;;"
131 (defun get-instance-id-from-start (command-output)
132   (subseq (second (split-at-char command-output #\newline)) 9 19))
        ;gah, magic numbers, I hoep they don't change their format...
133
134
135 (defun start-up-instance-on-the-cloud ()
136   (let* ((command-str (concatenate 'string *ec2-base-dir*
        "ec2-run-instances " *cur-ami-id* " --instance-type " *ami-type*
        *keypair-auth* *auth*))
137          (command-output
138           (if *simulate-cloud*
139               (coerce '(#\R #\E #\S #\E #\R #\V #\A #\T #\I #\O #\N
                    #\Tab #\r #\- #\3 #\c #\b #\7 #\3
140                           #\b #\5 #\1 #\Tab #\8 #\6 #\0 #\6 #\6 #\3 #\5
                              #\4 #\6 #\5 #\7 #\1 #\Tab #\d
141                           #\e #\f #\a #\u #\l #\t #\Newline #\I #\N #\S
                              #\T #\A #\N #\C #\E #\Tab #\i
142                           #\- #\c #\9 #\e #\2 #\e #\7 #\a #\5 #\Tab #\a
                              #\m #\i #\- #\a #\a #\7 #\0 #\8
143                           #\3 #\c #\3 #\Tab #\Tab #\Tab #\p #\e #\n #\d
                              #\i #\n #\g #\Tab #\f #\i #\r
144                           #\s #\t #\k #\e #\y #\p #\a #\i #\r #\0 #\2 #\2
                              #\3 #\1 #\1 #\Tab #\0 #\Tab
145                           #\Tab #\t #\1 #\. #\m #\i #\c #\r #\o #\Tab #\2
                              #\0 #\1 #\1 #\- #\0 #\3 #\-
146                           #\0 #\4 #\T #\0 #\2 #\: #\1 #\4 #\: #\3 #\8 #\+
                              #\0 #\0 #\0 #\0 #\Tab #\u #\s
147                           #\- #\e #\a #\s #\t #\- #\1 #\c #\Tab #\a #\k
                              #\i #\- #\4 #\0 #\7 #\d #\9 #\5
148                           #\2 #\9 #\Tab #\Tab #\Tab #\m #\o #\n #\i #\t
                              #\o #\r #\i #\n #\g #\- #\d #\i
149                           #\s #\a #\b #\l #\e #\d #\Tab #\Tab #\Tab #\Tab
                              #\Tab #\e #\b #\s #\Tab #\Tab
150                           #\Tab #\Tab #\Tab #\p #\a #\r #\a #\v #\i #\r
                              #\t #\u #\a #\l #\Tab #\Newline) 'string)
151               (run-command-get-string command-str)))
152           ;(eh (format t "Command output: ~a" command-output))
153           (instance-id (get-instance-id-from-start command-output)))
154     ;(format t "~%~%~S" (coerce command-output 'list))
155     (format t "~&Launched instance: ~a" instance-id)
156     command-str ; dummy, just to get rid of warning while testing
```

```
157       instance-id
158   ))
159
160   ;(start-up-instance-on-the-cloud)
161   ;(sleep 30)
162
163   (defun terminate-instance (instance-id-str)
164     (let ((command-str (concatenate 'string *ec2-base-dir*
          "ec2-terminate-instances " *auth* " " instance-id-str)))
165       (format t "~&Attempting to terminate ~a." instance-id-str)
166       (run-command-get-string command-str)))
167
168   (defun terminate-all-instances (instances)
169     (loop for (id ip status) in instances
170           when (string= "running" status)
171           do (terminate-instance id)))
172
173   ;(setf *ec2-instances* (get-instance-ids-ips-and-status))
174   ;(print *ec2-instances*)
175
176   (defun wait-until-started-and-get-ip (current-instance)
177     (format t "~&Waiting for ~a" current-instance)
178     (loop do (sleep *start-wait*);10
179        until (string= "running" (third (find current-instance
              (get-instance-ids-ips-and-status) :test #'string= :key
              #'car))))
180     (format t "~&Done waiting for: ~a" current-instance)
181     (second (find current-instance *ec2-instances* :test #'string= :key
          #'car)))
182
183   (defun wait-for-and-get-data (instance-ip n)
184     (format t "~&Waiting for ~a's data for number:~a" instance-ip n)
185     (let ((command-str (concatenate 'string "scp -o
          StrictHostKeyChecking=no " *login-auth* " ubuntu@" instance-ip
          ":~/" (write-to-string n) ".data " (write-to-string n) ".data")))
186     (loop do (sleep *data-wait*)
187           until (eql 0 (status-of-command-str command-str)))
188     (format t "~&Done waiting for ~a's data for number:~a" instance-ip
          n))) ; when it is successful, we have the file.
189
190
191   (defun copy-file-to-cloud-ip (filename instance-ip)
192     (let ((command-str (concatenate 'string "scp -o
          StrictHostKeyChecking=no " *login-auth* " " filename " ubuntu@"
          instance-ip ":~/" filename)))
193       (format t "~&**COPYING**~a" command-str)
194       (run-command-get-string command-str)))
195   ;scp -o StrictHostKeyChecking=no -i ../amazonec2/firstkeypair022311.pem
        seam.exe
196   ;ubuntu@ec2-50-16-124-232.compute-1.amazonaws.com:~/seam.exe
197
198   ;`(update-decision ,static-decision ,n)
199   ;sending the object? instead of the symbol
200   ;for the static decision.
```

```
201
202 (defun make-script-to-run (script-filename n static-decisions decisions)
203   ;decisions
204   (with-open-file (stream script-filename :direction :output :if-exists
        :supersede :if-does-not-exist :create)
205     (format stream
206             "~{~a~%~}"
207             (list
208              '(in-package :asdl.cfml)
209              '(make-wasteful-decisions
210                (symbol-to-task-resolver
211                 ',static-decisions)
212                (symbol-to-task-resolver
213                 ',(make-decision-list decisions)))
214              '(output-top-n-to-file ,n)
215              ;'(print *top-n-alternatives*)
216 ))))
217
218
219 (defun run-image-with-script (script-filename instance-ip)
220   (let ((command-str (concatenate 'string "ssh -o
        StrictHostKeyChecking=no " *login-auth* " ubuntu@" instance-ip "
        ./" *image-name* " " script-filename)))
221     (format t "~&**SSHING**~a" command-str)
222     (run-command-get-string command-str)))
223
224
225 (defparameter *node-prefix* "c4-")
226 (defparameter *number-of-nodes* 18)
227 (defparameter *number-of-cores-per-node* 12)
228 (defun get-cluster-computer-to-use (num)
229   (concatenate
230    'string *node-prefix*
231    (write-to-string
232     (mod num *number-of-nodes*))))
233
234 (defun run-script-on-cluster (script-filename cluster-computer)
235   (let ((command-str
236          (concatenate
237           'string
238           "ssh -o StrictHostKeyChecking=no "
239           cluster-computer
240           " 'cd /home/jiacobucci/gits/cfml/ && ./"
241           *image-name* " " script-filename "'")))
242     (run-command-get-string command-str)))
243
244 (defun wait-for-and-get-cluster-data (num)
245   (loop do (sleep *data-wait*)
246         until (eql 0 (status-of-command-str
247                       (concatenate 'string "ls " (write-to-string num)
                          ".data")))))
248   ;(format t "~&Cluster computer:~a  Num:~a" cluster-computer num))
249
```

```
250 (defun get-results-from-one-computer-pastec (n static-decisions
      decisions)
251   (let ((script-filename (concatenate 'string (write-to-string n)
        ".lsp"))
252         (cluster-computer (get-cluster-computer-to-use n)))
253     (make-script-to-run script-filename n static-decisions decisions)
254     (run-script-on-cluster script-filename cluster-computer)
255     (wait-for-and-get-cluster-data n)))
256     ;; (if (= 17 (mod n *number-of-nodes*))
257     ;;  (format t "~&Cluster computer:~a Num:~a" cluster-computer n)
258     ;;    '())))
259
260
261
262 ;I am going to assume that stuff works. Error checking comes later.
263 (defun get-results-from-one-computer (n static-decisions decisions)
264   ;(format t "~&  In get-results: ~a:~a" n static-decisions)
265   (let* ((script-filename (concatenate 'string (write-to-string n)
        ".lsp"))
266          (current-instance
267           ;(nth (- n 1) '("i-6d979201" "i-69979205")))
268           (start-up-instance-on-the-cloud))
269          (current-instance-ip
270           ;(nth (- n 1) '("ec2-50-17-85-200.compute-1.amazonaws.com"
                "ec2-50-16-70-132.compute-1.amazonaws.com"))))
271           (wait-until-started-and-get-ip current-instance)))
272     (format t "~& Sleeping for ~a for 60 seconds." n)
273     (sleep *let-boot-for-seconds*)
274     (copy-file-to-cloud-ip *image-name* current-instance-ip)
275
276     (make-script-to-run script-filename n static-decisions decisions)
277     (copy-file-to-cloud-ip script-filename current-instance-ip)
278
279     (run-image-with-script script-filename current-instance-ip)
280     (wait-for-and-get-data current-instance-ip n)
281     (format t "~&Got results from: ~a" current-instance)))
282
283
284 ;(print (type-of (car (car (list-decisions)))))
285
286
287 ;********************
288 ;(get-results-from-one-computer 1 'subtask1 (cdr (reverse
      (list-decisions))))
289 ;********************
290
291 ;(print (nth 0 '((SUBTASK1 2) (SUBTASK2 2) (SUBTASK4 2) (SUBTASK7 3))))
292
293
294
295 ;(terminate-all-instances *ec2-instances*)
296
297 (terpri)
298 (terpri)
```

```
299
300
301 (defun make-wasteful-decisions (made-decisions not-made-decisions)
302   (cond ((null not-made-decisions)
303          (make-listed-decisions made-decisions)
304          (output-decisions-and-metric-scores))
305         ((equal (cadar not-made-decisions) 1)
306          (make-wasteful-decisions (cons
307                                    (car not-made-decisions)
308                                    made-decisions)
309                                   (cdr not-made-decisions)))
310         (t (make-wasteful-decisions
311             (cons (car not-made-decisions) made-decisions)
312             (cdr not-made-decisions))
313            (make-wasteful-decisions
314             made-decisions
315             (cons (list (caar not-made-decisions)
316                         (- (cadar not-made-decisions) 1))
317                   (cdr not-made-decisions)))))))
318
319 (defparameter *highest-thread-id* 1)
320 (defun make-w-parallel-decisions (made-decisions not-made-decisions
321                                   dynamic-decisions)
322   (cond ((null not-made-decisions)
323          (launch-thread-with-sector
324           *highest-thread-id* made-decisions dynamic-decisions)
325          (incf *highest-thread-id*))
326         ((equal (cadar not-made-decisions) 1)
327          (make-w-parallel-decisions
328           (cons (car not-made-decisions) made-decisions)
329           (cdr not-made-decisions)
330           dynamic-decisions))
331         (t (make-w-parallel-decisions
332             (cons (car not-made-decisions) made-decisions)
333             (cdr not-made-decisions)
334             dynamic-decisions)
335            (make-w-parallel-decisions
336             made-decisions
337             (cons (list (caar not-made-decisions)
338                         (- (cadar not-made-decisions) 1))
339                   (cdr not-made-decisions))
340             dynamic-decisions))))
341
342 (defun launch-thread-with-sector (id-num static-decisions
343                                   dynamic-decisions)
344   (sb-thread:make-thread
345    (lambda ()
346      ;(format t "~&insidethread ~a" id-num)
347      (get-results-from-one-computer-pastec
348       id-num
349       static-decisions dynamic-decisions))))
350
351 (defun make-parallel-decisions-helper
352     (static-possible-decisions dynamic-decisions)
```

```
353     (make-wasteful-parallel-decisions
354      '()
355      static-possible-decisions dynamic-decisions))
356
357   ;; ;Works, I think. Need to work on how to make it use more computers.
358   ;; ;Maybe a list of variable decisions and a list of (soon to be) fixed
        decisions
359   ;; (defun make-parallel-decisions (static-decisions dynamic-decisions)
360   ;;   (let ((parallel-decision-num (apply #'* (mapcar #'cadr
        static-decisions)))
361   ;;         (parallel-decision-name (car (car decisions)))
362   ;;         (rest-of-decisions (cdr decisions)))
363   ;;       (loop for n from 1 to parallel-decision-num do
364   ;;         ;(format t "~&outsidethread~a" n)
365   ;;         ;(sleep 1.5)
366   ;;         (sb-thread:make-thread
367   ;;          (let ((m n)
368   ;;                (p-d-name parallel-decision-name)
369   ;;                (r-o-d rest-of-decisions))
370   ;;            (lambda ()
371   ;;              ;(sleep (random 5))
372   ;;              (format t "~&insidethread~a" m)
373   ;;              ;(get-results-from-one-computer m p-d-name r-o-d)
374   ;;              ;(sleep 2)
375   ;;                                        ;(sb-thread:terminate-thread
        sb-thread:*current-thread*)
376   ;;            )
377   ;;            ))
378   ;;          ;))
379
380
381   ;;   ;(update-decision parallel-decision-name n)
382   ;;         ;(make-wasteful-decisions '() rest-of-decisions)
383   ;; )))
384
385
386   ;; ((REMOVE-FROM-TARGET-LIST 2) (ASSIGN-WEAPON-AND-PLATFORM 2)
387   ;;  (ASSESS-ENGAGEMENT-CAPABILITY 2) (TASK-SENSOR 2))
388
389   ;; ((BATTLE-DAMAGE-ASSESSMENT 7) (ENGAGE-TO-DISRUPT 3)
390   ;;  (ENGAGE-TO-DESTROY 7) (UPDATE-TARGET-LIST 3)
391   ;;  (TRACK-UNTIL-STOPPED 4) (DISCRIMINATE-DECOYS 7)
392   ;;  (MANAGE-TARGET-MOVEMENT-DATA 3) (IDENTIFY 7)
393   ;;  (PASS-WARNING-AND-LOCATION-DATA 3) (FUSE-SENSOR-DATA 3)
        (WIDE-AREA-SEARCH 5)
394   ;;  (DETERMINE-SENSOR-AVAILABILITY 2)
395   ;;  (RECONCILE-TARGET-PRIORITIES 2))
396
397
398   ;************
399   ;216
400   (defparameter *just-temp-static-decisions*
401     '((REMOVE-FROM-TARGET-LIST 2) (ASSIGN-WEAPON-AND-PLATFORM 2)
402       (ASSESS-ENGAGEMENT-CAPABILITY 2)
```

```
403       (PASS-WARNING-AND-LOCATION-DATA 3)
404       (FUSE-SENSOR-DATA 3) (MANAGE-TARGET-MOVEMENT-DATA 3)))
405       ;8
406  ;(defparameter *just-temp-static-decisions*
407  ;'((REMOVE-FROM-TARGET-LIST 2) (ASSIGN-WEAPON-AND-PLATFORM 2)
408  ;  (ASSESS-ENGAGEMENT-CAPABILITY 2)))
409  (defparameter *just-temp-dynamic-decisions*
410    '((BATTLE-DAMAGE-ASSESSMENT 7) (ENGAGE-TO-DISRUPT 3)
411      (ENGAGE-TO-DESTROY 7) (UPDATE-TARGET-LIST 3)
412      (TRACK-UNTIL-STOPPED 4) (DISCRIMINATE-DECOYS 7)
413      (IDENTIFY 7) (WIDE-AREA-SEARCH 5)
414      (DETERMINE-SENSOR-AVAILABILITY 2) (TASK-SENSOR 2)
415      (RECONCILE-TARGET-PRIORITIES 2)))
416  ;(defparameter *just-temp-static-decisions*  '((SUBTASK4 2)))
417  ;(defparameter *just-temp-dynamic-decisions*  '((SUBTASK7 3) (SUBTASK2
         2) (SUBTASK1 2)))
418  ;(print (apply #'* (mapcar #'cadr *just-temp-static-decisions*)))
419  (make-w-parallel-decisions
420   '() *just-temp-static-decisions* *just-temp-dynamic-decisions*)
421
422  ;***********
423
424  ;(get-instance-ids-ips-and-status)
425  ;(format t "~&ec2-instances:~a" *ec2-instances*)
426  ;(terminate-all-instances *ec2-instances*)
427
428  ;(make-parallel-decisions '((a 10) (b 4)))
429
430
431  ;(sleep 5)
432
433  ;(format t "~%~%Before all that:~a" (sb-thread:list-all-threads))
434
435  (loop for thread in (sb-thread:list-all-threads)
436        do
437        ;(print thread)
438        (if (equal thread sb-thread:*current-thread*)
439            '()
440            (sb-thread:join-thread thread)))
441
442  ;;(get-instance-ids-ips-and-status)
443  ;;;(terminate-all-instances *ec2-instances*)
444  ;(format t "~&Sleeping for a mintute to let them terminate")
445  ;(sleep 60)
446  ;;;(format t "~&ec2-instances:~a" *ec2-instances*)
447
448  ;;;(format t "~%~%After all that: ~a" (sb-thread:list-all-threads))
449
450  ;; (time (loop repeat 10 do
451  ;;     (make-parallel-decisions (reverse (list-decisions)))))
452  (terpri)
453  (terpri)
```

APPENDIX C

RAW SOURCE CODE FOR RAAM

C.1 Source Code of raam.lisp

Listing C.1: raam.lisp

```
1  (defpackage :asdl.raam
2    (:use :common-lisp))
3  (in-package :asdl.raam)
4
5  (defparameter *systems* '())
6  (defparameter *tasks* '())
7  (defparameter *capabilities* '())
8  (defparameter *decisions* '())
9  (defparameter *dep-metrics* '())
10 (defparameter *indep-metrics* '())
11
12 (defparameter *current-metric* 'none)
13 (defparameter *mtsk* '())
14
15 (defparameter *top-n* 100)
16 (defparameter *top-n-alternatives*
17   (make-list *top-n* :initial-element '(0.0 '() '())))
18   ;0.0 is assumed to be OEC minimum.
19 (defparameter *oec-function* #'+)
20 (defparameter *portfolio-scores*
21   (make-hash-table :test #'equal))
22 (defparameter *portfolio-isodata*
23   (make-hash-table :test #'equal))
24
25 (defparameter *sorted-system-symbols* '())
26 (defparameter *sorted-task-symbols* '())
27 (defparameter *allowed-systems* '())
28   ;set to nil for all systems
29
30 ;; (task NAME DESCRIPTION LIST-OF-SUBTASKS)
31 (defclass task ()
32   ((name
33     :accessor name
34     :initarg :name)
35    (description
36     :accessor description
37     :initarg :description)
38    (subtasks
39     :accessor subtasks
40     :initarg :subtasks)
41 ;;   (parent ; This is filled in later
```

```
42 ;;     :accessor parent)
43     (possible-systems
44      :accessor possible-systems)
45     (score
46      :accessor score)
47     (leaf?
48      :accessor leaf?
49      :initform nil)
50     (current-decision
51      :accessor current-decision
52      :initform (lambda (x) (nth 0 x)))))
53
54 (defmethod print-object ((tsk task) stream)
55   (princ (name tsk) stream))
56
57 (defmacro task (nam description &rest subtasks)
58   `(if (boundp ',nam)
59        (format t "Already added that task: ~a~%" ',nam)
60        (let ((task-object
61               (make-instance 'task
62                              :name ',nam
63                              :description ,description
64                              :subtasks ',subtasks)))
65          (defparameter ,nam task-object)
66          (if (equal '(()) (subtasks task-object))
67              (setf (leaf? task-object) t)
68              '())
69          (setf *tasks* (cons task-object *tasks*))
70          (defgeneric ,nam (metrikk)))))
71
72 ;; (capability NAME DESCRIPTION MAIN-TASK METRICS
73 ;;   PORTFOLIO-METRICS)
74 (defclass capability ()
75   ((name
76     :accessor name
77     :initarg :name)
78    (description
79     :accessor description
80     :initarg :description)
81    (main-task
82     :accessor main-task
83     :initarg :main-task)
84    (metrics
85     :accessor metrics
86     :initarg :metrics)
87    (portfolio-metrics
88     :accessor portfolio-metrics
89     :initarg :portfolio-metrics)))
90
91 (defmethod print-object ((capabilit capability) stream)
92   (princ (name capabilit) stream))
93
94 (defmacro capability (nam description main-task metrics
95                       portfolio-metrics)
```

```
96    `(if (boundp ',nam)
97         (format t "Already added that capabilitity: ~a~%"
98                 ',nam)
99         (let ((capability-object
100              (make-instance 'capability
101                             :name ',nam
102                             :description ,description
103                             :main-task ,main-task
104                             :metrics ',metrics
105                             :portfolio-metrics
106                               ',portfolio-metrics)))
107           (defparameter ,nam capability-object)
108           (setf *capabilities*
109                 (cons capability-object
110                       *capabilities*))
111           (defparameter *dep-metrics* ',metrics)
112           (defparameter *indep-metrics* ',portfolio-metrics)
113           ,@(loop for metrik in metrics
114                collect `(defclass ,metrik () ()))
115           ,@(loop for metrik in metrics
116                collect
117                  `(defvar ,metrik
118                     (make-instance ',metrik)))
119           (defparameter *mtsk* ',main-task))))
120 (defun add-score-to-task (taskk fun-nam)
121   "Adds scores to all leaf tasks."
122   (if (equal '(()) (subtasks taskk))
123       (setf (subtasks taskk) (list (list fun-nam)))
124       (push (list fun-nam) (subtasks taskk))))
125
126 (defmacro metric (task sys &rest metric-score-pairs)
127   (if (or (null *allowed-systems*)
128           (find sys *allowed-systems*))
129       (let ((fun-name
130              (intern
131               (concatenate 'string (string task) "-"
132                            (string sys)))))
133         `(progn
134            (defgeneric ,fun-name (metrikk))
135            ,@(loop for (metri score) in metric-score-pairs
136                 collect `(defmethod ,fun-name
137                                     ((metrikk ,metri))
138                                     ,(coerce `,score
139                                              'single-float)))
140            (add-score-to-task ,task ',fun-name)
141            (defparameter ,fun-name ',sys)))))
142 ;; (system NAME DESCRIPTION SYSTEM-ATTRIBUTES)
143 (defclass system ()
144   ((name
145     :accessor name
146     :initarg :name)
147    (description
148     :accessor description
149     :initarg :description)
```

```
150     (system-attributes
151      :accessor system-attributes
152      :initarg :system-attributes)
153     (possible-tasks
154      :accessor possible-tasks)))
155
156 (defmacro system (nam description &rest system-attributes)
157   `(if (boundp ',nam)
158        (format t "Already added that system: ~a~%" ',nam)
159        (let ((system-object
160               (make-instance 'system
161                              :name ',nam
162                              :description ,description
163                              :system-attributes
164                                ',system-attributes)))
165          (defparameter ,nam system-object)
166          (setf *systems*
167                (cons system-object *systems*)))))
168
169 (defun get-selected-children (taskk)
170   (funcall (current-decision taskk) (subtasks taskk)))
171
172 (defmacro compute (task metric aggregation transformation)
173   `(defmethod ,task ((metrikk ,metric))
174      (let ((selected-children
175             (get-selected-children ,task)))
176        (declare (optimize (safety 0) (speed 3)))
177        (,transformation
178         (apply #',aggregation
179                (mapcar #'(lambda (x)
180                            (funcall x *current-metric*))
181                          selected-children))))))
182
183 (defun get-metric (metric sys)
184   (cadr (assoc metric (system-attributes sys))))
185
186 (defmacro portfolio-compute (nam aggregation
187                              transformation)
188   `(defun ,nam ()
189      (,transformation
190       (apply #',aggregation
191              (mapcar #'(lambda (sys)
192                          (get-metric ',nam (symbol-value sys)))
193                      (get-decided-systems))))))
194
195
196 (defmacro add-selector-to-leaf (tas metri)
197  `(defmethod ,tas ((metrikk ,metri))
198     (funcall (car (get-selected-children ,tas))
199              *current-metric*)))
200
201 (defmacro add-system-selectors-to-leafs ()
202   `(progn ,@(loop for taskk in
203                   (loop for task in *tasks*
```

```
204                     when (leaf? task)
205                     collect (name task))
206                 append (loop for metrik in *dep-metrics*
207                              collect
208                                `(add-selector-to-leaf
209                                  ,taskk ,metrik)))))
210
211
212
213 (defun insert-into-top (oec-val-alt)
214   (when (> (car oec-val-alt)
215            (car (elt *top-n-alternatives* 0)))
216     (setf (elt *top-n-alternatives* 0) oec-val-alt)
217     ;replace the min
218     ;put the min in the front, the sort is destructive...
219     (setf *top-n-alternatives*
220           (sort *top-n-alternatives* #'< :key #'car))))
221
222 (defun update-top-n (oec-function metrics)
223   (let* ((oec-val (apply oec-function metrics))
224          (oec-val-alt
225           (list oec-val
226                 (list-current-decisions) metrics)))
227     (insert-into-top oec-val-alt)))
228
229 (defun output-top-n-to-file (n)
230   (with-open-file
231       (stream
232        (concatenate 'string (write-to-string n) ".data")
233        :direction :output :if-exists :supersede
234        :if-does-not-exist :create)
235     (format stream "'~a~%" *top-n-alternatives*)))
236
237
238 (defun update-decision (taskk n)
239   (setf (current-decision taskk)
240         (lambda (x) (nth (- n 1) x))))
241
242 (defun symbol-to-task-resolver (decisions)
243   (loop for (task-symbol n) in decisions
244         collect (list `,(symbol-value task-symbol) n)))
245
246 (defun make-listed-decisions (decisions)
247   (loop for (tsk n) in decisions
248         do (update-decision tsk n)))
249
250 (defun make-wasteful-decisions
251     (made-decisions not-made-decisions)
252   (cond ((null not-made-decisions)
253          (make-listed-decisions made-decisions)
254          (output-decisions-and-metric-scores))
255         ((equal (cadar not-made-decisions) 1)
256          (make-wasteful-decisions
257           (cons (car not-made-decisions) made-decisions)
```

```
258           (cdr not-made-decisions)))
259         (t (make-wasteful-decisions
260             (cons (car not-made-decisions) made-decisions)
261             (cdr not-made-decisions))
262            (make-wasteful-decisions
263             made-decisions
264             (cons (list (caar not-made-decisions)
265                         (- (cadar not-made-decisions) 1))
266                   (cdr not-made-decisions))))))
267
268 (defun get-decided-tasks (mtsk)
269   (let ((childrens (get-selected-children mtsk)))
270     (if (null childrens)
271         '()
272         (list (name mtsk)
273                 (loop for child in childrens
274                    collect
275                      (if (leaf?
276                           (symbol-value child))
277                          child
278                          (get-decided-tasks
279                           (symbol-value child))))))))
280
281 (defun get-decided-leafs (mtsk)
282   (let ((childrens (get-selected-children mtsk)))
283     (if (null childrens)
284         '()
285         (loop for child in childrens
286            collect (if (leaf? (symbol-value child))
287                        (car (get-selected-children
288                              (symbol-value child)))
289                        (get-decided-leafs
290                         (symbol-value child)))))))
291
292
293 (defun decisions-to-portfolio (mtsk)
294   (let ((task-system-mappings
295          (flatten (get-decided-leafs mtsk))))
296     (mapcar #'symbol-value task-system-mappings)))
297
298 (defun get-decided-systems ()
299   (sort (delete-duplicates
300          (decisions-to-portfolio
301           (symbol-value *mtsk*)))
302                       #'string<))
303
304 (defun add-to-average (new-val old-val n)
305   (+ old-val (/ (- new-val old-val) (1+ n))))
306
307 (defstruct portfolio-data
308   metric-scores
309   oec-score
310   n)
311
```

```
312 (defstruct portfolio-isodata p-systems p-tasks)
313
314 (defun update-portfolio-score-for-all-metrics
315     (systems oec-val metric-vals)
316   (let ((pd (gethash systems *portfolio-scores*)))
317     (cond ((null pd) (setf
318                       (gethash systems *portfolio-scores*)
319                       (make-portfolio-data
320                        :metric-scores metric-vals
321                        :oec-score oec-val
322                        :n 1.0))
323            (setf (gethash systems *portfolio-isodata*)
324                  (make-portfolio-isodata
325                   :p-systems systems
326                   :p-tasks (flatten ;;fix w/task hierarchy
327                             (get-decided-tasks
328                              (symbol-value *mtsk*))))))
329           (t (let ((old-val (portfolio-data-oec-score pd))
330                    (n (portfolio-data-n pd))
331                    (old-metric-vals
332                     (portfolio-data-metric-scores pd)))
333                (setf (portfolio-data-metric-scores pd)
334                      (loop for metric-score in metric-vals
335                         for old-metric-score in
336                         old-metric-vals
337                         collect
338                           (add-to-average
339                            metric-score
340                            old-metric-score n)))
341                (setf (portfolio-data-oec-score pd)
342                      (add-to-average oec-val old-val n))
343                (incf (portfolio-data-n pd)))))))
344
345 (defun update-running-average-for-all-metrics
346     (oec-function metric-vals)
347   (let ((oec-val (apply oec-function metric-vals))
348         (systems (sort (delete-duplicates
349                         (decisions-to-portfolio
350                          (symbol-value *mtsk*)))
351                        #'string<)))
352     (update-portfolio-score-for-all-metrics
353      systems oec-val metric-vals)))
354
355
356 (defun output-portfolio-decisions-and-metric-scores ()
357 ;;(format t "~%~{~a, ~}~{~a, ~}" (list-current-decisions)
358   (update-running-average-for-all-metrics *oec-function*
359                           (loop for mtric in *dep-metrics*
360                              do (setf
361                                  *current-metric*
362                                  (symbol-value mtric))
363                              collect
364                                (funcall
365                                 (symbol-function *mtsk*)
```

```
366                                        *current-metric*))))
367
368 (defun make-wasteful-portfolio-decisions
369     (made-decisions not-made-decisions)
370   (make-wasteful-decisions-with-output-function
371    made-decisions not-made-decisions
372    #'output-portfolio-decisions-and-metric-scores))
373
374
375 (defun make-wasteful-decisions-with-output-function
376     (made-decisions not-made-decisions out-function)
377   (cond ((null not-made-decisions)
378          (make-listed-decisions made-decisions)
379          (funcall out-function))
380         ((equal (cadar not-made-decisions) 1)
381          (make-wasteful-decisions-with-output-function
382           (cons (car not-made-decisions) made-decisions)
383           (cdr not-made-decisions) out-function))
384         (t (make-wasteful-decisions-with-output-function
385             (cons (car not-made-decisions) made-decisions)
386             (cdr not-made-decisions) out-function)
387            (make-wasteful-decisions-with-output-function
388             made-decisions
389             (cons (list (caar not-made-decisions)
390                         (- (cadar not-made-decisions) 1))
391                   (cdr not-made-decisions))
392             out-function))))
393
394 (defparameter *limit* 2)
395 (defun
396     make-wasteful-decisions-wof-and-limit
397     (made-decisions not-made-decisions out-function)
398   (cond
399     ((equal *limit* 0) ())
400     ((null not-made-decisions)
401      (make-listed-decisions made-decisions)
402      (funcall out-function)
403      (setf *limit* (- *limit* 1)))
404     ((equal (cadar not-made-decisions) 1)
405      (make-wasteful-decisions-wof-and-limit
406       (cons (car not-made-decisions) made-decisions)
407       (cdr not-made-decisions) out-function))
408     (t (make-wasteful-decisions-wof-and-limit
409         (cons (car not-made-decisions) made-decisions)
410         (cdr not-made-decisions) out-function)
411        (make-wasteful-decisions-wof-and-limit
412         made-decisions
413         (cons (list (caar not-made-decisions)
414                     (- (cadar not-made-decisions) 1))
415               (cdr not-made-decisions)) out-function))))
416
417 (defun get-list-of-leaves ()
418   (loop for (tsk n) in (list-decisions)
419                 when (leaf? tsk)
```

```
420                 collect tsk))
421
422 (defun get-list-of-non-leaves ()
423   (loop for (tsk n) in (list-decisions)
424                     when (not (leaf? tsk))
425                     collect tsk))
426
427
428 (defun list-of-x-to-bits (current-list full-list)
429   (loop for item in full-list
430         collect (if (member item current-list) 1 0)))
431
432 (defun add-prefix-and-fix-for-jmp (prefix item)
433   (concatenate 'string prefix
434                (string-capitalize
435                 (substitute #\  #\- (symbol-name item)))))
436
437 (defun add-prefixs (prefix lst)
438   (loop for item in lst
439         collect (add-prefix-and-fix-for-jmp prefix item)))
440
441 (defun output-for-jmp-portfolio-data-for-all-metrics ()
442   ;Horrible let* due to code printing width...
443   (let* ((fmt-st "~&~{~a,~}~{~a,~}~{~a,~}OEC Average,")
444          (fmt-sr "Number of Alternatives,")
445          (fmt-tr "Number of Systems")
446          (fmt-str (concatenate 'string
447                                fmt-st fmt-sr fmt-tr)))
448     (format
449      t fmt-str
450      (add-prefixs "Task " *sorted-task-symbols*)
451      (add-prefixs "System " *sorted-system-symbols*)
452      (add-prefixs "Metric Average " *dep-metrics*)))
453   (loop for k being the hash-keys in *portfolio-isodata*
454      using (hash-value v)
455      do (let ((pd (gethash k *portfolio-scores*)))
456           (format t "~&~{~a,~}~{~a,~}~{~a,~}~a,~a,~a,"
457                   (list-of-x-to-bits
458                    (portfolio-isodata-p-tasks v)
459                    *sorted-task-symbols*)
460                   (list-of-x-to-bits
461                    (portfolio-isodata-p-systems v)
462                    *sorted-system-symbols*)
463                   (portfolio-data-metric-scores pd)
464                   (portfolio-data-oec-score pd)
465                   (portfolio-data-n pd)
466                   (apply #'+
467                          (list-of-x-to-bits
468                           (portfolio-isodata-p-systems v)
469                           *sorted-system-symbols* ))))))
470
471     ;collect (list v (gethash v *portfolio-scores*))))
472
473 (defun output-decisions-and-metric-scores-2 ()
```

```
474   (format t "~%~{~a, ~}~{~a, ~}" (list-current-decisions)
475           (loop for mtric in *dep-metrics*
476              do (setf *current-metric*
477                       (symbol-value mtric))
478              collect (funcall (symbol-function *mtsk*)
479                               *current-metric*))))
480
481 (defun dont-output-just-run ()
482   (loop for mtric in *dep-metrics*
483      do (setf *current-metric* (symbol-value mtric))
484      collect (funcall (symbol-function *mtsk*)
485                       *current-metric*)))
486
487 (defun dont-run-just-make-decisions ()
488   '())
489
490 (defun symbol-to-task (tsklist)
491   (loop for task-symbol in tsklist
492        collect `,(symbol-value task-symbol)))
493
494 (defun get-selected-system-from-task (tsk)
495   (symbol-value
496    (car
497     (funcall (current-decision tsk) (subtasks tsk)))))
498
499 (defun get-children (tsk)
500   (if (leaf? tsk)
501       '()
502       (funcall (current-decision tsk) (subtasks tsk))))
503
504 (defun get-next-level-decided-children (tasklist)
505   (loop for tsk in tasklist
506      append (symbol-to-task (get-children tsk))))
507
508 (defun get-next-level-decided-children2 (tasklist)
509   (loop for (tsk n) in tasklist
510      append (symbol-to-task (get-children tsk))))
511
512 (defun get-all-levels-decided-children (tsklist)
513   (if tsklist
514       (append tsklist (get-all-levels-decided-children
515                            (get-next-level-decided-children tsklist)))
516       '()))
517
518 ;(defun make-decision-list (decisions)
519 ;  (loop for (task-symbol n) in decisions
520 ;        collect (list task-symbol n)))
521
522 (defun make-decision-list (tsklist)
523   (loop for tsk in tsklist
524        collect (list tsk (length (subtasks tsk)))))
525
526 (defun p-d (m-d n-m-d outfunc) ;Permutate Decisions
527   (cond ((null n-m-d)
```

```
528           (make-listed-decisions m-d)
529           (let ((next-level-decided-children
530                  (get-next-level-decided-children2 m-d)))
531             (if (null next-level-decided-children)
532                 (funcall outfunc)
533                 (p-d '() (make-decision-list next-level-decided-children)
534                      outfunc))))
535         (t (p-d (cons (car n-m-d) m-d)
536                 (cdr n-m-d) outfunc)
537            (unless (equal (cadar n-m-d) 1)
538              (p-d m-d
539                   (cons (list (caar n-m-d)
540                               (- (cadar n-m-d) 1))
541                         (cdr n-m-d))
542                   outfunc)))))
543
544 (defun make-decisions-with-output-function (outfunc)
545   (p-d '()
546        (make-decision-list
547         (get-next-level-decided-children
548           (symbol-to-task (list *mtsk*))))
549        outfunc))
550
551 (defun p-d-limit (m-d n-m-d outfunc) ;Permutate Decisions
552   (cond ((equal *limit* 0) ())
553         ((null n-m-d)
554          (make-listed-decisions m-d)
555          (decf *limit*)
556          (let ((next-level-decided-children
557                 (get-next-level-decided-children2 m-d)))
558            (if (null next-level-decided-children)
559                (progn (funcall outfunc))
560                (p-d-limit
561                 '()
562                 (make-decision-list next-level-decided-children)
563                 outfunc))))
564         ((equal (cadar n-m-d) 1)
565          (p-d-limit (cons (car n-m-d) m-d)
566                     (cdr n-m-d) outfunc))
567         (t (p-d-limit (cons (car n-m-d) m-d)
568                       (cdr n-m-d) outfunc)
569            (p-d-limit m-d
570                       (cons (list (caar n-m-d)
571                                   (- (cadar n-m-d) 1))
572                             (cdr n-m-d))
573                       outfunc))))
574
575 (defun make-decisions-with-output-function-and-limit (outfunc)
576   (p-d-limit '()
577              (make-decision-list
578               (get-next-level-decided-children
579                 (symbol-to-task (list *mtsk*))))
580              outfunc))
581
```

```
582
583 (defun list-number-of-decisions (taskks)
584   "Creates a list of the number of decisions for each task.
585    It is sorted in the same order as the taskks arguement."
586   (loop for taskk in taskks
587         collect (list-length (subtasks taskk))))
588
589 (defun list-decisions ()
590   "Creates a list of all tasks with more than one decision."
591   (loop for tsk in *tasks*
592         for decision in (list-number-of-decisions *tasks*)
593         when (> decision 1)
594       collect (list tsk decision)))
595
596 (defun list-all-decisions ()
597   "Creates a list of all tasks with decisions."
598   (loop for tsk in *tasks*
599         for decision in (list-number-of-decisions *tasks*)
600         collect (list tsk decision)))
601
602 (defun list-current-decisions ()
603   "Returns a list of the currently made decisions as a set of
604    decision positions."
605   (loop for tsk in *tasks*
606       collect
607         (position (funcall (current-decision tsk) (subtasks tsk))
608                   (subtasks tsk))))
609
610 (defun list-current-decisions-2 ()
611   "Returns a list of the current decisions with either the subtasks
612    or the system that is chosen"
613   (loop for tsk in *tasks*
614       collect
615         (if (leaf? tsk)
616             (symbol-value (car (funcall (current-decision tsk) (subtasks
                tsk))))
617             (funcall (current-decision tsk) (subtasks tsk)))))
618
619 (defun list-operational-decisions ()
620   "Returns a list of only the operational decisions"
621   (loop for tsk in *tasks*
622         for decision in (list-number-of-decisions *tasks*)
623         when (and (> decision 1) (not (leaf? tsk)))
624       collect (list tsk decision)))
625
626 (defun count-number-of-alternatives (capabil)
627   "Count the number of alternatives given the capability."
628   (count-from-task-tree (make-task-tree (main-task capabil))))
629
630 (defun make-task-tree (taskk)
631   (loop for taskkks in (subtasks taskk)
632       collect (loop for tas in taskkks
633                     collect (if (eql 'symbol (type-of (symbol-value tas)))
634                                 tas
```

```
635                                  (make-task-tree (symbol-value tas))))))
636
637 ;for style warning for mutually recursive functions
638 (declaim (ftype function count-from-task-tree))
639 (defun count-from-task-tree-helper (task-decision-tree)
640       (apply #'* (mapcar #'count-from-task-tree task-decision-tree)))
641
642 (defun count-from-task-tree (task-decision-tree)
643   (if (listp task-decision-tree)
644       (apply #'+ (mapcar #'count-from-task-tree-helper
                         task-decision-tree))
645       1))
646
647 (defun count-alternatives (tsk)
648   "Count the number of alternatives given any task."
649   (count-from-task-tree (make-task-tree tsk)))
650
651 (defun output-decisions-and-metric-scores-old (decisions)
652 ;  (setf *decisions* decisions)
653 ;  (format t "~%~{~a, ~}~{~a, ~}" decisions
654   (loop for mtric in *dep-metrics*
655      do (setf *current-metric* (symbol-value mtric))
656      collect (funcall (symbol-function *mtsk*) *current-metric*)))
657 ;)
658
659 (defun output-decisions-and-metric-scores ();(decisions)
660 ;;(format t "~%~{~a, ~}~{~a, ~}" (list-current-decisions)
661   (update-top-n *oec-function*
662                 (loop for mtric in *dep-metrics*
663                    do (setf *current-metric* (symbol-value mtric))
664                    collect (funcall (symbol-function *mtsk*)
                                    *current-metric*)))
665 )
666 ;;)
667
668
669
670 (defun remove-dashes-from-symbol-to-string (sym)
671   "Removes dashes in a symbol and converts it to a string.
672    Used to interoperate with software such as JMP."
673   (substitute-if #\_ (complement #'alphanumericp)
674                  (prin1-to-string sym)))
675
676 (defun flatten (lst)
677  "Removes nesting in a list."
678  (cond ((null lst) '())
679        ((atom lst) (list lst))
680        (t (nconc (flatten (car lst))
681                  (flatten (cdr lst))))))
682
683 (defun sum (&rest lst)
684   (the single-float (apply #'+ lst)))
685
686 (defun product (&rest lst)
```

```
687   (the single-float (apply #'* lst)))
688 (defun ident (val)
689   "The identity transformation function. Does nothing to the input."
690   val)
691 ;for now, to interface with the rest of the world, going
692 ;to use sbcl specific stuff. If I don't use sbcl,
693 ;it should fail
694 #+sbcl
695 (defun main-prog ()
696   (let ((load-file-name (second sb-ext:*posix-argv*)))
697     (if load-file-name
698         (load load-file-name) ;; error handling goes here.
699         '())))
700
701 (main-prog)
```

APPENDIX D

SEAD MODEL INPUT

D.1 Source Code of the SEAD Model

Listing D.1: sead_seam.lisp

```
1  (load "raam.lisp")
2  (in-package :asdl.raam)
3  ;;set *allowed-systems* to nil for all systems
4  (declaim (optimize (speed 3) (safety 0) (debug 0)))
5  (defparameter *allowed-systems* nil)
6
7  ;all
8  ;(defparameter *allowed-systems* '(AH-64 CENTRAL-C2 CVN DDG E-2 EA-6B
       F/A-18 INTEL-SATELLITE M1 SOF X-47B))
9
10 ;less
11
12 ;row 522
13 ;(defparameter *allowed-systems* '(AH-64 CENTRAL-C2 CVN E-2 F/A-18 SOF
       X-47B))
14
15 ;row 751
16 ;(defparameter *allowed-systems* '(AH-64 CVN E-2 F/A-18 SOF))
17
18 ;row 1219
19 ;(defparameter *allowed-systems* '(CENTRAL-C2 E-2 F/A-18 SOF))
20
21 ;row 1223
22 ;(defparameter *allowed-systems* '(CENTRAL-C2 F/A-18 SOF))
23
24 ;row 1224
25 ;(defparameter *allowed-systems* '(AH-64 CENTRAL-C2 F/A-18 SOF))
26
27 ;row 1231
28 ;(defparameter *allowed-systems* '(CENTRAL-C2 DDG F/A-18 SOF))
29
30 ;(defparameter *allowed-systems* '(CENTRAL-C2 DDG EA-6B SOF))
31
32 ;;==TASKS==
33 (task conduct-sead
34     "Conduct SEAD to prepare battlefield for follow on attacks"
35     (Detect Identify Correlate-and-Track Target-Assignment
          Weapon-Control))
36
37 (task Detect "Detect the enemy positions"
38      (Reconcile-Target-Priorities
```

```
39          Determine-Sensor-Availability
40          Task-Sensor
41          Wide-Area-Search
42          Fuse-Sensor-Data
43          Pass-Warning-and-Location-Data)
44  ;       (Allocate-sensors
45  ;        Gather-Target-Attributes
46  ;        Geolocate-Targets)
47  )
48
49  (task Reconcile-Target-Priorities "Determine target priorities" ())
50  (task Determine-Sensor-Availability "Determine which sensors are
        available" ())
51  (task Task-Sensor "Task each sensor" ())
52  (task Wide-Area-Search
53        "Conduct a wide area search" ())
54  (task Fuse-Sensor-Data "Fuse sensor data" ())
55  (task Pass-Warning-and-Location-Data "Pass warning and location data
56        to identification platforms" ())
57
58  ;(task Allocate-sensors "Allocate sensors between systems" ())
59  ;(task Gather-Target-Attributes "Combine sensor information into target
        attributes" ())
60  ;(task Geolocate-Targets "Determine the geo-location of the targets" ())
61
62  (task Identify "Identify the types of targets" ())
63
64  (task Correlate-and-Track
65        "Correlate and Track the targets"
66        (Manage-Target-Movement-Data
67         Discriminate-Decoys
68         Track-Until-Stopped))
69
70  (task Manage-Target-Movement-Data "Manage target movement data" ())
71  (task Discriminate-Decoys "Discriminate launch and support systems from
        decoys" ())
72  (task Track-Until-Stopped "Track enemy systems until stopped" ())
73
74  (task Target-Assignment
75        "Assign system to each target, specifying weapon"
76        (Update-Target-List
77         Assess-Engagement-Capability
78         Assign-Weapon-and-Platform))
79
80  (task Update-Target-List "Update the target list" ())
81  (task Assess-Engagement-Capability "Assess the different engagement
        capabilities of the weapon-system pairings" ())
82  (task Assign-Weapon-and-Platform "Decide the weapon-system pairings" ())
83
84  (task Weapon-Control "Weapon Control"
85        (Engage-to-Destroy
86         Engage-to-Disrupt
87         Battle-Damage-Assessment
88         Remove-from-Target-List))
```

```
89
90  (task Engage-to-Destroy "Engage to destroy targets" ())
91  (task Engage-to-Disrupt "Engage to disrupt targets" ())
92  (task Battle-Damage-Assessment "Conduct battle damage assessments" ())
93  (task Remove-from-Target-List "Remove destroyed or disrupted targets
        from the target list" ())
94
95  ;;==CAPABILITY==
96  (capability complete-tasks "Complete the SEAD tasks"
97              conduct-sead (P-success Complexity Time-to-completion
                    Maintainability)
98              (Cost Risk))
99
100 ;;==METRICS==
101 ;CVN
102 (metric Reconcile-Target-Priorities CVN
103   (P-success 0.98) (Complexity 5) (Time-to-completion 5)
        (Maintainability 1))
104 (metric Determine-Sensor-Availability CVN
105   (P-success 0.99) (Complexity 5) (Time-to-completion 12)
        (Maintainability 4))
106 (metric Task-Sensor CVN
107   (P-success 0.98) (Complexity 1) (Time-to-completion 22)
        (Maintainability 4))
108 (metric Fuse-Sensor-Data CVN
109   (P-success 0.99) (Complexity 3) (Time-to-completion 16)
        (Maintainability 3))
110 (metric Pass-Warning-and-Location-Data CVN
111   (P-success 0.97) (Complexity 3) (Time-to-completion 13)
        (Maintainability 2))
112 (metric Manage-Target-Movement-Data CVN
113   (P-success 0.98) (Complexity 4) (Time-to-completion 15)
        (Maintainability 3))
114 (metric Update-Target-List CVN
115   (P-success 0.99) (Complexity 1) (Time-to-completion 21)
        (Maintainability 8))
116 (metric Assess-Engagement-Capability CVN
117   (P-success 0.99) (Complexity 2) (Time-to-completion 16)
        (Maintainability 5))
118 (metric Assign-Weapon-and-Platform CVN
119   (P-success 0.995) (Complexity 5) (Time-to-completion 18)
        (Maintainability 4))
120 (metric Remove-from-Target-List CVN
121   (P-success 0.96) (Complexity 2) (Time-to-completion 14)
        (Maintainability 8))
122
123 ;Central-C2
124 (metric Reconcile-Target-Priorities Central-C2
125   (P-success 0.98) (Complexity 4) (Time-to-completion 9)
        (Maintainability 10))
126 (metric Determine-Sensor-Availability Central-C2
127   (P-success 0.95) (Complexity 5) (Time-to-completion 7)
        (Maintainability 8))
128 (metric Task-Sensor Central-C2
```

```
129    (P-success 0.99) (Complexity 2) (Time-to-completion 15)
           (Maintainability 2))
130  (metric Fuse-Sensor-Data Central-C2
131    (P-success 0.97) (Complexity 2) (Time-to-completion 12)
           (Maintainability 4))
132  (metric Pass-Warning-and-Location-Data Central-C2
133    (P-success 0.99) (Complexity 1) (Time-to-completion 12)
           (Maintainability 6))
134  (metric Manage-Target-Movement-Data Central-C2
135    (P-success 0.95) (Complexity 5) (Time-to-completion 8)
           (Maintainability 6))
136  (metric Update-Target-List Central-C2
137    (P-success 0.98) (Complexity 1) (Time-to-completion 19)
           (Maintainability 4))
138  (metric Assess-Engagement-Capability Central-C2
139    (P-success 0.98) (Complexity 1) (Time-to-completion 22)
           (Maintainability 3))
140  (metric Assign-Weapon-and-Platform Central-C2
141    (P-success 0.99) (Complexity 3) (Time-to-completion 12)
           (Maintainability 3))
142  (metric Remove-from-Target-List Central-C2
143    (P-success 0.95) (Complexity 3) (Time-to-completion 11)
           (Maintainability 9))
144
145  ;;Intel-Satellite
146  (metric Wide-Area-Search Intel-Satellite
147    (P-success 0.87) (Complexity 5) (Time-to-completion 7)
           (Maintainability 2))
148  (metric Discriminate-Decoys Intel-Satellite
149    (P-success 0.80) (Complexity 5) (Time-to-completion 15)
           (Maintainability 6))
150  (metric Track-Until-Stopped Intel-Satellite
151    (P-success 0.95) (Complexity 2) (Time-to-completion 15)
           (Maintainability 4))
152  (metric Battle-Damage-Assessment Intel-Satellite
153    (P-success 0.85) (Complexity 2) (Time-to-completion 19)
           (Maintainability 8))
154  (metric Identify Intel-Satellite
155    (P-success 0.85) (Complexity 1) (Time-to-completion 18)
           (Maintainability 10))
156
157  ;;X-47B
158  (metric Wide-Area-Search X-47B
159    (P-success 0.90) (Complexity 1) (Time-to-completion 12)
           (Maintainability 10))
160  (metric Identify X-47B
161    (P-success 0.85) (Complexity 4) (Time-to-completion 21)
           (Maintainability 7))
162  (metric Discriminate-Decoys X-47B
163    (P-success 0.90) (Complexity 1) (Time-to-completion 15)
           (Maintainability 7))
164  (metric Track-Until-Stopped X-47B
165    (P-success 0.95) (Complexity 3) (Time-to-completion 9)
           (Maintainability 1))
```

```
166 (metric Engage-to-Destroy X-47B
167   (P-success 0.80) (Complexity 2) (Time-to-completion 12)
        (Maintainability 2))
168 (metric Engage-to-Disrupt X-47B
169   (P-success 0.90) (Complexity 5) (Time-to-completion 19)
        (Maintainability 8))
170 (metric Battle-Damage-Assessment X-47B
171   (P-success 0.99) (Complexity 1) (Time-to-completion 20)
        (Maintainability 4))
172
173 ;F/A-18
174 (metric Wide-Area-Search F/A-18
175   (P-success 0.90) (Complexity 4) (Time-to-completion 8)
        (Maintainability 6))
176 (metric Discriminate-Decoys F/A-18
177   (P-success 0.86) (Complexity 5) (Time-to-completion 17)
        (Maintainability 5))
178 (metric Engage-to-Destroy F/A-18
179   (P-success 0.97) (Complexity 2) (Time-to-completion 12)
        (Maintainability 10))
180 (metric Battle-Damage-Assessment F/A-18
181   (P-success 0.76) (Complexity 5) (Time-to-completion 12)
        (Maintainability 4))
182 (metric Identify F/A-18
183   (P-success 0.9) (Complexity 4) (Time-to-completion 7)
        (Maintainability 2))
184
185 ;AH-64
186 (metric Wide-Area-Search AH-64
187   (P-success 0.80) (Complexity 2) (Time-to-completion 4)
        (Maintainability 5))
188 (metric Discriminate-Decoys AH-64
189   (P-success 0.96) (Complexity 2) (Time-to-completion 10)
        (Maintainability 10))
190 (metric Identify AH-64
191   (P-success 0.79) (Complexity 5) (Time-to-completion 15)
        (Maintainability 3))
192 (metric Engage-to-Destroy AH-64
193   (P-success 0.87) (Complexity 5) (Time-to-completion 5)
        (Maintainability 5))
194 (metric Battle-Damage-Assessment AH-64
195   (P-success 0.98) (Complexity 1) (Time-to-completion 16)
        (Maintainability 1))
196
197 ;EA-6B
198 (metric Wide-Area-Search EA-6B
199   (P-success 0.90) (Complexity 4) (Time-to-completion 3)
        (Maintainability 10))
200 (metric Identify EA-6B
201   (P-success 0.999) (Complexity 1) (Time-to-completion 14)
        (Maintainability 4))
202 (metric Discriminate-Decoys EA-6B
203   (P-success 0.80) (Complexity 5) (Time-to-completion 8)
        (Maintainability 6))
```

```
204 (metric Engage-to-Destroy EA-6B
205   (P-success 0.94) (Complexity 4) (Time-to-completion 3)
        (Maintainability 5))
206 (metric Engage-to-Disrupt EA-6B
207   (P-success 0.95) (Complexity 2) (Time-to-completion 21)
        (Maintainability 3))
208 (metric Battle-Damage-Assessment EA-6B
209   (P-success 0.98) (Complexity 1) (Time-to-completion 19)
        (Maintainability 2))
210
211 ;E-2
212 (metric Fuse-Sensor-Data E-2
213   (P-success 0.98) (Complexity 1) (Time-to-completion 8)
        (Maintainability 10))
214 (metric Pass-Warning-and-Location-Data E-2
215   (P-success 0.99) (Complexity 2) (Time-to-completion 17)
        (Maintainability 3))
216 (metric Manage-Target-Movement-Data E-2
217   (P-success 0.99) (Complexity 1) (Time-to-completion 7)
        (Maintainability 2))
218 (metric Track-Until-Stopped E-2
219   (P-success 0.98) (Complexity 2) (Time-to-completion 5)
        (Maintainability 10))
220 (metric Update-Target-List E-2
221   (P-success 0.97) (Complexity 5) (Time-to-completion 17)
        (Maintainability 4))
222
223 ;M1
224 (metric Discriminate-Decoys M1
225   (P-success 0.70) (Complexity 2) (Time-to-completion 7)
        (Maintainability 4))
226 (metric Engage-to-Destroy M1
227   (P-success 0.80) (Complexity 3) (Time-to-completion 14)
        (Maintainability 7))
228 (metric Battle-Damage-Assessment M1
229   (P-success 0.99) (Complexity 4) (Time-to-completion 19)
        (Maintainability 3))
230 (metric Identify M1
231   (P-success 0.999) (Complexity 1) (Time-to-completion 21)
        (Maintainability 6))
232
233 ;DDG
234 (metric Engage-to-Destroy DDG
235   (P-success 0.99) (Complexity 1) (Time-to-completion 12)
        (Maintainability 3))
236
237 ;SOF
238 (metric Identify SOF
239   (P-success 0.98) (Complexity 1) (Time-to-completion 4)
        (Maintainability 8))
240 (metric Discriminate-Decoys SOF
241   (P-success 0.76) (Complexity 4) (Time-to-completion 11)
        (Maintainability 4))
242 (metric Track-Until-Stopped SOF
```

```
243   (P-success 0.98) (Complexity 3) (Time-to-completion 15)
          (Maintainability 8))
244 (metric Engage-to-Destroy SOF
245   (P-success 0.99) (Complexity 5) (Time-to-completion 10)
          (Maintainability 5))
246 (metric Engage-to-Disrupt SOF
247   (P-success 0.99) (Complexity 5) (Time-to-completion 19)
          (Maintainability 6))
248 (metric Battle-Damage-Assessment SOF
249   (P-success 0.99) (Complexity 2) (Time-to-completion 17)
          (Maintainability 2))
250
251 ;;==SYSTEMS==
252 (system CVN "Nuclear powered aircraft carrier "
253   (Cost 10000.0) (Risk 0.1))
254 (system Central-C2 "Local command and control"
255   (Cost 15.0) (Risk 0.2))
256 (system Intel-Satellite "Intelligence satellites"
257   (Cost 3000.0) (Risk 0.4))
258 (system X-47B "X-47B Unmanned Aerial Vehicle"
259   (Cost 80.0) (Risk 0.8))
260 (system F/A-18 "F/A-18 fighter"
261   (Cost 68.0) (Risk 0.05))
262 (system AH-64 "AH-64 Attack Helicopter"
263   (Cost 20.0) (Risk 0.05))
264 (system EA-6B "EA-6B Electronic Warfare aircraft"
265   (Cost 70.0) (Risk 0.05))
266 (system E-2 "Airborne Early Warning aircraft"
267   (Cost 100.0) (Risk 0.05))
268 (system M1 "M1 Abrams tank"
269   (Cost 0.25) (Risk 0.05))
270 (system DDG "DDG-1000 Zumwalt-class destroyer"
271   (Cost 2000.0) (Risk 0.6))
272 (system SOF "Special Operations Forces"
273   (Cost  20.0) (Risk 0.2))
274
275 (defun sum (&rest lst)
276   (the single-float (apply #'+ lst)))
277
278 (defun product (&rest lst)
279   (the single-float (apply #'* lst)))
280
281
282 ;P-success
283 (compute conduct-sead P-success product ident)
284 (compute Detect P-success product ident)
285 (compute Correlate-and-Track P-success product ident)
286 (compute Target-Assignment P-success product ident)
287 (compute Weapon-Control P-success product ident)
288
289 ;Complexity
290 (compute conduct-sead Complexity product ident)
291 (compute Detect Complexity sum ident)
292 (compute Correlate-and-Track Complexity max ident)
```

```
(compute Target-Assignment Complexity min ident)
(compute Weapon-Control Complexity sum ident)

;Time-to-completion
(compute conduct-sead Time-to-completion sum ident)
(compute Detect Time-to-completion sum ident)
(compute Correlate-and-Track Time-to-completion sum ident)
(compute Target-Assignment Time-to-completion sum ident)
(compute Weapon-Control Time-to-completion sum ident)

;Maintainability
(compute conduct-sead Maintainability min ident)
(compute Detect Maintainability min ident)
(compute Correlate-and-Track Maintainability min ident)
(compute Target-Assignment Maintainability min ident)
(compute Weapon-Control Maintainability min ident)

(add-system-selectors-to-leafs)

(portfolio-compute Cost + ident)
(portfolio-compute Risk * ident)

;(defparameter *dep-metrics* '(P-success Complexity Time-to-completion
    Maintainability))
;(defparameter *mtsk* 'conduct-sead)

;(pprint (list-number-of-decisions *tasks*))

;(format t "~{~a, ~}~{~a, ~}" *tasks* *dep-metrics*)
;(pprint (list-number-of-decisions *tasks*))
;(pprint (apply #'*
;  '(2 7 3 7 1 2 2 3 1 4 7 3 1 7 3 3 5 2 2 2 1 1)))
;(pprint (apply #'*
;'(2 7 1 1 1 2 2 1 1 4 1 1 1 7 3 3 5 2 1 2 1 1)))

;                    '(2 7 3 7 1 2 2 3 1 4 7 3 1 7 3 3 5 2 2 2 1 1)
;                    '(2 7 1 1 1 2 2 1 1 4 1 1 1 7 3 3 5 2 1 2 1 1)
;(make-decisions '() '(2 7 1 1 1 2 2 1 1 4 1 1 1 7 3 3 5 2 1 2 1 1))

;**********************
;(print (list-decisions))
;(pprint (count-from-task-tree (make-task-tree conduct-sead)))
;(make-wasteful-portfolio-decisions '() (list-decisions))

;(print (loop for k being the hash-keys in *portfolio-systems*
;using (hash-value v)
;            collect (list v (gethash v *portfolio-scores*))))
;**********************

;041711
(defparameter *sorted-system-symbols* '(AH-64 CENTRAL-C2 CVN DDG E-2
    EA-6B F/A-18 INTEL-SATELLITE M1 SOF X-47B))
;I removed tasks that don't have any subtasks... manually, very ugly.
(defparameter *sorted-task-symbols*
```

```
345     '(ASSESS-ENGAGEMENT-CAPABILITY ASSIGN-WEAPON-AND-PLATFORM
346       BATTLE-DAMAGE-ASSESSMENT
347       DETERMINE-SENSOR-AVAILABILITY DISCRIMINATE-DECOYS
348       ENGAGE-TO-DESTROY ENGAGE-TO-DISRUPT
349       FUSE-SENSOR-DATA IDENTIFY
350       MANAGE-TARGET-MOVEMENT-DATA
351       PASS-WARNING-AND-LOCATION-DATA
352       RECONCILE-TARGET-PRIORITIES
353       REMOVE-FROM-TARGET-LIST
354       TASK-SENSOR TRACK-UNTIL-STOPPED
355       UPDATE-TARGET-LIST
356       WIDE-AREA-SEARCH) )
357   ;(format t "~&~S" *sorted-system-symbols*)
358   ;(format t "~&~S" *sorted-task-symbols*)
359
360   (defun system-portfolio-view-out ()
361     ;(format t "~&~A" (mapcar #'symbol-value (flatten (get-decided-leafs
          (symbol-value *mtsk*)))))
362     (if (not (equal (length *allowed-systems*) (length
          (get-decided-systems))))
363         '()
364     (let ((systems-to-tasks
365            (mapcar #'(lambda (x)
366                        (1+ (position
367                             (symbol-value x)
368                             *sorted-system-symbols*))
369                        (add-prefix-and-fix-for-jmp
370                         ""
371                         (symbol-value x)))
372                    (sort (flatten (get-decided-leafs
373                                    (symbol-value *mtsk*)))
374                          #'string<))))
375       ;(format t "~&~a" (length *sorted-task-symbols*))
376       (format t "~&~{~a,~}~{~a,~}~{~a,~}~{~a,~}"
377               systems-to-tasks
378               (list-of-x-to-bits (get-decided-systems);*allowed-systems*
379                                  *sorted-system-symbols*)
380               (loop for mtric in *dep-metrics*
381                  do (setf *current-metric* (symbol-value mtric))
382                  collect (funcall (symbol-function *mtsk*)
383                                   *current-metric*))
384               (loop for mtric in *indep-metrics*
385                     collect (funcall mtric))))))
386
387   ;; (defun update-portfolio-score-for-all-metrics
388   ;           (systems oec-val metric-vals)
389   ;;   ;(print (mapcar #'type-of (flatten
390   ;            (get-decided-tasks (symbol-value *mtsk*)))))
391   ;;   (let ((pd (gethash systems *portfolio-scores*)))
392   ;;     (cond ((null pd) (setf (gethash systems *portfolio-scores*)
393   ;;                            (make-portfolio-data :metric-scores
        metric-vals
394   ;;                                                 :oec-score oec-val
395   ;;                                                 :n 1.0))
```

```
396 ;;          (setf (gethash systems *portfolio-isodata*)
397 ;;                (make-portfolio-isodata :p-systems systems
398 ;;                                        :p-tasks (flatten ;;horrible
        idea, fix when task hierarchy
399 ;;                                                 (get-decided-tasks
        ;... changes
400 ;;                                                  (symbol-value
        *mtsk*))))))
401 ;;        (t (let ((old-val (portfolio-data-oec-score pd))
402 ;;                 (n (portfolio-data-n pd))
403 ;;                 (old-metric-vals (portfolio-data-metric-scores pd)))
404 ;;             (setf (portfolio-data-metric-scores pd)
405 ;;                   (loop for metric-score in metric-vals
406 ;;                         for old-metric-score in old-metric-vals
407 ;;                         collect (add-to-average metric-score
        old-metric-score n)))
408 ;;             (setf (portfolio-data-oec-score pd) (add-to-average
        oec-val old-val n))
409 ;;             (incf (portfolio-data-n pd)))))
410 ;; ))
411
412
413 ;(print (count-number-of-alternatives complete-tasks))
414
415 ;; (format t "~&~{~a,~}~{~a,~}~{~a,~}~{~a,~}"
416 ;;         (add-prefixs "Task " *sorted-task-symbols*)
417 ;;         (add-prefixs "System " *sorted-system-symbols*)
418 ;;         (add-prefixs "Metric " *dep-metrics*)
419 ;;         (add-prefixs "Metric " *indep-metrics*))
420 ;; (make-wasteful-decisions-with-output-function '() (list-decisions)
        #'system-portfolio-view-out)
421
422 ;(print (cost))
423
424 ;(print (list-decisions))
425
426 ;(output-for-jmp-portfolio-data-for-all-metrics)
427 ;041711
428
429
430 ;(print (type-of (symbol-value Reconcile-Target-Priorities-central-c2)))
431 ;(print (eql 'task (type-of (symbol-value
        Reconcile-Target-Priorities-central-c2))))
432 ;(print (eql 'task (type-of central-c2)))
433 ;(print (type-of (type-of Reconcile-Target-Priorities)))
434
435 ;(print (make-task-tree conduct-sead))
436
437 (time (make-wasteful-decisions-with-output-function '()
        (list-decisions) #'dont-run-just-make-decisions))
438 (time (make-decisions-with-output-function
        #'dont-run-just-make-decisions))
439
440 (terpri)
```

APPENDIX E

SEAD MODEL PORTFOLIO VIEW OUTPUT

The following data is the result of the portfolio analysis of the SEAD example.

Table 22: SEAD Portfolio Data

S1	S2	S3	S4	S5	S6	S7	S8	S9	S10	S11	M1	M2	M3	M4	OEC	#A
0	0	1	0	0	0	0	1	0	0	1	0.335	2345.8	258.45	1.000	2.11	31
0	1	1	0	0	0	0	1	0	0	1	0.319	2393.9	245.95	1.258	2.16	31682
0	0	1	0	0	0	1	1	0	0	1	0.362	3327.7	251.72	1.000	2.13	261
0	1	1	0	0	0	1	1	0	0	1	0.345	3316.0	239.23	1.298	2.20	266742
1	0	1	0	0	0	0	1	0	0	1	0.353	3424.8	248.16	1.000	2.10	261
1	1	1	0	0	0	0	1	0	0	1	0.337	3535.7	235.66	1.195	2.12	266742
0	0	1	0	0	1	0	1	0	0	1	0.400	1837.7	248.58	1.000	2.19	537
0	1	1	0	0	1	0	1	0	0	1	0.381	1847.6	236.07	1.288	2.25	548814
0	0	1	0	1	0	0	1	0	0	1	0.338	2991.1	247.63	1.000	2.07	705
0	1	1	0	1	0	0	1	0	0	1	0.325	2532.4	239.77	1.349	2.19	205310
0	0	1	0	1	0	1	1	0	0	1	0.365	4382.2	241.04	1.000	2.10	5755
0	1	1	0	1	0	1	1	0	0	1	0.350	3596.3	233.20	1.380	2.23	1670410
1	0	1	0	1	0	0	1	0	0	1	0.356	4189.1	237.46	1.000	2.07	5755
1	1	1	0	1	0	0	1	0	0	1	0.342	3608.4	229.62	1.257	2.14	1670410

Continued on Next Page...

Table 22 – Continued

S1	S2	S3	S4	S5	S6	S7	S8	S9	S10	S11	M1	M2	M3	M4	OEC	#A
0	0	1	0	1	1	0	1	0	0	1	0.401	2439.5	238.06	1.000	2.15	11735
0	1	1	0	1	1	0	1	0	0	1	0.385	2025.7	230.53	1.386	2.29	3402770
1	0	1	0	0	0	1	1	0	0	1	0.366	4059.3	245.39	1.000	2.12	740
1	1	1	0	0	0	1	1	0	0	1	0.349	4102.4	232.88	1.243	2.16	756280
0	0	1	0	0	1	1	1	0	0	1	0.401	2569.9	246.77	1.000	2.19	1595
0	1	1	0	0	1	1	1	0	0	1	0.382	2553.8	234.25	1.310	2.26	1630090
1	0	1	0	1	0	1	1	0	0	1	0.368	5147.9	234.90	1.000	2.08	15580
1	1	1	0	1	0	1	1	0	0	1	0.353	4298.9	227.90	1.266	2.16	4498760
0	0	1	0	1	1	1	1	0	0	1	0.402	3450.2	236.57	1.000	2.15	32885
0	1	1	0	1	1	1	1	0	0	1	0.388	2813.8	230.06	1.478	2.34	9472470
1	0	1	0	0	1	0	1	0	0	1	0.397	2651.0	244.27	1.000	2.17	1595
1	1	1	0	0	1	0	1	0	0	1	0.378	2705.3	231.77	1.226	2.20	1630090
1	0	1	0	1	1	0	1	0	0	1	0.397	3376.8	234.08	1.000	2.13	32885
1	1	1	0	1	1	0	1	0	0	1	0.382	2862.5	227.48	1.430	2.29	9472470
0	0	1	0	0	0	0	1	1	0	1	0.343	1986.7	256.99	1.000	2.12	169

Continued on Next Page...

Table 22 – Continued

S1	S2	S3	S4	S5	S6	S7	S8	S9	S10	S11	M1	M2	M3	M4	OEC	#A
0	1	1	0	0	0	0	1	1	0	1	0.327	2062.9	244.50	1.293	2.19	172718
0	0	1	0	0	0	1	1	1	0	1	0.359	2746.5	251.99	1.000	2.13	529
0	1	1	0	0	0	1	1	1	0	1	0.342	2798.7	239.48	1.336	2.22	540638
1	0	1	0	0	0	0	1	1	0	1	0.351	2779.9	249.30	1.000	2.10	529
1	1	1	0	0	0	0	1	1	0	1	0.334	2911.4	236.79	1.252	2.15	540638
0	0	1	0	0	1	0	1	1	0	1	0.386	1583.2	250.67	1.000	2.17	1135
0	1	1	0	0	1	0	1	1	0	1	0.368	1626.3	238.16	1.314	2.25	1159970
0	0	1	0	1	0	0	1	1	0	1	0.345	2418.1	246.18	1.000	2.08	3767
0	1	1	0	1	0	0	1	1	0	1	0.332	2099.6	238.31	1.377	2.21	1094674
0	0	1	0	1	0	1	1	1	0	1	0.361	3448.1	241.36	1.000	2.09	11247
0	1	1	0	1	0	1	1	1	0	1	0.345	2914.3	233.57	1.417	2.24	3251234
1	0	1	0	1	0	0	1	1	0	1	0.352	3278.1	238.69	1.000	2.07	11247
1	1	1	0	1	0	0	1	1	0	1	0.339	2886.2	231.09	1.305	2.16	3251234
0	0	1	0	1	1	0	1	1	0	1	0.387	2009.2	240.35	1.000	2.14	23649
0	1	1	0	1	1	0	1	1	0	1	0.372	1713.0	233.60	1.389	2.28	6820478

Continued on Next Page...

Table 22 – Continued

S1	S2	S3	S4	S5	S6	S7	S8	S9	S10	S11	M1	M2	M3	M4	OEC	#A
0	0	1	0	0	0	0	1	0	1	1	0.414	2110.7	251.07	1.000	2.23	537
0	1	1	0	0	0	0	1	0	1	1	0.394	2182.5	238.56	1.358	2.32	548814
0	0	1	0	0	0	1	1	0	1	1	0.412	2919.8	248.19	1.000	2.21	1779
0	1	1	0	0	0	1	1	0	1	1	0.392	2952.9	235.66	1.377	2.32	1818138
1	0	1	0	0	0	0	1	0	1	1	0.404	2976.8	245.41	1.000	2.19	1779
1	1	1	0	0	0	0	1	0	1	1	0.385	3097.8	232.88	1.282	2.24	1818138
0	0	1	0	0	1	0	1	0	1	1	0.438	1772.6	245.58	1.000	2.25	2707
0	1	1	0	0	1	0	1	0	1	1	0.417	1802.9	233.06	1.378	2.35	2766554
0	0	1	0	1	0	0	1	0	1	1	0.416	2810.3	240.67	1.000	2.19	10631
0	1	1	0	1	0	0	1	0	1	1	0.399	2366.6	232.82	1.414	2.34	3047282
0	0	1	0	1	0	1	1	0	1	1	0.413	3945.7	238.09	1.000	2.18	33805
0	1	1	0	1	0	1	1	0	1	1	0.398	3268.8	232.12	1.488	2.37	9639910
1	0	1	0	1	0	0	1	0	1	1	0.405	3816.8	235.34	1.000	2.15	33805
1	1	1	0	1	0	0	1	0	1	1	0.390	3280.3	228.88	1.445	2.32	9639910
0	0	1	0	1	1	0	1	0	1	1	0.439	2416.6	235.56	1.000	2.21	51037

Continued on Next Page...

Table 22 – Continued

S1	S2	S3	S4	S5	S6	S7	S8	S9	S10	S11	M1	M2	M3	M4	OEC	#A
0	1	1	0	1	1	0	1	0	1	1	0.421	1970.7	228.50	1.496	2.40	1.4539014e7
1	0	1	0	0	1	1	1	0	0	1	0.398	3189.0	243.48	1.000	2.17	2164
1	1	1	0	0	1	1	1	0	0	1	0.379	3219.7	230.93	1.264	2.22	2211608
1	0	1	0	1	1	1	1	0	0	1	0.398	4137.1	233.66	1.000	2.13	41292
1	1	1	0	1	1	1	1	0	0	1	0.383	3452.9	226.69	1.465	2.31	1.1781224e7
1	0	1	0	0	0	1	1	1	0	1	0.362	3358.6	246.79	1.000	2.11	712
1	1	1	0	0	0	1	1	1	0	1	0.345	3459.3	234.29	1.303	2.19	727664
0	0	1	0	0	1	1	1	1	0	1	0.390	2190.0	248.84	1.000	2.17	1631
0	1	1	0	0	1	1	1	1	0	1	0.372	2228.0	236.35	1.338	2.26	1666882
1	0	1	0	1	0	1	1	1	0	1	0.363	4078.7	236.47	1.000	2.08	14136
1	1	1	0	1	0	1	1	1	0	1	0.350	3502.3	228.78	1.335	2.19	4053392
0	0	1	0	1	1	1	1	1	0	1	0.391	2810.2	238.91	1.000	2.14	31377
0	1	1	0	1	1	1	1	1	0	1	0.374	2340.6	231.71	1.482	2.32	8961694
1	0	1	0	0	0	1	1	0	1	1	0.405	3595.8	244.13	1.000	2.18	2550
1	1	1	0	0	0	1	1	0	1	1	0.385	3665.5	231.63	1.309	2.25	2606100

Continued on Next Page...

Table 22 – Continued

S1	S2	S3	S4	S5	S6	S7	S8	S9	S10	S11	M1	M2	M3	M4	OEC	#A
0	0	1	0	0	1	1	1	0	1	1	0.430	2437.7	244.80	1.000	2.23	3965
0	1	1	0	0	1	1	1	0	1	1	0.410	2443.7	232.63	1.384	2.34	4052230
1	0	1	0	1	0	1	1	0	1	1	0.406	4703.1	234.43	1.000	2.15	45834
1	1	1	0	1	0	1	1	0	1	1	0.390	3942.5	228.11	1.478	2.33	1.2973548e7
0	0	1	0	1	1	1	1	0	1	1	0.431	3340.6	235.17	1.000	2.20	70515
0	1	1	0	1	1	1	1	0	1	1	0.408	2870.3	229.22	1.516	2.39	1.6777216e7
1	0	1	0	0	1	0	1	1	0	1	0.385	2244.2	246.72	1.000	2.16	1631
1	1	1	0	0	1	0	1	1	0	1	0.366	2333.6	234.22	1.272	2.21	1666882
1	0	1	0	1	1	0	1	1	0	1	0.385	2747.1	236.81	1.000	2.12	31377
1	1	1	0	1	1	0	1	1	0	1	0.372	2362.8	230.22	1.433	2.29	8961694
1	0	1	0	0	1	0	1	0	1	1	0.425	2514.3	242.66	1.000	2.21	3965
1	1	1	0	0	1	0	1	0	1	1	0.405	2574.6	230.49	1.285	2.27	4052230
1	0	1	0	1	1	0	1	0	1	1	0.426	3314.2	233.05	1.000	2.18	70515
1	1	1	0	1	1	0	1	0	1	1	0.414	2864.2	225.85	1.474	2.37	1.6777216e7
0	0	1	0	0	0	0	1	1	1	1	0.398	1796.4	252.35	1.000	2.20	1135

Continued on Next Page...

Table 22 – Continued

S1	S2	S3	S4	S5	S6	S7	S8	S9	S10	S11	M1	M2	M3	M4	OEC	#A
0	1	1	0	0	0	0	1	1	1	1	0.379	1884.5	239.84	1.374	2.31	1159970
0	0	1	0	0	0	1	1	1	1	1	0.399	2468.1	249.84	1.000	2.19	1824
0	1	1	0	0	0	1	1	1	1	1	0.380	2542.2	237.30	1.395	2.31	1864128
1	0	1	0	0	0	0	1	1	1	1	0.390	2506.5	247.46	1.000	2.17	1824
1	1	1	0	0	0	0	1	1	1	1	0.371	2636.8	234.95	1.312	2.24	1864128
0	0	1	0	0	1	0	1	1	1	1	0.417	1557.3	248.06	1.000	2.22	2803
0	1	1	0	0	1	0	1	1	1	1	0.397	1612.1	235.59	1.398	2.34	2864666
0	0	1	0	1	0	0	1	1	1	1	0.399	2290.8	242.08	1.000	2.17	21809
0	1	1	0	1	0	0	1	1	1	1	0.385	1982.0	234.97	1.370	2.30	6227998
0	0	1	0	1	0	1	1	1	1	1	0.400	3195.4	240.00	1.000	2.16	33024
0	1	1	0	1	0	1	1	1	1	1	0.384	2653.9	233.11	1.489	2.35	9356928
1	0	1	0	1	0	0	1	1	1	1	0.391	3106.2	237.66	1.000	2.13	33024
1	1	1	0	1	0	0	1	1	1	1	0.379	2682.5	231.00	1.457	2.31	9356928
0	0	1	0	1	1	0	1	1	1	1	0.418	2039.0	238.32	1.000	2.19	50205
0	1	1	0	1	1	0	1	1	1	1	0.405	1741.6	231.74	1.500	2.39	1.420391e7

Continued on Next Page...

Table 22 – Continued

S1	S2	S3	S4	S5	S6	S7	S8	S9	S10	S11	M1	M2	M3	M4	OEC	#A
1	0	1	0	0	1	1	1	0	1	1	0.421	3038.6	242.41	1.000	2.21	2670
1	1	1	0	0	1	1	1	0	1	1	0.401	3078.9	229.86	1.319	2.28	2728740
1	0	1	0	1	1	1	1	0	1	1	0.421	4053.4	233.25	1.000	2.17	44082
1	1	1	0	1	1	1	1	0	1	1	0.404	3364.5	225.89	1.482	2.35	1.2325404e7
1	0	1	0	0	0	1	1	1	1	1	0.394	3069.6	246.10	1.000	2.17	1272
1	1	1	0	0	0	1	1	1	1	1	0.375	3177.2	233.61	1.338	2.26	1299984
0	0	1	0	0	1	1	1	1	1	1	0.415	2113.7	247.18	1.000	2.21	2016
0	1	1	0	0	1	1	1	1	1	1	0.395	2163.3	234.65	1.400	2.33	2060352
1	0	1	0	1	0	1	1	1	1	1	0.394	3872.7	236.83	1.000	2.14	21288
1	1	1	0	1	0	1	1	1	1	1	0.380	3307.3	229.67	1.346	2.25	5964336
0	0	1	0	1	1	1	1	1	1	1	0.415	2784.3	237.96	1.000	2.18	33408
0	1	1	0	1	1	1	1	1	1	1	0.398	2309.8	231.08	1.487	2.36	9346176
1	0	1	0	0	1	0	1	1	1	1	0.408	2177.1	245.25	1.000	2.19	2016
1	1	1	0	0	1	0	1	1	1	1	0.389	2267.1	232.75	1.331	2.27	2060352
1	0	1	0	1	1	0	1	1	1	1	0.409	2773.4	236.04	1.000	2.16	33408

Continued on Next Page...

Table 22 – Continued

S1	S2	S3	S4	S5	S6	S7	S8	S9	S10	S11	M1	M2	M3	M4	OEC	#A
1	1	1	0	1	1	0	1	1	1	1	0.396	2340.5	229.52	1.459	2.34	9346176
1	0	1	0	0	1	1	1	1	0	1	0.389	2721.1	245.85	1.000	2.16	1080
1	1	1	0	0	1	1	1	1	0	1	0.370	2802.9	233.36	1.298	2.23	1103760
1	0	1	0	1	1	1	1	1	0	1	0.389	3384.4	236.47	1.000	2.12	18696
1	1	1	0	1	1	1	1	1	0	1	0.375	2896.5	229.08	1.308	2.22	5264112
1	0	1	0	0	1	1	1	1	1	1	0.408	2643.3	244.83	1.000	2.19	534
1	1	1	0	0	1	1	1	1	1	1	0.388	2723.0	232.32	1.344	2.28	545748
1	0	1	0	1	1	1	1	1	1	1	0.408	3406.6	236.29	1.000	2.16	8010
1	1	1	0	1	1	1	1	1	1	1	0.391	2896.1	228.57	1.349	2.27	2205420
0	0	1	1	0	0	0	1	0	0	1	0.415	2122.8	258.45	1.000	2.26	31
0	1	1	1	0	0	0	1	0	0	1	0.395	2176.8	245.95	1.290	2.32	31682
0	0	1	1	0	0	1	1	0	0	1	0.398	3168.0	250.82	1.000	2.20	115
0	1	1	1	0	0	1	1	0	0	1	0.379	3161.7	238.31	1.313	2.27	117530
1	0	1	1	0	0	0	1	0	0	1	0.417	2745.7	251.22	1.000	2.23	115
1	1	1	1	0	0	0	1	0	0	1	0.397	2872.8	238.72	1.196	2.25	117530

Continued on Next Page...

Table 22 – Continued

S1	S2	S3	S4	S5	S6	S7	S8	S9	S10	S11	M1	M2	M3	M4	OEC	#A
0	0	1	1	0	1	0	1	0	0	1	0.456	1436.8	252.93	1.000	2.31	245
0	1	1	1	0	1	0	1	0	0	1	0.435	1461.9	240.43	1.296	2.37	250390
0	0	1	1	1	0	0	1	0	0	1	0.419	2706.9	247.63	1.000	2.22	705
0	1	1	1	1	0	0	1	0	0	1	0.402	2303.0	239.77	1.386	2.35	205310
0	0	1	1	1	0	1	1	0	0	1	0.401	4189.4	240.12	1.000	2.16	2525
0	1	1	1	1	0	1	1	0	0	1	0.385	3443.1	232.27	1.392	2.30	732550
1	0	1	1	1	0	0	1	0	0	1	0.420	3344.1	240.52	1.000	2.20	2525
1	1	1	1	1	0	0	1	0	0	1	0.403	2922.4	232.64	1.250	2.26	732550
0	0	1	1	1	1	0	1	0	0	1	0.459	1925.1	242.50	1.000	2.28	5275
0	1	1	1	1	1	0	1	0	0	1	0.441	1616.0	234.67	1.382	2.40	1527050
1	0	1	1	0	0	1	1	0	0	1	0.405	3549.2	246.77	1.000	2.19	170
1	1	1	1	0	0	1	1	0	0	1	0.386	3617.3	234.27	1.247	2.23	173740
0	0	1	1	0	1	1	1	0	0	1	0.434	2220.2	248.87	1.000	2.26	390
0	1	1	1	0	1	1	1	0	0	1	0.414	2224.9	236.36	1.313	2.33	398580
1	0	1	1	1	0	1	1	0	0	1	0.406	4493.1	236.31	1.000	2.16	3510

Continued on Next Page...

Table 22 – Continued

S1	S2	S3	S4	S5	S6	S7	S8	S9	S10	S11	M1	M2	M3	M4	OEC	#A
1	1	1	1	1	0	1	1	0	0	1	0.390	3791.8	228.47	1.289	2.24	1011220
0	0	1	1	1	1	1	1	0	0	1	0.436	2998.2	238.85	1.000	2.22	7770
0	1	1	1	1	1	1	1	0	0	1	0.419	2473.7	231.03	1.374	2.34	2228940
1	0	1	1	0	1	0	1	0	0	1	0.452	1989.3	249.21	1.000	2.29	390
1	1	1	1	0	1	0	1	0	0	1	0.431	2067.6	236.71	1.212	2.31	398580
1	0	1	1	1	1	0	1	0	0	1	0.453	2546.2	239.17	1.000	2.25	7770
1	1	1	1	1	1	0	1	0	0	1	0.435	2194.5	231.42	1.264	2.32	2228940
0	0	1	1	0	0	0	1	1	0	1	0.426	1630.6	255.22	1.000	2.26	69
0	1	1	1	0	0	0	1	1	0	1	0.406	1713.2	242.71	1.310	2.33	70518
0	0	1	1	0	0	1	1	1	0	1	0.415	2334.4	250.40	1.000	2.23	115
0	1	1	1	0	0	1	1	1	0	1	0.396	2411.4	237.90	1.362	2.32	117530
1	0	1	1	0	0	0	1	1	0	1	0.421	2220.5	249.88	1.000	2.23	115
1	1	1	1	0	0	0	1	1	0	1	0.401	2356.5	237.38	1.271	2.29	117530
0	0	1	1	0	1	0	1	1	0	1	0.457	1226.4	252.01	1.000	2.31	261
0	1	1	1	0	1	0	1	1	0	1	0.435	1280.0	239.51	1.321	2.38	266742

Continued on Next Page...

Table 22 – Continued

S1	S2	S3	S4	S5	S6	S7	S8	S9	S10	S11	M1	M2	M3	M4	OEC	#A
0	0	1	1	1	0	0	1	1	0	1	0.429	1952.0	244.38	1.000	2.23	1531
0	1	1	1	1	0	0	1	1	0	1	0.412	1719.4	236.53	1.391	2.36	444682
0	0	1	1	1	0	1	1	1	0	1	0.417	2865.4	239.78	1.000	2.19	2397
0	1	1	1	1	0	1	1	1	0	1	0.401	2461.0	231.90	1.418	2.34	691334
1	0	1	1	1	0	0	1	1	0	1	0.423	2574.7	239.29	1.000	2.20	2397
1	1	1	1	1	0	0	1	1	0	1	0.406	2304.3	231.42	1.315	2.29	691334
0	0	1	1	1	1	0	1	1	0	1	0.458	1542.0	241.85	1.000	2.27	5259
0	1	1	1	1	1	0	1	1	0	1	0.440	1339.6	234.01	1.388	2.40	1510698
0	0	1	1	0	0	0	1	0	1	1	0.460	1659.2	251.86	1.000	2.31	245
0	1	1	1	0	0	0	1	0	1	1	0.438	1736.9	239.36	1.371	2.41	250390
0	0	1	1	0	0	1	1	0	1	1	0.441	2538.4	248.28	1.000	2.26	445
0	1	1	1	0	0	1	1	0	1	1	0.420	2588.3	235.78	1.385	2.37	454790
1	0	1	1	0	0	0	1	0	1	1	0.452	2317.1	248.18	1.000	2.29	445
1	1	1	1	0	0	0	1	0	1	1	0.431	2446.2	235.69	1.272	2.33	454790
0	0	1	1	0	1	0	1	0	1	1	0.479	1327.1	248.92	1.000	2.34	683

Continued on Next Page...

Table 22 – Continued

S1	S2	S3	S4	S5	S6	S7	S8	S9	S10	S11	M1	M2	M3	M4	OEC	#A
0	1	1	1	0	1	0	1	0	1	1	0.456	1368.3	236.41	1.379	2.44	698026
0	0	1	1	1	0	0	1	0	1	1	0.462	2212.8	241.52	1.000	2.28	4779
0	1	1	1	1	0	0	1	0	1	1	0.444	1885.7	233.64	1.409	2.42	1367338
0	0	1	1	1	0	1	1	0	1	1	0.443	3426.8	238.37	1.000	2.23	8211
0	1	1	1	1	0	1	1	0	1	1	0.425	2852.7	230.56	1.408	2.37	2332442
1	0	1	1	1	0	0	1	0	1	1	0.454	2967.3	238.26	1.000	2.25	8211
1	1	1	1	1	0	0	1	0	1	1	0.436	2571.7	230.43	1.306	2.34	2332442
0	0	1	1	1	1	0	1	0	1	1	0.480	1819.8	239.05	1.000	2.30	12453
0	1	1	1	1	1	0	1	0	1	1	0.461	1530.6	231.35	1.420	2.45	3531766
1	0	1	1	0	1	1	1	0	0	1	0.436	2622.6	246.57	1.000	2.25	276
1	1	1	1	0	1	1	1	0	0	1	0.415	2683.1	234.06	1.247	2.29	282072
1	0	1	1	1	1	1	1	0	0	1	0.436	3407.2	237.09	1.000	2.21	4908
1	1	1	1	1	1	1	1	0	0	1	0.419	2884.1	229.30	1.272	2.29	1387176
1	0	1	1	0	0	1	1	1	0	1	0.416	2810.8	247.06	1.000	2.21	78
1	1	1	1	0	0	1	1	1	0	1	0.396	2936.2	234.56	1.338	2.30	79716

Continued on Next Page...

Table 22 – Continued

S1	S2	S3	S4	S5	S6	S7	S8	S9	S10	S11	M1	M2	M3	M4	OEC	#A
0	0	1	1	0	1	1	1	1	0	1	0.444	1793.4	249.49	1.000	2.27	198
0	1	1	1	0	1	1	1	1	0	1	0.423	1856.0	236.99	1.347	2.36	202356
1	0	1	1	1	0	1	1	1	0	1	0.416	3333.7	236.98	1.000	2.18	1458
1	1	1	1	1	0	1	1	1	0	1	0.400	2917.3	229.12	1.356	2.29	414876
0	0	1	1	1	1	1	1	1	0	1	0.444	2263.8	239.91	1.000	2.24	3546
0	1	1	1	1	1	1	1	1	0	1	0.427	1945.5	232.12	1.381	2.37	1003212
1	0	1	1	0	0	1	1	0	1	1	0.439	3075.3	245.83	1.000	2.25	336
1	1	1	1	0	0	1	1	0	1	1	0.418	3166.6	233.34	1.295	2.31	343392
0	0	1	1	0	1	1	1	0	1	1	0.458	2001.9	246.93	1.000	2.29	534
0	1	1	1	0	1	1	1	0	1	1	0.436	2030.6	234.45	1.390	2.40	545748
1	0	1	1	1	0	1	1	0	1	1	0.440	4015.6	236.49	1.000	2.22	5712
1	1	1	1	1	0	1	1	0	1	1	0.423	3395.4	228.70	1.306	2.31	1604064
0	0	1	1	1	1	1	1	0	1	1	0.459	2753.2	237.63	1.000	2.26	8970
0	1	1	1	1	1	1	1	0	1	1	0.441	2283.7	229.86	1.404	2.40	2514540
1	0	1	1	0	1	0	1	1	0	1	0.451	1731.0	249.15	1.000	2.29	198

Continued on Next Page...

Table 22 – Continued

S1	S2	S3	S4	S5	S6	S7	S8	S9	S10	S11	M1	M2	M3	M4	OEC	#A
1	1	1	1	0	1	0	1	1	0	1	0.429	1832.6	236.63	1.269	2.33	202356
1	0	1	1	1	1	0	1	1	0	1	0.451	2093.8	239.59	1.000	2.25	3546
1	1	1	1	1	1	0	1	1	0	1	0.434	1856.0	231.79	1.294	2.34	1003212
1	0	1	1	0	1	0	1	0	1	1	0.470	1892.8	246.92	1.000	2.31	534
1	1	1	1	0	1	0	1	0	1	1	0.448	1972.3	234.44	1.286	2.37	545748
1	0	1	1	1	1	0	1	0	1	1	0.471	2507.2	237.62	1.000	2.28	8970
1	1	1	1	1	1	0	1	0	1	1	0.452	2144.6	229.86	1.306	2.37	2514540
0	0	1	1	0	0	0	1	1	1	1	0.464	1419.6	251.94	1.000	2.32	261
0	1	1	1	0	0	0	1	1	1	1	0.442	1507.9	239.44	1.384	2.42	266742
0	0	1	1	0	0	1	1	1	1	1	0.451	2052.6	249.32	1.000	2.29	228
0	1	1	1	0	0	1	1	1	1	1	0.430	2140.5	236.81	1.402	2.40	233016
1	0	1	1	0	0	0	1	1	1	1	0.452	1987.6	248.54	1.000	2.29	228
1	1	1	1	0	0	0	1	1	1	1	0.430	2117.3	236.05	1.321	2.36	233016
0	0	1	1	0	1	0	1	1	1	1	0.476	1191.4	249.55	1.000	2.33	354
0	1	1	1	0	1	0	1	1	1	1	0.454	1250.2	237.08	1.400	2.45	361788

Continued on Next Page...

Table 22 – Continued

S1	S2	S3	S4	S5	S6	S7	S8	S9	S10	S11	M1	M2	M3	M4	OEC	#A
0	0	1	1	1	0	0	1	1	1	1	0.466	1792.2	241.85	1.000	2.29	4875
0	1	1	1	1	0	0	1	1	1	1	0.448	1569.7	233.99	1.413	2.43	1387050
0	0	1	1	1	0	1	1	1	1	1	0.452	2619.6	239.88	1.000	2.25	3900
0	1	1	1	1	0	1	1	1	1	1	0.434	2252.0	232.09	1.420	2.40	1096200
1	0	1	1	1	0	0	1	1	1	1	0.453	2439.8	239.12	1.000	2.25	3900
1	1	1	1	1	0	0	1	1	1	1	0.435	2155.2	231.32	1.333	2.36	1096200
0	0	1	1	1	1	0	1	1	1	1	0.477	1550.7	240.16	1.000	2.30	5982
0	1	1	1	1	1	0	1	1	1	1	0.458	1337.8	232.36	1.416	2.44	1678404
1	0	1	1	0	1	1	1	0	1	1	0.454	2481.5	245.45	1.000	2.28	150
1	1	1	1	0	1	1	1	0	1	1	0.432	2544.0	232.96	1.307	2.34	153300
1	0	1	1	1	1	1	1	0	1	1	0.454	3329.0	236.92	1.000	2.25	2250
1	1	1	1	1	1	1	1	0	1	1	0.436	2804.4	229.22	1.303	2.33	619500
1	0	1	1	0	0	1	1	1	1	1	0.444	2590.5	246.73	1.000	2.27	66
1	1	1	1	0	0	1	1	1	1	1	0.423	2710.9	234.24	1.333	2.35	67452
0	0	1	1	0	1	1	1	1	1	1	0.462	1699.1	248.24	1.000	2.30	108

Continued on Next Page...

Table 22 – Continued

S1	S2	S3	S4	S5	S6	S7	S8	S9	S10	S11	M1	M2	M3	M4	OEC	#A
0	1	1	1	0	1	1	1	1	1	1	0.440	1764.3	235.74	1.409	2.42	110376
1	0	1	1	1	0	1	1	1	1	1	0.444	3237.0	238.19	1.000	2.23	990
1	1	1	1	1	0	1	1	1	1	1	0.427	2808.4	230.50	1.327	2.33	272580
0	0	1	1	1	1	1	1	1	1	1	0.462	2218.2	239.71	1.000	2.27	1620
0	1	1	1	1	1	1	1	1	1	1	0.444	1894.5	232.00	1.402	2.41	446040
1	0	1	1	0	1	0	1	1	1	1	0.465	1686.9	247.70	1.000	2.31	108
1	1	1	1	0	1	0	1	1	1	1	0.443	1781.4	235.21	1.331	2.38	110376
1	0	1	1	1	1	0	1	1	1	1	0.465	2138.5	239.17	1.000	2.27	1620
1	1	1	1	1	1	0	1	1	1	1	0.447	1863.8	231.48	1.326	2.37	446040
0	0	1	0	0	1	1	1	0	0	0	0.416	2176.3	244.72	1.000	2.21	146
0	1	1	0	0	1	1	1	0	0	0	0.396	2127.8	232.22	1.538	2.39	149212
1	0	1	0	0	1	1	1	0	0	0	0.408	3017.5	242.48	1.000	2.18	460
1	1	1	0	0	1	1	1	0	0	0	0.389	3025.8	229.98	1.387	2.29	470120
0	0	1	0	1	1	1	1	0	0	0	0.418	2980.2	232.49	1.000	2.16	4030
0	1	1	0	1	1	1	1	0	0	0	0.402	2381.1	224.57	1.533	2.37	1195460

Continued on Next Page...

Table 22 – Continued

S1	S2	S3	S4	S5	S6	S7	S8	S9	S10	S11	M1	M2	M3	M4	OEC	#A
1	0	1	0	1	1	1	1	0	0	0	0.408	3893.3	231.02	1.000	2.14	11380
1	1	1	0	1	1	1	1	0	0	0	0.392	3215.2	223.31	1.406	2.28	3342360
0	0	1	0	0	1	1	1	1	0	0	0.398	1854.5	248.84	1.000	2.19	322
0	1	1	0	0	1	1	1	1	0	0	0.379	1878.4	236.33	1.553	2.38	329084
1	0	1	0	0	1	1	1	1	0	0	0.394	2526.4	246.25	1.000	2.17	496
1	1	1	0	0	1	1	1	1	0	0	0.376	2606.0	233.76	1.435	2.31	506912
0	0	1	0	1	1	1	1	1	0	0	0.399	2376.5	237.10	1.000	2.14	8142
0	1	1	0	1	1	1	1	1	0	0	0.383	1986.8	229.18	1.551	2.36	2396324
1	0	1	0	1	1	1	1	1	0	0	0.394	3078.3	235.43	1.000	2.13	10896
1	1	1	0	1	1	1	1	1	0	0	0.378	2642.0	227.52	1.447	2.29	3161312
0	0	1	0	0	1	1	1	0	1	0	0.448	2173.3	242.14	1.000	2.25	1250
0	1	1	0	0	1	1	1	0	1	0	0.427	2159.6	229.64	1.533	2.43	1277500
1	0	1	0	0	1	1	1	0	1	0	0.433	2968.9	240.78	1.000	2.22	1838
1	1	1	0	0	1	1	1	0	1	0	0.413	2991.5	228.25	1.420	2.35	1878436
0	0	1	0	1	1	1	1	0	1	0	0.449	3058.4	231.23	1.000	2.21	25870

Continued on Next Page...

Table 22 – Continued

S1	S2	S3	S4	S5	S6	S7	S8	S9	S10	S11	M1	M2	M3	M4	OEC	#A
0	1	1	0	1	1	1	1	0	1	0	0.428	2425.8	224.32	1.550	2.42	7455140
1	0	1	0	1	1	1	1	0	1	0	0.434	4009.0	230.45	1.000	2.18	35154
1	1	1	0	1	1	1	1	0	1	0	0.415	3326.8	222.82	1.465	2.35	1.0032988e7
0	0	1	0	0	1	1	1	1	1	0	0.426	1886.7	246.15	1.000	2.23	1305
0	1	1	0	0	1	1	1	1	1	0	0.405	1922.1	233.66	1.548	2.42	1333710
1	0	1	0	0	1	1	1	1	1	0	0.415	2543.3	244.46	1.000	2.20	936
1	1	1	0	0	1	1	1	1	1	0	0.396	2616.3	231.96	1.453	2.35	956592
0	0	1	0	1	1	1	1	1	1	0	0.426	2513.4	235.63	1.000	2.19	25239
0	1	1	0	1	1	1	1	1	1	0	0.409	2097.6	228.65	1.565	2.41	7213458
1	0	1	0	1	1	1	1	1	1	0	0.416	3271.9	234.78	1.000	2.17	16248
1	1	1	0	1	1	1	1	1	1	0	0.401	2776.9	227.20	1.444	2.33	4576656
1	0	1	0	0	1	0	1	0	0	0	0.411	2343.5	241.21	1.000	2.18	146
1	1	1	0	0	1	0	1	0	0	0	0.391	2368.9	228.72	1.343	2.27	149212
1	0	1	0	1	1	0	1	0	0	0	0.412	3014.4	228.92	1.000	2.14	4030
1	1	1	0	1	1	0	1	0	0	0	0.396	2514.2	221.01	1.351	2.25	1195460

Continued on Next Page...

Table 22 – Continued

S1	S2	S3	S4	S5	S6	S7	S8	S9	S10	S11	M1	M2	M3	M4	OEC	#A
1	0	1	0	0	1	0	1	1	0	0	0.390	1933.7	246.11	1.000	2.16	322
1	1	1	0	0	1	0	1	1	0	0	0.372	2011.0	233.60	1.408	2.29	329084
1	0	1	0	1	1	0	1	1	0	0	0.391	2334.6	234.42	1.000	2.12	8142
1	1	1	0	1	1	0	1	1	0	0	0.375	2031.1	226.59	1.407	2.26	2396324
1	0	1	0	0	1	0	1	0	1	0	0.441	2314.2	239.38	1.000	2.23	1250
1	1	1	0	0	1	0	1	0	1	0	0.420	2355.5	226.88	1.380	2.33	1277500
1	0	1	0	1	1	0	1	0	1	0	0.442	3112.3	228.51	1.000	2.19	25870
1	1	1	0	1	1	0	1	0	1	0	0.428	2587.6	221.70	1.427	2.35	7455140
1	0	1	0	0	1	0	1	1	1	0	0.416	1980.1	243.76	1.000	2.20	1305
1	1	1	0	0	1	0	1	1	1	0	0.396	2056.5	231.26	1.436	2.34	1333710
1	0	1	0	1	1	0	1	1	1	0	0.417	2536.9	233.30	1.000	2.16	25239
1	1	1	0	1	1	0	1	1	1	0	0.402	2162.8	226.55	1.430	2.32	7213458
0	0	1	0	0	1	0	1	0	0	0	0.426	1021.3	242.50	1.000	2.21	16
0	1	1	0	0	1	0	1	0	0	0	0.405	1000.0	230.00	1.531	2.40	16352
0	0	1	0	1	1	0	1	0	0	0	0.430	1442.1	229.48	1.000	2.17	480

Continued on Next Page...

Table 22 – Continued

S1	S2	S3	S4	S5	S6	S7	S8	S9	S10	S11	M1	M2	M3	M4	OEC	#A
0	1	1	0	1	1	0	1	0	0	0	0.413	1155.8	221.55	1.529	2.37	143360
0	0	1	0	0	1	0	1	1	0	0	0.393	980.6	249.15	1.000	2.18	92
0	1	1	0	0	1	0	1	1	0	0	0.375	998.7	236.65	1.544	2.37	94024
0	0	1	0	1	1	0	1	1	0	0	0.396	1270.9	236.62	1.000	2.14	2612
0	1	1	0	1	1	0	1	1	0	0	0.380	1072.1	228.70	1.536	2.35	776664
0	0	1	0	0	1	0	1	0	1	0	0.464	1175.0	240.48	1.000	2.28	368
0	1	1	0	0	1	0	1	0	1	0	0.442	1178.9	228.00	1.524	2.45	376096
0	0	1	0	1	1	0	1	0	1	0	0.466	1667.4	229.13	1.000	2.24	8096
0	1	1	0	1	1	0	1	0	1	0	0.447	1354.8	221.22	1.524	2.43	2349312
0	0	1	0	0	1	0	1	1	1	0	0.430	1129.2	245.46	1.000	2.23	790
0	1	1	0	0	1	0	1	1	1	0	0.409	1160.7	232.97	1.536	2.42	807380
0	0	1	0	1	1	0	1	1	1	0	0.431	1507.8	234.33	1.000	2.19	16634
0	1	1	0	1	1	0	1	1	1	0	0.415	1270.8	226.94	1.537	2.40	4803148
0	0	1	1	0	1	0	1	0	0	0	0.448	698.8	251.50	1.000	2.29	16
0	1	1	1	0	1	0	1	0	0	0	0.427	700.0	239.00	1.531	2.47	16352

Continued on Next Page...

Table 22 – Continued

S1	S2	S3	S4	S5	S6	S7	S8	S9	S10	S11	M1	M2	M3	M4	OEC	#A
0	0	1	1	0	1	1	1	0	0	0	0.428	1866.2	248.38	1.000	2.24	65
0	1	1	1	0	1	1	1	0	0	0	0.408	1846.1	235.88	1.539	2.43	66430
1	0	1	1	0	1	0	1	0	0	0	0.451	1604.1	248.80	1.000	2.29	65
1	1	1	1	0	1	0	1	0	0	0	0.430	1668.4	236.31	1.319	2.36	66430
0	0	1	1	1	1	0	1	0	0	0	0.453	987.5	238.48	1.000	2.25	480
0	1	1	1	1	1	0	1	0	0	0	0.435	809.7	230.55	1.529	2.45	143360
0	0	1	1	1	1	1	1	0	0	0	0.430	2567.1	236.18	1.000	2.20	1775
0	1	1	1	1	1	1	1	0	0	0	0.413	2077.7	228.26	1.534	2.40	526050
1	0	1	1	1	1	0	1	0	0	0	0.453	2068.2	236.58	1.000	2.24	1775
1	1	1	1	1	1	0	1	0	0	0	0.435	1778.5	228.63	1.329	2.34	526050
1	0	1	1	0	1	1	1	0	0	0	0.435	2421.1	246.79	1.000	2.25	110
1	1	1	1	0	1	1	1	0	0	0	0.415	2476.1	234.29	1.371	2.35	112420
1	0	1	1	1	1	1	1	0	0	0	0.435	3118.6	235.51	1.000	2.21	2610
1	1	1	1	1	1	1	1	0	0	0	0.418	2633.0	227.62	1.374	2.33	763420
0	0	1	1	0	1	0	1	1	0	0	0.455	751.4	251.74	1.000	2.30	38

Continued on Next Page...

Table 22 – Continued

S1	S2	S3	S4	S5	S6	S7	S8	S9	S10	S11	M1	M2	M3	M4	OEC	#A
0	1	1	1	0	1	0	1	1	0	0	0.434	781.3	239.24	1.546	2.49	38836
0	0	1	1	0	1	1	1	1	0	0	0.444	1525.4	250.21	1.000	2.28	73
0	1	1	1	0	1	1	1	1	0	0	0.423	1578.8	237.70	1.558	2.47	74606
1	0	1	1	0	1	0	1	1	0	0	0.450	1451.9	249.62	1.000	2.29	73
1	1	1	1	0	1	0	1	1	0	0	0.429	1546.2	237.11	1.411	2.41	74606
0	0	1	1	1	1	0	1	1	0	0	0.458	958.1	239.25	1.000	2.26	1066
0	1	1	1	1	1	0	1	1	0	0	0.440	827.9	231.33	1.538	2.47	316652
0	0	1	1	1	1	1	1	1	0	0	0.445	1904.8	238.65	1.000	2.23	1767
0	1	1	1	1	1	1	1	1	0	0	0.427	1636.5	230.74	1.550	2.45	517874
1	0	1	1	1	1	0	1	1	0	0	0.451	1720.2	238.13	1.000	2.25	1767
1	1	1	1	1	1	0	1	1	0	0	0.433	1542.3	230.21	1.409	2.39	517874
0	0	1	1	0	1	0	1	0	1	0	0.479	820.7	245.67	1.000	2.32	168
0	1	1	1	0	1	0	1	0	1	0	0.456	838.1	233.18	1.524	2.50	171696
0	0	1	1	0	1	1	1	0	1	0	0.459	1751.4	245.36	1.000	2.29	311
0	1	1	1	0	1	1	1	0	1	0	0.437	1761.4	232.87	1.536	2.47	317842

Continued on Next Page...

Table 22 – Continued

S1	S2	S3	S4	S5	S6	S7	S8	S9	S10	S11	M1	M2	M3	M4	OEC	#A
1	0	1	1	0	1	0	1	0	1	0	0.472	1669.8	245.26	1.000	2.31	311
1	1	1	1	0	1	0	1	0	1	0	0.450	1731.9	232.75	1.368	2.41	317842
0	0	1	1	1	1	0	1	0	1	0	0.480	1168.2	234.29	1.000	2.28	3624
0	1	1	1	1	1	0	1	0	1	0	0.461	966.3	226.34	1.521	2.48	1049328
0	0	1	1	1	1	1	1	0	1	0	0.460	2464.3	234.61	1.000	2.25	6201
0	1	1	1	1	1	1	1	0	1	0	0.442	2014.9	226.70	1.527	2.45	1779022
1	0	1	1	1	1	0	1	0	1	0	0.473	2260.2	234.51	1.000	2.27	6201
1	1	1	1	1	1	0	1	0	1	0	0.455	1913.3	226.59	1.371	2.39	1779022
1	0	1	1	0	1	1	1	1	0	0	0.443	2025.8	248.48	1.000	2.27	60
1	1	1	1	0	1	1	1	1	0	0	0.422	2138.5	235.98	1.433	2.40	61320
1	0	1	1	1	1	1	1	1	0	0	0.443	2404.2	238.18	1.000	2.23	1188
1	1	1	1	1	1	1	1	1	0	0	0.425	2131.7	230.33	1.429	2.38	340536
1	0	1	1	0	1	1	1	0	1	0	0.457	2378.4	244.91	1.000	2.28	240
1	1	1	1	0	1	1	1	0	1	0	0.435	2430.4	232.42	1.401	2.39	245280
1	0	1	1	1	1	1	1	0	1	0	0.458	3220.4	235.04	1.000	2.24	4272

Continued on Next Page...

Table 22 – Continued

S1	S2	S3	S4	S5	S6	S7	S8	S9	S10	S11	M1	M2	M3	M4	OEC	#A
1	1	1	1	1	1	1	1	0	1	0	0.440	2694.7	227.19	1.400	2.38	1207584
0	0	1	1	0	1	0	1	1	1	0	0.479	865.5	247.84	1.000	2.33	182
0	1	1	1	0	1	0	1	1	1	0	0.456	904.8	235.33	1.541	2.51	186004
0	0	1	1	0	1	1	1	1	1	0	0.467	1514.8	248.02	1.000	2.31	162
0	1	1	1	0	1	1	1	1	1	0	0.445	1570.4	235.53	1.554	2.50	165564
1	0	1	1	0	1	0	1	1	1	0	0.467	1509.6	247.21	1.000	2.31	162
1	1	1	1	0	1	0	1	1	1	0	0.445	1595.9	234.70	1.439	2.44	165564
0	0	1	1	1	1	0	1	1	1	0	0.481	1146.9	236.86	1.000	2.29	3690
0	1	1	1	1	1	0	1	1	1	0	0.462	982.0	228.93	1.536	2.50	1060780
0	0	1	1	1	1	1	1	1	1	0	0.468	1987.7	237.99	1.000	2.27	2910
0	1	1	1	1	1	1	1	1	1	0	0.449	1693.4	230.14	1.546	2.48	823620
1	0	1	1	1	1	0	1	1	1	0	0.468	1917.4	237.21	1.000	2.27	2910
1	1	1	1	1	1	0	1	1	1	0	0.450	1674.4	229.33	1.434	2.43	823620
1	0	1	1	0	1	1	1	1	0	1	0.442	2202.4	247.59	1.000	2.26	54
1	1	1	1	0	1	1	1	1	0	1	0.421	2312.2	235.09	1.303	2.33	55188

Continued on Next Page...

Table 22 – Continued

S1	S2	S3	S4	S5	S6	S7	S8	S9	S10	S11	M1	M2	M3	M4	OEC	#A
1	0	1	1	1	1	1	1	1	0	1	0.442	2689.2	239.06	1.000	2.23	810
1	1	1	1	1	1	1	1	1	0	1	0.425	2361.5	231.37	1.297	2.32	223020
1	0	1	1	0	1	1	1	1	1	0	0.459	2041.2	247.13	1.000	2.29	48
1	1	1	1	0	1	1	1	1	1	0	0.438	2140.0	234.62	1.451	2.43	49056
1	0	1	1	1	1	1	1	1	1	0	0.460	2578.8	238.59	1.000	2.26	720
1	1	1	1	1	1	1	1	1	1	0	0.441	2239.4	230.88	1.443	2.42	198240
0	0	1	0	0	0	1	1	0	1	0	0.437	2772.0	244.93	1.000	2.25	184
0	1	1	0	0	0	1	1	0	1	0	0.417	2762.7	232.45	1.532	2.43	188048
1	0	1	0	0	0	1	1	0	1	0	0.423	3737.9	241.83	1.000	2.21	606
1	1	1	0	0	0	1	1	0	1	0	0.403	3777.5	229.33	1.397	2.32	619332
0	0	1	0	1	0	1	1	0	1	0	0.439	3894.5	233.49	1.000	2.20	4104
0	1	1	0	1	0	1	1	0	1	0	0.422	3140.5	225.54	1.527	2.41	1192688
1	0	1	0	1	0	1	1	0	1	0	0.424	4961.6	230.85	1.000	2.17	12898
1	1	1	0	1	0	1	1	0	1	0	0.407	4099.0	223.28	1.401	2.31	3728956
0	0	1	0	0	0	1	1	1	1	0	0.415	2301.6	248.54	1.000	2.22	395

Continued on Next Page...

Table 22 – Continued

S1	S2	S3	S4	S5	S6	S7	S8	S9	S10	S11	M1	M2	M3	M4	OEC	#A
0	1	1	0	0	0	1	1	1	1	0	0.396	2353.6	236.03	1.545	2.41	403690
1	0	1	0	0	0	1	1	1	1	0	0.408	3117.0	245.35	1.000	2.19	616
1	1	1	0	0	0	1	1	1	1	0	0.389	3213.4	232.84	1.442	2.33	629552
0	0	1	0	1	0	1	1	1	1	0	0.416	3026.6	237.26	1.000	2.18	8565
0	1	1	0	1	0	1	1	1	1	0	0.400	2536.3	229.32	1.545	2.39	2481430
1	0	1	0	1	0	1	1	1	1	0	0.408	3910.3	234.75	1.000	2.15	12408
1	1	1	0	1	0	1	1	1	1	0	0.393	3334.7	226.88	1.441	2.31	3564176
1	0	1	0	0	0	0	1	0	1	0	0.424	2937.3	240.87	1.000	2.21	184
1	1	1	0	0	0	0	1	0	1	0	0.404	3030.3	228.36	1.352	2.30	188048
1	0	1	0	1	0	0	1	0	1	0	0.426	3862.0	229.45	1.000	2.17	4104
1	1	1	0	1	0	0	1	0	1	0	0.409	3252.1	221.54	1.350	2.28	1192688
1	0	1	0	0	0	0	1	1	1	0	0.401	2380.5	245.39	1.000	2.18	395
1	1	1	0	0	0	0	1	1	1	0	0.382	2495.9	232.92	1.414	2.30	403690
1	0	1	0	1	0	0	1	1	1	0	0.402	2961.3	234.21	1.000	2.14	8565
1	1	1	0	1	0	0	1	1	1	0	0.386	2571.3	226.26	1.408	2.28	2481430

Continued on Next Page...

Table 22 – Continued

S1	S2	S3	S4	S5	S6	S7	S8	S9	S10	S11	M1	M2	M3	M4	OEC	#A
0	0	1	0	0	0	0	1	1	1	0	0.412	1288.4	249.30	1.000	2.21	92
0	1	1	0	0	0	0	1	1	1	0	0.392	1345.4	236.80	1.500	2.38	94024
0	0	1	0	1	0	0	1	1	1	0	0.414	1680.2	237.48	1.000	2.17	2116
0	1	1	0	1	0	0	1	1	1	0	0.397	1433.1	229.51	1.500	2.36	616952
0	0	1	1	0	0	0	1	0	1	0	0.453	990.0	247.00	1.000	2.28	16
0	1	1	1	0	0	0	1	0	1	0	0.431	1022.4	234.50	1.500	2.44	16352
0	0	1	1	0	0	1	1	0	1	0	0.441	2390.7	245.74	1.000	2.26	84
0	1	1	1	0	0	1	1	0	1	0	0.420	2399.2	233.24	1.533	2.44	85848
1	0	1	1	0	0	0	1	0	1	0	0.454	2181.3	245.45	1.000	2.28	84
1	1	1	1	0	0	0	1	0	1	0	0.433	2284.6	232.94	1.336	2.36	85848
0	0	1	1	1	0	0	1	0	1	0	0.455	1400.9	235.17	1.000	2.24	368
0	1	1	1	1	0	0	1	0	1	0	0.437	1161.4	227.21	1.500	2.43	107296
0	0	1	1	1	0	1	1	0	1	0	0.443	3363.2	234.32	1.000	2.21	1868
0	1	1	1	1	0	1	1	0	1	0	0.425	2732.7	226.39	1.529	2.42	542696
1	0	1	1	1	0	0	1	0	1	0	0.456	2863.3	234.08	1.000	2.24	1868

Continued on Next Page...

Table 22 – Continued

S1	S2	S3	S4	S5	S6	S7	S8	S9	S10	S11	M1	M2	M3	M4	OEC	#A
1	1	1	1	1	0	0	1	0	1	0	0.438	2449.7	226.15	1.337	2.34	542696
1	0	1	1	0	0	1	1	0	1	0	0.447	3184.2	244.73	1.000	2.26	146
1	1	1	1	0	0	1	1	0	1	0	0.426	3248.2	232.22	1.380	2.37	149212
1	0	1	1	1	0	1	1	0	1	0	0.448	4211.2	233.91	1.000	2.22	3054
1	1	1	1	1	0	1	1	0	1	0	0.430	3517.9	226.00	1.377	2.35	881188
0	0	1	1	0	0	0	1	1	1	0	0.466	1028.2	249.37	1.000	2.31	38
0	1	1	1	0	0	0	1	1	1	0	0.444	1087.6	236.87	1.500	2.48	38836
0	0	1	1	0	0	1	1	1	1	0	0.460	1918.3	248.67	1.000	2.30	91
0	1	1	1	0	0	1	1	1	1	0	0.438	1989.8	236.17	1.552	2.49	93002
1	0	1	1	0	0	0	1	1	1	0	0.457	1863.9	247.52	1.000	2.29	91
1	1	1	1	0	0	0	1	1	1	0	0.436	1981.6	235.01	1.425	2.42	93002
0	0	1	1	1	0	0	1	1	1	0	0.468	1319.8	237.54	1.000	2.27	874
0	1	1	1	1	0	0	1	1	1	0	0.450	1144.3	229.57	1.500	2.46	254828
0	0	1	1	1	0	1	1	1	1	0	0.461	2464.7	237.52	1.000	2.26	1941
0	1	1	1	1	0	1	1	1	1	0	0.443	2105.7	229.60	1.545	2.47	561302

Continued on Next Page...

Table 22 – Continued

S1	S2	S3	S4	S5	S6	S7	S8	S9	S10	S11	M1	M2	M3	M4	OEC	#A
1	0	1	1	1	0	0	1	1	1	0	0.460	2277.6	236.49	1.000	2.25	1941
1	1	1	1	1	0	0	1	1	1	0	0.442	2018.4	228.57	1.418	2.40	561302
1	0	1	1	0	0	1	1	1	1	0	0.458	2647.1	246.87	1.000	2.29	72
1	1	1	1	0	0	1	1	1	1	0	0.436	2766.8	234.37	1.439	2.42	73584
1	0	1	1	1	0	1	1	1	1	0	0.458	3231.3	236.73	1.000	2.25	1368
1	1	1	1	1	0	1	1	1	1	0	0.440	2817.3	228.87	1.433	2.41	390096
0	0	1	0	0	0	0	1	0	1	0	0.453	1386.0	245.00	1.000	2.27	16
0	1	1	0	0	0	0	1	0	1	0	0.431	1411.9	232.50	1.500	2.44	16352
0	0	1	0	1	0	0	1	0	1	0	0.455	1961.2	233.17	1.000	2.23	368
0	1	1	0	1	0	0	1	0	1	0	0.437	1603.9	225.21	1.500	2.42	107296
0	0	1	0	0	0	0	0	0	0	1	0.391	2880.0	260.00	1.000	2.22	1
0	1	1	0	0	0	0	0	0	0	1	0.372	3118.6	247.50	1.000	2.14	1022
0	0	1	0	0	0	1	0	0	0	1	0.384	4255.0	247.61	1.000	2.16	31
0	1	1	0	0	0	1	0	0	0	1	0.366	4312.4	235.11	1.000	2.08	31682
1	0	1	0	0	0	0	0	0	0	1	0.382	3860.1	244.52	1.000	2.14	31

Continued on Next Page...

Table 22 – Continued

S1	S2	S3	S4	S5	S6	S7	S8	S9	S10	S11	M1	M2	M3	M4	OEC	#A
1	1	1	0	0	0	0	0	0	0	1	0.363	4163.8	232.02	1.000	2.06	31682
0	0	1	0	0	1	0	0	0	0	1	0.447	2071.4	244.25	1.000	2.26	63
0	1	1	0	0	1	0	0	0	0	1	0.426	2122.0	231.75	1.000	2.17	64386
0	0	1	0	1	0	0	0	0	0	1	0.397	3228.4	249.68	1.000	2.19	31
0	1	1	0	1	0	0	0	0	0	1	0.381	2994.0	241.96	1.278	2.27	9282
0	0	1	0	1	0	1	0	0	0	1	0.390	5452.1	237.29	1.000	2.13	961
0	1	1	0	1	0	1	0	0	0	1	0.375	4583.6	229.57	1.294	2.22	287742
1	0	1	0	1	0	0	0	0	0	1	0.388	4338.3	234.19	1.000	2.11	961
1	1	1	0	1	0	0	0	0	0	1	0.373	4001.6	226.47	1.151	2.13	287742
0	0	1	0	1	1	0	0	0	0	1	0.454	2665.3	234.07	1.000	2.23	1937
0	1	1	0	1	1	0	0	0	0	1	0.436	2266.3	226.35	1.292	2.32	579614
1	0	1	0	0	0	1	0	0	0	1	0.381	4631.0	240.50	1.000	2.13	180
1	1	1	0	0	0	1	0	0	0	1	0.363	4770.8	228.00	1.000	2.04	183960
0	0	1	0	0	1	1	0	0	0	1	0.426	2975.8	241.15	1.000	2.21	391
0	1	1	0	0	1	1	0	0	0	1	0.406	2976.2	228.65	1.000	2.12	399602

Continued on Next Page...

Table 22 – Continued

S1	S2	S3	S4	S5	S6	S7	S8	S9	S10	S11	M1	M2	M3	M4	OEC	#A
1	0	1	0	1	0	1	0	0	0	1	0.387	5738.0	230.18	1.000	2.10	5580
1	1	1	0	1	0	1	0	0	0	1	0.372	4931.9	222.45	1.202	2.14	1670760
0	0	1	0	1	1	1	0	0	0	1	0.432	3977.5	231.21	1.000	2.18	11625
0	1	1	0	1	1	1	0	0	0	1	0.415	3283.8	223.64	1.282	2.26	3469550
1	0	1	0	0	1	0	0	0	0	1	0.426	2881.8	238.91	1.000	2.20	391
1	1	1	0	0	1	0	0	0	0	1	0.406	2997.6	226.40	1.000	2.12	399602
1	0	1	0	1	1	0	0	0	0	1	0.432	3579.5	229.01	1.000	2.17	11625
1	1	1	0	1	1	0	0	0	0	1	0.415	3110.1	221.83	1.196	2.21	3469550
0	0	1	0	0	0	0	0	1	0	1	0.377	2112.0	256.27	1.000	2.18	15
0	1	1	0	0	0	0	0	1	0	1	0.359	2267.1	243.77	1.000	2.10	15330
0	0	1	0	0	0	1	0	1	0	1	0.378	3075.8	248.33	1.000	2.15	115
0	1	1	0	0	0	1	0	1	0	1	0.360	3188.8	235.84	1.000	2.07	117530
1	0	1	0	0	0	0	0	1	0	1	0.368	2944.5	245.41	1.000	2.12	115
1	1	1	0	0	0	0	0	1	0	1	0.350	3164.8	232.91	1.000	2.04	117530
0	0	1	0	0	1	0	0	1	0	1	0.416	1673.0	245.98	1.000	2.21	245

Continued on Next Page...

Table 22 – Continued

S1	S2	S3	S4	S5	S6	S7	S8	S9	S10	S11	M1	M2	M3	M4	OEC	#A
0	1	1	0	0	1	0	0	1	0	1	0.397	1746.8	233.49	1.000	2.13	250390
0	0	1	0	1	0	0	0	1	0	1	0.383	2367.5	245.94	1.000	2.15	465
0	1	1	0	1	0	0	0	1	0	1	0.368	2176.6	238.21	1.311	2.25	139230
0	0	1	0	1	0	1	0	1	0	1	0.384	3766.6	238.01	1.000	2.12	3565
0	1	1	0	1	0	1	0	1	0	1	0.369	3263.9	230.30	1.308	2.22	1067430
1	0	1	0	1	0	0	0	1	0	1	0.374	3309.8	235.09	1.000	2.09	3565
1	1	1	0	1	0	0	0	1	0	1	0.359	3040.2	227.38	1.214	2.14	1067430
0	0	1	0	1	1	0	0	1	0	1	0.422	2079.8	235.97	1.000	2.18	7355
0	1	1	0	1	1	0	0	1	0	1	0.406	1817.1	228.29	1.309	2.28	2196810
0	0	1	0	0	0	0	0	0	1	1	0.464	2148.6	249.84	1.000	2.31	63
0	1	1	0	0	0	0	0	0	1	1	0.442	2307.7	237.34	1.271	2.36	64386
0	0	1	0	0	0	1	0	0	1	1	0.441	3273.1	245.24	1.000	2.25	537
0	1	1	0	0	0	1	0	0	1	1	0.420	3378.4	232.75	1.278	2.31	548814
1	0	1	0	0	0	0	0	0	1	1	0.432	3099.1	242.35	1.000	2.23	537
1	1	1	0	0	0	0	0	0	1	1	0.412	3333.0	229.86	1.180	2.23	548814

Continued on Next Page...

Table 22 – Continued

S1	S2	S3	S4	S5	S6	S7	S8	S9	S10	S11	M1	M2	M3	M4	OEC	#A
0	0	1	0	0	1	0	0	0	1	1	0.480	1849.2	242.15	1.000	2.31	813
0	1	1	0	0	1	0	0	0	1	1	0.457	1919.5	229.67	1.277	2.36	830886
0	0	1	0	1	0	0	0	0	1	1	0.467	2736.6	238.99	1.000	2.28	1441
0	1	1	0	1	0	0	0	0	1	1	0.448	2397.1	231.12	1.368	2.40	419902
0	0	1	0	1	0	1	0	0	1	1	0.444	4336.4	234.54	1.000	2.22	11975
0	1	1	0	1	0	1	0	0	1	1	0.427	3644.9	227.20	1.385	2.35	3480050
1	0	1	0	1	0	0	0	0	1	1	0.435	3829.7	231.68	1.000	2.19	11975
1	1	1	0	1	0	0	0	0	1	1	0.418	3386.5	224.04	1.227	2.24	3480050
0	0	1	0	1	1	0	0	0	1	1	0.483	2475.0	231.52	1.000	2.28	17955
0	1	1	0	1	1	0	0	0	1	1	0.464	2087.2	224.49	1.322	2.38	5212410
1	0	1	0	0	1	1	0	0	0	1	0.415	3478.6	237.59	1.000	2.18	960
1	1	1	0	0	1	1	0	0	0	1	0.396	3537.1	225.11	1.000	2.09	981120
1	0	1	0	1	1	1	0	0	0	1	0.420	4508.5	227.91	1.000	2.15	26880
1	1	1	0	1	1	1	0	0	0	1	0.403	3794.9	220.78	1.338	2.26	7983360
1	0	1	0	0	0	1	0	1	0	1	0.373	3585.6	242.04	1.000	2.12	280

Continued on Next Page...

Table 22 – Continued

S1	S2	S3	S4	S5	S6	S7	S8	S9	S10	S11	M1	M2	M3	M4	OEC	#A
1	1	1	0	0	0	1	0	1	0	1	0.355	3743.9	229.54	1.000	2.04	286160
0	0	1	0	0	1	1	0	1	0	1	0.409	2337.0	243.52	1.000	2.19	675
0	1	1	0	0	1	1	0	1	0	1	0.390	2396.3	231.03	1.000	2.10	689850
1	0	1	0	1	0	1	0	1	0	1	0.379	4321.3	231.71	1.000	2.09	8680
1	1	1	0	1	0	1	0	1	0	1	0.364	3785.7	223.93	1.239	2.15	2598960
0	0	1	0	1	1	1	0	1	0	1	0.414	3016.4	233.70	1.000	2.16	19085
0	1	1	0	1	1	1	0	1	0	1	0.398	2577.6	226.49	1.303	2.25	5672870
1	0	1	0	0	0	1	0	0	1	1	0.425	3908.0	240.91	1.000	2.21	1530
1	1	1	0	0	0	1	0	0	1	1	0.404	4060.3	228.41	1.209	2.23	1563660
0	0	1	0	0	1	1	0	0	1	1	0.457	2661.0	241.35	1.000	2.27	2385
0	1	1	0	0	1	1	0	0	1	1	0.436	2699.3	228.88	1.293	2.33	2437470
1	0	1	0	1	0	1	0	0	1	1	0.428	5015.7	230.50	1.000	2.17	32710
1	1	1	0	1	0	1	0	0	1	1	0.410	4253.3	223.79	1.424	2.33	9461620
0	0	1	0	1	1	1	0	0	1	1	0.460	3611.3	231.00	1.000	2.23	50015
0	1	1	0	1	1	1	0	0	1	1	0.440	2897.5	223.68	1.484	2.41	1.443533e7

Continued on Next Page...

Table 22 – Continued

S1	S2	S3	S4	S5	S6	S7	S8	S9	S10	S11	M1	M2	M3	M4	OEC	#A
1	0	1	0	0	1	0	0	1	0	1	0.403	2327.5	241.29	1.000	2.17	675
1	1	1	0	0	1	0	0	1	0	1	0.384	2452.0	228.80	1.000	2.08	689850
1	0	1	0	1	1	0	0	1	0	1	0.408	2840.2	231.54	1.000	2.14	19085
1	1	1	0	1	1	0	0	1	0	1	0.392	2514.7	224.92	1.264	2.22	5672870
1	0	1	0	0	1	0	0	0	1	1	0.453	2632.0	239.23	1.000	2.25	2385
1	1	1	0	0	1	0	0	0	1	1	0.432	2752.5	226.73	1.196	2.26	2437470
1	0	1	0	1	1	0	0	0	1	1	0.456	3412.9	228.91	1.000	2.22	50015
1	1	1	0	1	1	0	0	0	1	1	0.444	2840.2	222.24	1.443	2.39	1.443533e7
0	0	1	0	0	0	0	0	1	1	1	0.433	1718.6	252.26	1.000	2.26	245
0	1	1	0	0	0	0	0	1	1	1	0.412	1840.9	239.76	1.299	2.33	250390
0	0	1	0	0	0	1	0	1	1	1	0.423	2578.7	247.94	1.000	2.23	960
0	1	1	0	0	0	1	0	1	1	1	0.403	2700.4	235.43	1.294	2.29	981120
1	0	1	0	0	0	0	0	1	1	1	0.408	2504.2	245.07	1.000	2.19	960
1	1	1	0	0	0	0	0	1	1	1	0.388	2688.9	232.57	1.229	2.22	981120
0	0	1	0	0	1	0	0	1	1	1	0.447	1579.8	245.39	1.000	2.26	1465

Continued on Next Page...

Table 22 – Continued

S1	S2	S3	S4	S5	S6	S7	S8	S9	S10	S11	M1	M2	M3	M4	OEC	#A
0	1	1	0	0	1	0	0	1	1	1	0.426	1660.9	232.88	1.300	2.33	1497230
0	0	1	0	1	0	0	0	1	1	1	0.436	2103.3	241.35	1.000	2.23	5515
0	1	1	0	1	0	0	0	1	1	1	0.419	1860.1	233.46	1.393	2.36	1604330
0	0	1	0	1	0	1	0	1	1	1	0.427	3275.7	237.38	1.000	2.20	20640
0	1	1	0	1	0	1	0	1	1	1	0.410	2830.7	230.34	1.349	2.31	5974080
1	0	1	0	1	0	0	0	1	1	1	0.411	3029.4	234.55	1.000	2.16	20640
1	1	1	0	1	0	0	0	1	1	1	0.395	2707.5	227.46	1.327	2.26	5974080
0	0	1	0	1	1	0	0	1	1	1	0.450	2036.2	234.90	1.000	2.23	30935
0	1	1	0	1	1	0	0	1	1	1	0.436	1752.5	228.53	1.474	2.42	8935570
1	0	1	0	0	1	1	0	0	1	1	0.441	3256.0	239.11	1.000	2.23	3166
1	1	1	0	0	1	1	0	0	1	1	0.420	3334.3	226.69	1.193	2.24	3235652
1	0	1	0	1	1	1	0	0	1	1	0.443	4310.6	229.16	1.000	2.20	61698
1	1	1	0	1	1	1	0	0	1	1	0.426	3533.7	221.44	1.447	2.36	1.6777216e7
1	0	1	0	0	0	1	0	1	1	1	0.408	3207.8	243.52	1.000	2.19	1388
1	1	1	0	0	0	1	0	1	1	1	0.389	3368.1	231.02	1.233	2.22	1418536

Continued on Next Page...

Table 22 – Continued

S1	S2	S3	S4	S5	S6	S7	S8	S9	S10	S11	M1	M2	M3	M4	OEC	#A
0	0	1	0	0	1	1	0	1	1	1	0.436	2211.0	244.51	1.000	2.24	2222
0	1	1	0	0	1	1	0	1	1	1	0.415	2283.5	232.00	1.312	2.31	2270884
1	0	1	0	1	0	1	0	1	1	1	0.411	4009.5	233.40	1.000	2.15	27924
1	1	1	0	1	0	1	0	1	1	1	0.397	3470.4	226.61	1.385	2.29	8019928
0	0	1	0	1	1	1	0	1	1	1	0.438	2901.2	234.43	1.000	2.21	43506
0	1	1	0	1	1	1	0	1	1	1	0.419	2436.9	227.69	1.486	2.39	1.2453532e7
1	0	1	0	0	1	0	0	1	1	1	0.427	2221.2	242.37	1.000	2.22	2222
1	1	1	0	0	1	0	0	1	1	1	0.406	2343.9	229.91	1.238	2.25	2270884
1	0	1	0	1	1	0	0	1	1	1	0.429	2813.2	232.33	1.000	2.18	43506
1	1	1	0	1	1	0	0	1	1	1	0.421	2403.1	226.22	1.438	2.36	1.2453532e7
1	0	1	0	0	1	1	0	1	0	1	0.400	2805.4	240.22	1.000	2.16	832
1	1	1	0	0	1	1	0	1	0	1	0.381	2909.3	227.71	1.000	2.08	850304
1	0	1	0	1	1	1	0	1	0	1	0.405	3560.5	230.61	1.000	2.13	21312
1	1	1	0	1	1	1	0	1	0	1	0.390	3068.5	223.50	1.287	2.22	6280064
1	0	1	0	0	1	1	0	1	1	1	0.421	2752.0	242.12	1.000	2.21	1476

Continued on Next Page...

Table 22 – Continued

S1	S2	S3	S4	S5	S6	S7	S8	S9	S10	S11	M1	M2	M3	M4	OEC	#A
1	1	1	0	0	1	1	0	1	1	1	0.401	2858.2	229.61	1.241	2.24	1508472
1	0	1	0	1	1	1	0	1	1	1	0.423	3568.1	232.64	1.000	2.17	25980
1	1	1	0	1	1	1	0	1	1	1	0.409	3053.3	225.92	1.344	2.29	7332360
0	0	1	1	0	0	0	0	0	0	1	0.483	2592.0	260.00	1.000	2.39	1
0	1	1	1	0	0	0	0	0	0	1	0.460	2821.6	247.50	1.000	2.30	1022
0	0	1	1	0	0	1	0	0	0	1	0.426	3945.6	247.20	1.000	2.23	15
0	1	1	1	0	0	1	0	0	0	1	0.406	4007.2	234.70	1.000	2.15	15330
1	0	1	1	0	0	0	0	0	0	1	0.451	3024.0	247.73	1.000	2.28	15
1	1	1	1	0	0	0	0	0	0	1	0.429	3300.3	235.23	1.000	2.19	15330
0	0	1	1	0	1	0	0	0	0	1	0.508	1614.8	248.65	1.000	2.39	31
0	1	1	1	0	1	0	0	0	0	1	0.484	1674.2	236.15	1.000	2.30	31682
0	0	1	1	1	0	0	0	0	0	1	0.491	2905.5	249.68	1.000	2.36	31
0	1	1	1	1	0	0	0	0	0	1	0.472	2708.8	241.96	1.340	2.47	9282
0	0	1	1	1	0	1	0	0	0	1	0.433	5069.3	236.88	1.000	2.21	465
0	1	1	1	1	0	1	0	0	0	1	0.417	4268.8	229.16	1.307	2.30	139230

Continued on Next Page...

Table 22 – Continued

S1	S2	S3	S4	S5	S6	S7	S8	S9	S10	S11	M1	M2	M3	M4	OEC	#A
1	0	1	1	1	0	0	0	0	0	1	0.458	3398.9	237.41	1.000	2.25	465
1	1	1	1	1	0	0	0	0	0	1	0.440	3172.0	229.71	1.158	2.27	139230
0	0	1	1	1	1	0	0	0	0	1	0.516	2097.0	238.53	1.000	2.37	945
0	1	1	1	1	1	0	0	0	0	1	0.496	1804.2	230.83	1.304	2.45	282590
1	0	1	1	0	0	1	0	0	0	1	0.423	3934.5	242.14	1.000	2.21	50
1	1	1	1	0	0	1	0	0	0	1	0.403	4080.4	229.64	1.000	2.12	51100
0	0	1	1	0	1	1	0	0	0	1	0.461	2511.7	243.60	1.000	2.28	115
0	1	1	1	0	1	1	0	0	0	1	0.439	2528.3	231.10	1.000	2.19	117530
1	0	1	1	1	0	1	0	0	0	1	0.430	4883.7	231.82	1.000	2.18	1550
1	1	1	1	1	0	1	0	0	0	1	0.413	4225.1	224.10	1.198	2.22	464100
0	0	1	1	1	1	1	0	0	0	1	0.468	3386.1	233.80	1.000	2.26	3325
0	1	1	1	1	1	1	0	0	0	1	0.450	2815.1	226.12	1.285	2.34	990150
1	0	1	1	0	1	0	0	0	0	1	0.482	2098.2	244.00	1.000	2.32	115
1	1	1	1	0	1	0	0	0	0	1	0.460	2220.2	231.50	1.000	2.23	117530
1	0	1	1	1	1	0	0	0	0	1	0.489	2642.5	234.19	1.000	2.30	3325

Continued on Next Page...

Table 22 – Continued

S1	S2	S3	S4	S5	S6	S7	S8	S9	S10	S11	M1	M2	M3	M4	OEC	#A
1	1	1	1	1	1	0	0	0	0	1	0.470	2335.3	226.49	1.175	2.32	990150
0	0	1	1	0	0	0	0	1	0	1	0.465	1789.7	254.86	1.000	2.33	7
0	1	1	1	0	0	0	0	1	0	1	0.443	1930.5	242.36	1.000	2.25	7154
0	0	1	1	0	0	1	0	1	0	1	0.436	2607.7	247.03	1.000	2.25	31
0	1	1	1	0	0	1	0	1	0	1	0.415	2728.6	234.53	1.000	2.17	31682
1	0	1	1	0	0	0	0	1	0	1	0.439	2345.8	246.19	1.000	2.25	31
1	1	1	1	0	0	0	0	1	0	1	0.418	2540.2	233.69	1.000	2.17	31682
0	0	1	1	0	1	0	0	1	0	1	0.487	1303.8	247.62	1.000	2.35	69
0	1	1	1	0	1	0	0	1	0	1	0.464	1377.7	235.13	1.000	2.26	70518
0	0	1	1	1	0	0	0	1	0	1	0.473	2006.2	244.53	1.000	2.31	217
0	1	1	1	1	0	0	0	1	0	1	0.454	1853.4	236.81	1.340	2.42	64974
0	0	1	1	1	0	1	0	1	0	1	0.443	3166.6	236.71	1.000	2.22	961
0	1	1	1	1	0	1	0	1	0	1	0.426	2774.5	228.98	1.320	2.32	287742
1	0	1	1	1	0	0	0	1	0	1	0.446	2637.4	235.87	1.000	2.23	961
1	1	1	1	1	0	0	0	1	0	1	0.429	2441.5	228.15	1.230	2.28	287742

Continued on Next Page...

Table 22 – Continued

S1	S2	S3	S4	S5	S6	S7	S8	S9	S10	S11	M1	M2	M3	M4	OEC	#A
0	0	1	1	1	1	0	0	1	0	1	0.494	1631.1	237.74	1.000	2.32	2027
0	1	1	1	1	1	0	0	1	0	1	0.475	1444.2	230.04	1.308	2.41	604394
0	0	1	1	0	0	0	0	0	1	1	0.513	1681.5	250.71	1.000	2.41	31
0	1	1	1	0	0	0	0	0	1	1	0.488	1825.1	238.21	1.290	2.46	31682
0	0	1	1	0	0	1	0	0	1	1	0.472	2785.2	245.64	1.000	2.31	161
0	1	1	1	0	0	1	0	0	1	1	0.450	2895.0	233.15	1.290	2.37	164542
1	0	1	1	0	0	0	0	0	1	1	0.481	2359.5	245.22	1.000	2.33	161
1	1	1	1	0	0	0	0	0	1	1	0.459	2570.4	232.72	1.183	2.33	164542
0	0	1	1	0	1	0	0	0	1	1	0.522	1385.4	245.72	1.000	2.40	245
0	1	1	1	0	1	0	0	0	1	1	0.497	1457.0	233.23	1.289	2.45	250390
0	0	1	1	1	0	0	0	0	1	1	0.516	2142.5	239.80	1.000	2.37	705
0	1	1	1	1	0	0	0	0	1	1	0.495	1892.8	231.94	1.386	2.50	205310
0	0	1	1	1	0	1	0	0	1	1	0.476	3672.8	235.03	1.000	2.28	3519
0	1	1	1	1	0	1	0	0	1	1	0.457	3107.2	227.19	1.379	2.40	1020418
1	0	1	1	1	0	0	0	0	1	1	0.485	2917.7	234.60	1.000	2.29	3519

Continued on Next Page...

Table 22 – Continued

S1	S2	S3	S4	S5	S6	S7	S8	S9	S10	S11	M1	M2	M3	M4	OEC	#A
1	1	1	1	1	0	0	0	0	1	1	0.466	2608.7	226.74	1.244	2.35	1020418
0	0	1	1	1	1	0	0	0	1	1	0.524	1861.6	235.11	1.000	2.37	5275
0	1	1	1	1	1	0	0	0	1	1	0.503	1593.2	227.25	1.374	2.49	1527050
1	0	1	1	0	1	1	0	0	0	1	0.452	2762.9	241.01	1.000	2.26	170
1	1	1	1	0	1	1	0	0	0	1	0.430	2845.2	228.50	1.000	2.17	173740
1	0	1	1	1	1	1	0	0	0	1	0.458	3620.7	231.45	1.000	2.23	4470
1	1	1	1	1	1	1	0	0	0	1	0.440	3088.5	223.75	1.190	2.26	1320340
1	0	1	1	0	0	1	0	1	0	1	0.425	2943.4	242.26	1.000	2.21	42
1	1	1	1	0	0	1	0	1	0	1	0.405	3101.1	229.76	1.000	2.13	42924
0	0	1	1	0	1	1	0	1	0	1	0.460	1871.2	244.50	1.000	2.29	115
0	1	1	1	0	1	1	0	1	0	1	0.438	1945.7	232.01	1.000	2.20	117530
1	0	1	1	1	0	1	0	1	0	1	0.432	3531.9	231.94	1.000	2.19	1302
1	1	1	1	1	0	1	0	1	0	1	0.415	3124.3	224.22	1.239	2.25	389844
0	0	1	1	1	1	1	0	1	0	1	0.466	2421.2	234.83	1.000	2.26	3069
0	1	1	1	1	1	1	0	1	0	1	0.448	2092.6	227.13	1.269	2.33	907718

Continued on Next Page...

Table 22 – Continued

S1	S2	S3	S4	S5	S6	S7	S8	S9	S10	S11	M1	M2	M3	M4	OEC	#A
1	0	1	1	0	0	1	0	0	1	1	0.459	3261.3	242.91	1.000	2.28	280
1	1	1	1	0	0	1	0	0	1	1	0.437	3417.2	230.42	1.205	2.29	286160
0	0	1	1	0	1	1	0	0	1	1	0.485	2163.8	243.91	1.000	2.33	445
0	1	1	1	0	1	1	0	0	1	1	0.462	2214.7	231.40	1.295	2.39	454790
1	0	1	1	1	0	1	0	0	1	1	0.463	4170.1	232.71	1.000	2.24	5736
1	1	1	1	1	0	1	0	0	1	1	0.445	3578.4	224.88	1.267	2.32	1650992
0	0	1	1	1	1	1	0	0	1	1	0.488	2936.8	233.74	1.000	2.30	8883
0	1	1	1	1	1	1	0	0	1	1	0.469	2451.3	225.95	1.367	2.41	2548826
1	0	1	1	0	1	0	0	1	0	1	0.466	1750.6	243.98	1.000	2.29	115
1	1	1	1	0	1	0	0	1	0	1	0.444	1871.9	231.48	1.000	2.21	117530
1	0	1	1	1	1	0	0	1	0	1	0.473	2161.3	234.36	1.000	2.27	3069
1	1	1	1	1	1	0	0	1	0	1	0.454	1939.8	226.67	1.208	2.31	907718
1	0	1	1	0	1	0	0	0	1	1	0.498	1945.4	243.80	1.000	2.35	445
1	1	1	1	0	1	0	0	0	1	1	0.474	2063.3	231.31	1.199	2.36	454790
1	0	1	1	1	1	0	0	0	1	1	0.500	2538.4	233.62	1.000	2.32	8883

Continued on Next Page...

Table 22 – Continued

S1	S2	S3	S4	S5	S6	S7	S8	S9	S10	S11	M1	M2	M3	M4	OEC	#A
1	1	1	1	1	1	0	0	0	1	1	0.481	2201.1	225.83	1.252	2.38	2548826
0	0	1	1	0	0	0	0	1	1	1	0.498	1354.4	252.17	1.000	2.38	69
0	1	1	1	0	0	0	0	1	1	1	0.474	1461.3	239.67	1.326	2.46	70518
0	0	1	1	0	0	1	0	1	1	1	0.475	2120.0	247.77	1.000	2.33	170
0	1	1	1	0	0	1	0	1	1	1	0.452	2242.1	235.27	1.311	2.39	173740
1	0	1	1	0	0	0	0	1	1	1	0.467	1963.5	246.39	1.000	2.31	170
1	1	1	1	0	0	0	0	1	1	1	0.445	2125.9	233.89	1.240	2.34	173740
0	0	1	1	0	1	0	0	1	1	1	0.504	1221.8	247.30	1.000	2.38	261
0	1	1	1	0	1	0	0	1	1	1	0.480	1300.7	234.79	1.324	2.45	266742
0	0	1	1	1	0	0	0	1	1	1	0.501	1644.3	241.23	1.000	2.35	1531
0	1	1	1	1	0	0	0	1	1	1	0.482	1464.3	233.36	1.411	2.49	444682
0	0	1	1	1	0	1	0	1	1	1	0.478	2660.8	237.43	1.000	2.29	3510
0	1	1	1	1	0	1	0	1	1	1	0.460	2312.4	229.57	1.386	2.42	1011220
1	0	1	1	1	0	0	0	1	1	1	0.471	2369.6	236.06	1.000	2.27	3510
1	1	1	1	1	0	0	0	1	1	1	0.453	2123.2	228.22	1.300	2.36	1011220

Continued on Next Page...

Table 22 – Continued

S1	S2	S3	S4	S5	S6	S7	S8	S9	S10	S11	M1	M2	M3	M4	OEC	#A
0	0	1	1	1	1	0	0	1	1	1	0.507	1569.9	236.95	1.000	2.34	5259
0	1	1	1	1	1	0	0	1	1	1	0.487	1371.4	229.10	1.391	2.47	1510698
1	0	1	1	0	1	1	0	0	1	1	0.472	2613.3	242.63	1.000	2.30	336
1	1	1	1	0	1	1	0	0	1	1	0.450	2701.5	230.13	1.219	2.32	343392
1	0	1	1	1	1	1	0	0	1	1	0.475	3476.7	233.08	1.000	2.27	6000
1	1	1	1	1	1	1	0	0	1	1	0.456	2942.2	225.26	1.262	2.34	1696800
1	0	1	1	0	0	1	0	1	1	1	0.457	2676.2	244.45	1.000	2.28	138
1	1	1	1	0	0	1	0	1	1	1	0.436	2833.1	231.94	1.235	2.31	141036
0	0	1	1	0	1	1	0	1	1	1	0.482	1775.2	246.07	1.000	2.33	228
0	1	1	1	0	1	1	0	1	1	1	0.459	1855.4	233.56	1.317	2.40	233016
1	0	1	1	1	0	1	0	1	1	1	0.460	3331.6	234.80	1.000	2.25	2550
1	1	1	1	1	0	1	0	1	1	1	0.442	2903.7	226.99	1.280	2.33	724500
0	0	1	1	1	1	1	0	1	1	1	0.484	2320.4	236.42	1.000	2.30	4092
0	1	1	1	1	1	1	0	1	1	1	0.465	1986.9	228.63	1.358	2.41	1158024
1	0	1	1	0	1	0	0	1	1	1	0.482	1708.8	245.29	1.000	2.33	228

Continued on Next Page...

Table 22 – Continued

S1	S2	S3	S4	S5	S6	S7	S8	S9	S10	S11	M1	M2	M3	M4	OEC	#A
1	1	1	1	0	1	0	0	1	1	1	0.459	1825.1	232.79	1.248	2.36	233016
1	0	1	1	1	1	0	0	1	1	1	0.484	2170.4	235.65	1.000	2.30	4092
1	1	1	1	1	1	0	0	1	1	1	0.465	1904.7	227.82	1.285	2.37	1158024
1	0	1	1	0	1	1	0	1	0	1	0.450	2183.3	242.24	1.000	2.26	78
1	1	1	1	0	1	1	0	1	0	1	0.428	2302.4	229.74	1.000	2.17	79716
1	0	1	1	1	1	1	0	1	0	1	0.454	2794.1	232.83	1.000	2.23	1746
1	1	1	1	1	1	1	0	1	0	1	0.436	2446.2	225.12	1.165	2.25	507612
1	0	1	1	0	1	1	0	1	1	1	0.466	2223.2	244.48	1.000	2.30	66
1	1	1	1	0	1	1	0	1	1	1	0.444	2333.3	231.99	1.248	2.33	67452
1	0	1	1	1	1	1	0	1	1	1	0.467	2906.6	235.95	1.000	2.26	990
1	1	1	1	1	1	1	0	1	1	1	0.448	2499.3	228.25	1.244	2.32	272580
0	0	1	0	1	1	1	0	0	0	0	0.458	3731.7	221.65	1.000	2.19	496
0	1	1	0	1	1	1	0	0	0	0	0.439	2968.3	214.14	1.529	2.39	159712
1	0	1	0	1	1	1	0	0	0	0	0.439	4303.7	221.33	1.000	2.16	2880
1	1	1	0	1	1	1	0	0	0	0	0.421	3609.3	213.84	1.350	2.27	927360

Continued on Next Page...

Table 22 – Continued

S1	S2	S3	S4	S5	S6	S7	S8	S9	S10	S11	M1	M2	M3	M4	OEC	#A
0	0	1	0	1	1	1	0	1	0	0	0.432	2521.3	227.95	1.000	2.17	1840
0	1	1	0	1	1	1	0	1	0	0	0.414	2141.4	220.46	1.555	2.39	592480
1	0	1	0	1	1	1	0	1	0	0	0.418	3147.5	226.51	1.000	2.14	4480
1	1	1	0	1	1	1	0	1	0	0	0.400	2765.1	219.02	1.418	2.29	1442560
0	0	1	0	1	1	1	0	0	1	0	0.486	3358.6	224.28	1.000	2.26	8060
0	1	1	0	1	1	1	0	0	1	0	0.467	2718.0	216.35	1.532	2.46	2402120
1	0	1	0	1	1	1	0	0	1	0	0.461	4288.1	224.68	1.000	2.21	21610
1	1	1	0	1	1	1	0	0	1	0	0.443	3559.8	217.41	1.381	2.34	6405420
0	0	1	0	1	1	1	0	1	1	0	0.456	2576.5	230.37	1.000	2.22	13815
0	1	1	0	1	1	1	0	1	1	0	0.438	2170.4	222.57	1.553	2.44	4094930
1	0	1	0	1	1	1	0	1	1	0	0.436	3380.3	230.20	1.000	2.19	17964
1	1	1	0	1	1	1	0	1	1	0	0.420	2882.1	222.51	1.423	2.34	5277608
0	0	1	0	0	1	1	0	0	1	0	0.484	2405.0	237.54	1.000	2.30	276
0	1	1	0	0	1	1	0	0	1	0	0.461	2405.6	225.03	1.532	2.48	282072
1	0	1	0	0	1	1	0	0	1	0	0.459	3207.3	237.71	1.000	2.26	790

Continued on Next Page...

Table 22 – Continued

S1	S2	S3	S4	S5	S6	S7	S8	S9	S10	S11	M1	M2	M3	M4	OEC	#A
1	1	1	0	0	1	1	0	0	1	0	0.437	3258.8	225.22	1.385	2.36	807380
0	0	1	0	0	1	1	0	1	1	0	0.453	1955.5	243.50	1.000	2.27	505
0	1	1	0	0	1	1	0	1	1	0	0.432	2010.6	231.00	1.557	2.46	516110
1	0	1	0	0	1	1	0	1	1	0	0.432	2627.2	242.88	1.000	2.23	724
1	1	1	0	0	1	1	0	1	1	0	0.412	2724.4	230.38	1.441	2.36	739928
0	0	1	1	1	1	1	0	0	0	0	0.472	3061.5	226.00	1.000	2.24	240
0	1	1	1	1	1	1	0	0	0	0	0.453	2463.4	218.51	1.530	2.44	77280
1	0	1	1	1	1	1	0	0	0	0	0.466	3289.9	226.38	1.000	2.23	800
1	1	1	1	1	1	1	0	0	0	0	0.447	2814.2	218.88	1.341	2.33	257600
0	0	1	1	1	1	1	0	1	0	0	0.476	1972.0	230.16	1.000	2.26	496
0	1	1	1	1	1	1	0	1	0	0	0.457	1717.0	222.68	1.558	2.47	159712
1	0	1	1	1	1	1	0	1	0	0	0.464	2407.2	229.55	1.000	2.24	672
1	1	1	1	1	1	1	0	1	0	0	0.445	2172.8	222.07	1.422	2.38	216384
0	0	1	1	1	1	1	0	0	1	0	0.496	2642.0	228.13	1.000	2.29	2364
0	1	1	1	1	1	1	0	0	1	0	0.477	2166.1	220.15	1.529	2.49	702408

Continued on Next Page...

Table 22 – Continued

S1	S2	S3	S4	S5	S6	S7	S8	S9	S10	S11	M1	M2	M3	M4	OEC	#A
1	0	1	1	1	1	1	0	0	1	0	0.483	3343.8	229.77	1.000	2.27	3726
1	1	1	1	1	1	1	0	0	1	0	0.464	2811.1	221.77	1.380	2.40	1097572
0	0	1	1	1	1	1	0	1	1	0	0.495	2016.5	233.24	1.000	2.31	2325
0	1	1	1	1	1	1	0	1	1	0	0.475	1724.1	225.23	1.553	2.52	684950
1	0	1	1	1	1	1	0	1	1	0	0.477	2663.2	234.58	1.000	2.28	1560
1	1	1	1	1	1	1	0	1	1	0	0.458	2309.0	226.58	1.438	2.43	451920
0	0	1	1	0	1	1	0	0	1	0	0.494	1907.2	241.36	1.000	2.34	84
0	1	1	1	0	1	1	0	0	1	0	0.471	1928.2	228.85	1.533	2.51	85848
1	0	1	1	0	1	1	0	0	1	0	0.480	2500.7	242.59	1.000	2.31	146
1	1	1	1	0	1	1	0	0	1	0	0.457	2574.9	230.10	1.375	2.41	149212
0	0	1	1	0	1	1	0	1	1	0	0.492	1552.2	246.12	1.000	2.35	91
0	1	1	1	0	1	1	0	1	1	0	0.468	1619.2	233.62	1.561	2.54	93002
1	0	1	1	0	1	1	0	1	1	0	0.473	2074.3	246.49	1.000	2.32	72
1	1	1	1	0	1	1	0	1	1	0	0.451	2186.7	233.99	1.439	2.45	73584
0	0	1	0	1	0	1	0	0	1	0	0.478	4648.8	225.58	1.000	2.25	496

Continued on Next Page...

Table 22 – Continued

S1	S2	S3	S4	S5	S6	S7	S8	S9	S10	S11	M1	M2	M3	M4	OEC	#A
0	1	1	0	1	0	1	0	0	1	0	0.458	3792.0	217.64	1.529	2.44	148512
1	0	1	0	1	0	1	0	0	1	0	0.451	5431.9	224.12	1.000	2.19	4030
1	1	1	0	1	0	1	0	0	1	0	0.433	4573.6	216.16	1.350	2.30	1206660
0	0	1	0	1	0	1	0	1	1	0	0.451	3055.1	232.23	1.000	2.22	2015
0	1	1	0	1	0	1	0	1	1	0	0.433	2607.4	224.28	1.555	2.44	603330
1	0	1	0	1	0	1	0	1	1	0	0.430	3991.5	229.47	1.000	2.17	6820
1	1	1	0	1	0	1	0	1	1	0	0.413	3460.6	221.51	1.425	2.32	2042040
0	0	1	0	0	0	1	0	0	1	0	0.478	3307.5	239.00	1.000	2.30	16
0	1	1	0	0	0	1	0	0	1	0	0.455	3357.7	226.50	1.533	2.47	16352
1	0	1	0	0	0	1	0	0	1	0	0.451	4134.1	237.54	1.000	2.24	130
1	1	1	0	0	0	1	0	0	1	0	0.430	4260.2	225.03	1.355	2.33	132860
0	0	1	0	0	0	1	0	1	1	0	0.451	2361.0	245.65	1.000	2.27	65
0	1	1	0	0	0	1	0	1	1	0	0.430	2462.1	233.15	1.563	2.47	66430
1	0	1	0	0	0	1	0	1	1	0	0.430	3206.7	242.89	1.000	2.22	220
1	1	1	0	0	0	1	0	1	1	0	0.410	3356.0	230.41	1.431	2.35	224840

Continued on Next Page...

Table 22 – Continued

S1	S2	S3	S4	S5	S6	S7	S8	S9	S10	S11	M1	M2	M3	M4	OEC	#A
0	0	1	1	1	0	1	0	0	1	0	0.483	3818.6	226.58	1.000	2.26	248
0	1	1	1	1	0	1	0	0	1	0	0.463	3138.1	218.62	1.528	2.46	74256
1	0	1	1	1	0	1	0	0	1	0	0.474	4390.1	227.37	1.000	2.25	1178
1	1	1	1	1	0	1	0	0	1	0	0.455	3734.8	219.43	1.345	2.35	352716
0	0	1	1	1	0	1	0	1	1	0	0.494	2382.9	232.74	1.000	2.30	589
0	1	1	1	1	0	1	0	1	1	0	0.475	2066.3	224.78	1.558	2.52	176358
1	0	1	1	1	0	1	0	1	1	0	0.478	3232.9	231.66	1.000	2.27	1116
1	1	1	1	1	0	1	0	1	1	0	0.459	2845.6	223.71	1.427	2.42	334152
0	0	1	1	0	0	1	0	0	1	0	0.482	2716.9	240.00	1.000	2.31	8
0	1	1	1	0	0	1	0	0	1	0	0.460	2778.8	227.50	1.531	2.49	8176
1	0	1	1	0	0	1	0	0	1	0	0.474	3351.0	240.79	1.000	2.30	38
1	1	1	1	0	0	1	0	0	1	0	0.451	3484.2	228.29	1.349	2.38	38836
0	0	1	1	0	0	1	0	1	1	0	0.494	1873.4	246.16	1.000	2.36	19
0	1	1	1	0	0	1	0	1	1	0	0.471	1976.7	233.66	1.566	2.55	19418
1	0	1	1	0	0	1	0	1	1	0	0.478	2638.3	245.08	1.000	2.32	36

Continued on Next Page...

Table 22 – Continued

S1	S2	S3	S4	S5	S6	S7	S8	S9	S10	S11	M1	M2	M3	M4	OEC	#A
1	1	1	1	0	0	1	0	1	1	0	0.455	2789.6	232.58	1.434	2.45	36792
1	0	1	0	1	1	0	0	0	0	0	0.461	3223.1	218.55	1.000	2.19	496
1	1	1	0	1	1	0	0	0	0	0	0.442	2784.2	211.05	1.242	2.25	159712
1	0	1	0	1	1	0	0	1	0	0	0.422	2312.9	225.03	1.000	2.14	1840
1	1	1	0	1	1	0	0	1	0	0	0.405	2089.3	217.53	1.379	2.27	592480
1	0	1	0	1	1	0	0	0	1	0	0.479	3192.7	221.45	1.000	2.23	8060
1	1	1	0	1	1	0	0	0	1	0	0.460	2703.6	213.52	1.339	2.34	2402120
1	0	1	0	1	1	0	0	1	1	0	0.441	2506.1	227.62	1.000	2.19	13815
1	1	1	0	1	1	0	0	1	1	0	0.424	2179.9	219.98	1.414	2.33	4094930
1	0	1	0	0	1	0	0	0	1	0	0.478	2425.6	234.70	1.000	2.28	276
1	1	1	0	0	1	0	0	0	1	0	0.456	2517.2	222.20	1.319	2.35	282072
1	0	1	0	0	1	0	0	1	1	0	0.439	1990.6	240.73	1.000	2.23	505
1	1	1	0	0	1	0	0	1	1	0	0.418	2097.7	228.24	1.416	2.35	516110
1	0	1	1	1	1	0	0	0	0	0	0.504	2072.7	226.53	1.000	2.30	240
1	1	1	1	1	1	0	0	0	0	0	0.483	1844.9	219.05	1.233	2.35	77280

Continued on Next Page...

Table 22 – Continued

S1	S2	S3	S4	S5	S6	S7	S8	S9	S10	S11	M1	M2	M3	M4	OEC	#A
1	0	1	1	1	1	0	0	1	0	0	0.481	1665.6	229.32	1.000	2.27	496
1	1	1	1	1	1	0	0	1	0	0	0.461	1541.4	221.81	1.389	2.39	159712
1	0	1	1	1	1	0	0	0	1	0	0.509	2235.7	227.77	1.000	2.31	2364
1	1	1	1	1	1	0	0	0	1	0	0.488	1926.7	219.79	1.318	2.40	702408
1	0	1	1	1	1	0	0	1	1	0	0.490	1872.0	232.02	1.000	2.29	2325
1	1	1	1	1	1	0	0	1	1	0	0.470	1651.7	224.00	1.425	2.44	684950
1	0	1	1	0	1	0	0	0	1	0	0.508	1686.4	241.07	1.000	2.36	84
1	1	1	1	0	1	0	0	0	1	0	0.484	1783.4	228.57	1.310	2.42	85848
1	0	1	1	0	1	0	0	1	1	0	0.488	1487.7	244.97	1.000	2.34	91
1	1	1	1	0	1	0	0	1	1	0	0.465	1591.9	232.47	1.425	2.46	93002
1	0	1	0	1	0	0	0	0	1	0	0.453	3878.9	220.58	1.000	2.18	496
1	1	1	0	1	0	0	0	0	1	0	0.435	3412.0	212.63	1.250	2.24	148512
1	0	1	0	1	0	0	0	1	1	0	0.416	2716.5	227.64	1.000	2.14	2015
1	1	1	0	1	0	0	0	1	1	0	0.399	2430.9	219.67	1.399	2.28	603330
1	0	1	0	0	0	0	0	0	1	0	0.453	3078.0	234.00	1.000	2.23	16

Continued on Next Page...

Table 22 – Continued

S1	S2	S3	S4	S5	S6	S7	S8	S9	S10	S11	M1	M2	M3	M4	OEC	#A
1	1	1	0	0	0	0	0	0	1	0	0.432	3307.6	221.50	1.250	2.27	16352
1	0	1	0	0	0	0	0	1	1	0	0.416	2262.5	241.06	1.000	2.19	65
1	1	1	0	0	0	0	0	1	1	0	0.396	2429.8	228.55	1.406	2.31	66430
1	0	1	1	1	0	0	0	0	1	0	0.483	2729.6	225.08	1.000	2.25	248
1	1	1	1	1	0	0	0	0	1	0	0.463	2437.2	217.12	1.250	2.31	74256
1	0	1	1	1	0	0	0	1	1	0	0.469	2016.2	230.05	1.000	2.25	589
1	1	1	1	1	0	0	0	1	1	0	0.450	1826.4	222.11	1.421	2.39	176358
1	0	1	1	0	0	0	0	0	1	0	0.483	2166.0	238.50	1.000	2.30	8
1	1	1	1	0	0	0	0	0	1	0	0.460	2362.5	226.00	1.250	2.34	8176
1	0	1	1	0	0	0	0	1	1	0	0.469	1692.0	243.47	1.000	2.30	19
1	1	1	1	0	0	0	0	1	1	0	0.447	1834.1	230.97	1.428	2.42	19418
0	0	1	0	1	1	0	0	0	0	0	0.517	1316.3	217.00	1.000	2.28	16
0	1	1	0	1	1	0	0	0	0	0	0.496	1052.6	209.51	1.500	2.47	5152
0	0	1	0	1	1	0	0	1	0	0	0.446	1160.2	226.07	1.000	2.19	240
0	1	1	0	1	1	0	0	1	0	0	0.428	1005.7	218.58	1.560	2.41	77280

Continued on Next Page...

Table 22 – Continued

S1	S2	S3	S4	S5	S6	S7	S8	S9	S10	S11	M1	M2	M3	M4	OEC	#A
0	0	1	0	1	1	0	0	0	1	0	0.526	1530.9	219.62	1.000	2.31	976
0	1	1	0	1	1	0	0	0	1	0	0.505	1258.6	211.67	1.500	2.49	291872
0	0	1	0	1	1	0	0	1	1	0	0.472	1413.9	227.12	1.000	2.24	3790
0	1	1	0	1	1	0	0	1	1	0	0.453	1209.5	219.15	1.546	2.45	1129380
0	0	1	0	0	1	0	0	0	1	0	0.526	1086.8	233.00	1.000	2.36	32
0	1	1	0	0	1	0	0	0	1	0	0.501	1111.5	220.50	1.500	2.52	32704
0	0	1	0	0	1	0	0	1	1	0	0.470	1074.4	240.48	1.000	2.29	130
0	1	1	0	0	1	0	0	1	1	0	0.448	1125.0	227.98	1.552	2.48	132860
0	0	1	1	1	1	0	0	0	0	0	0.544	877.5	226.00	1.000	2.37	16
0	1	1	1	1	1	0	0	0	0	0	0.522	720.2	218.51	1.500	2.55	5152
0	0	1	1	1	1	0	0	1	0	0	0.508	877.5	229.43	1.000	2.32	112
0	1	1	1	1	1	0	0	1	0	0	0.488	775.4	221.94	1.564	2.53	36064
0	0	1	1	1	1	0	0	0	1	0	0.539	1068.8	225.05	1.000	2.36	480
0	1	1	1	1	1	0	0	0	1	0	0.518	895.2	217.08	1.500	2.54	143360
0	0	1	1	1	1	0	0	1	1	0	0.518	1082.9	230.12	1.000	2.34	1066

Continued on Next Page...

Table 22 – Continued

S1	S2	S3	S4	S5	S6	S7	S8	S9	S10	S11	M1	M2	M3	M4	OEC	#A
0	1	1	1	1	1	0	0	1	1	0	0.497	942.1	222.13	1.551	2.54	316652
0	0	1	1	0	1	0	0	0	1	0	0.539	756.0	238.50	1.000	2.41	16
0	1	1	1	0	1	0	0	0	1	0	0.514	787.3	226.00	1.500	2.56	16352
0	0	1	1	0	1	0	0	1	1	0	0.517	824.0	243.47	1.000	2.39	38
0	1	1	1	0	1	0	0	1	1	0	0.492	875.5	230.97	1.559	2.57	38836
0	1	0	0	0	0	0	1	0	0	1	0.304	2416.1	233.45	1.516	2.21	31
0	1	0	0	0	0	1	1	0	0	1	0.328	3275.3	226.72	1.559	2.25	261
1	1	0	0	0	0	0	1	0	0	1	0.320	3603.8	223.16	1.352	2.12	261
0	1	0	0	0	1	0	1	0	0	1	0.362	1839.4	223.58	1.544	2.30	537
0	1	0	0	1	0	0	1	0	0	1	0.313	2088.6	229.94	1.681	2.30	705
0	1	0	0	1	0	1	1	0	0	1	0.338	2969.0	223.36	1.700	2.33	5755
1	1	0	0	1	0	0	1	0	0	1	0.330	2946.3	219.78	1.454	2.18	5755
0	1	0	0	1	1	0	1	0	0	1	0.372	1679.5	220.38	1.687	2.37	11735
1	1	0	0	0	0	1	1	0	0	1	0.332	4105.1	220.39	1.435	2.17	740
0	1	0	0	0	1	1	1	0	0	1	0.363	2515.7	221.77	1.577	2.31	1595

Continued on Next Page...

Table 22 – Continued

S1	S2	S3	S4	S5	S6	S7	S8	S9	S10	S11	M1	M2	M3	M4	OEC	#A
1	1	0	0	1	0	1	1	0	0	1	0.341	3526.3	217.23	1.523	2.22	15580
0	1	0	0	1	1	1	1	0	0	1	0.373	2340.9	218.91	1.692	2.37	32885
1	1	0	0	0	1	0	1	0	0	1	0.360	2730.9	219.27	1.404	2.21	1595
1	1	0	0	1	1	0	1	0	0	1	0.368	2352.9	216.41	1.495	2.26	32885
0	1	0	0	0	0	0	1	1	0	1	0.311	2114.4	231.99	1.544	2.24	169
0	1	0	0	0	0	1	1	1	0	1	0.325	2822.2	226.99	1.609	2.28	529
1	1	0	0	0	0	0	1	1	0	1	0.318	3004.1	224.30	1.439	2.17	529
0	1	0	0	0	1	0	1	1	0	1	0.350	1652.5	225.67	1.567	2.29	1135
0	1	0	0	1	0	0	1	1	0	1	0.320	1714.5	228.49	1.693	2.31	3767
0	1	0	0	1	0	1	1	1	0	1	0.334	2384.0	223.69	1.724	2.34	11247
1	1	0	0	1	0	0	1	1	0	1	0.327	2342.2	221.02	1.530	2.22	11247
0	1	0	0	1	1	0	1	1	0	1	0.359	1410.4	222.69	1.689	2.36	23649
0	1	0	0	0	0	0	1	0	1	1	0.375	2228.6	226.07	1.685	2.40	537
0	1	0	0	0	0	1	1	0	1	1	0.373	2959.5	223.19	1.703	2.40	1779
1	1	0	0	0	0	0	1	0	1	1	0.366	3179.7	220.41	1.506	2.27	1779

Continued on Next Page...

Table 22 – Continued

S1	S2	S3	S4	S5	S6	S7	S8	S9	S10	S11	M1	M2	M3	M4	OEC	#A
0	1	0	0	0	1	0	1	0	1	1	0.397	1814.5	220.58	1.700	2.43	2707
0	1	0	0	1	0	0	1	0	1	1	0.385	1970.5	223.03	1.762	2.45	10631
0	1	0	0	1	0	1	1	0	1	1	0.383	2701.5	220.45	1.765	2.43	33805
1	1	0	0	1	0	0	1	0	1	1	0.376	2692.8	217.71	1.556	2.31	33805
0	1	0	0	1	1	0	1	0	1	1	0.407	1669.6	217.93	1.761	2.47	51037
1	1	0	0	0	1	1	1	0	0	1	0.361	3212.1	218.48	1.458	2.23	2164
1	1	0	0	1	1	1	1	0	0	1	0.369	2840.0	216.03	1.528	2.27	41292
1	1	0	0	0	0	1	1	1	0	1	0.328	3519.7	221.79	1.531	2.22	712
0	1	0	0	0	1	1	1	1	0	1	0.354	2244.0	223.84	1.608	2.32	1631
1	1	0	0	1	0	1	1	1	0	1	0.337	2848.0	218.83	1.595	2.26	14136
0	1	0	0	1	1	1	1	1	0	1	0.362	1946.4	221.27	1.695	2.36	31377
1	1	0	0	0	0	1	1	0	1	1	0.367	3700.9	219.13	1.548	2.29	2550
0	1	0	0	0	1	1	1	0	1	1	0.390	2437.0	219.80	1.714	2.42	3965
1	1	0	0	1	0	1	1	0	1	1	0.376	3243.2	216.82	1.588	2.32	45834
0	1	0	0	1	1	1	1	0	1	1	0.399	2276.2	217.56	1.759	2.45	70515

Continued on Next Page...

Table 22 – Continued

S1	S2	S3	S4	S5	S6	S7	S8	S9	S10	S11	M1	M2	M3	M4	OEC	#A
1	1	0	0	0	1	0	1	1	0	1	0.349	2393.6	221.72	1.464	2.23	1631
1	1	0	0	1	1	0	1	1	0	1	0.357	1949.5	219.18	1.536	2.27	31377
1	1	0	0	0	1	0	1	0	1	1	0.386	2617.6	217.66	1.543	2.32	3965
1	1	0	0	1	1	0	1	0	1	1	0.395	2308.1	215.45	1.582	2.34	70515
0	1	0	0	0	0	0	1	1	1	1	0.361	1948.6	227.35	1.696	2.38	1135
0	1	0	0	0	0	1	1	1	1	1	0.362	2587.4	224.84	1.715	2.39	1824
1	1	0	0	0	0	0	1	1	1	1	0.353	2732.0	222.46	1.559	2.28	1824
0	1	0	0	0	1	0	1	1	1	1	0.378	1646.8	223.06	1.711	2.41	2803
0	1	0	0	1	0	0	1	1	1	1	0.370	1632.2	224.44	1.763	2.42	21809
0	1	0	0	1	0	1	1	1	1	1	0.371	2226.7	222.38	1.764	2.42	33024
1	1	0	0	1	0	0	1	1	1	1	0.363	2219.0	220.05	1.600	2.31	33024
0	1	0	0	1	1	0	1	1	1	1	0.388	1432.3	220.72	1.758	2.44	50205
1	1	0	0	0	1	1	1	0	1	1	0.382	3090.3	217.41	1.575	2.32	2670
1	1	0	0	1	1	1	1	0	1	1	0.391	2782.3	215.68	1.597	2.35	44082
1	1	0	0	0	0	1	1	1	1	1	0.357	3245.4	221.10	1.592	2.30	1272

Continued on Next Page...

Table 22 – Continued

S1	S2	S3	S4	S5	S6	S7	S8	S9	S10	S11	M1	M2	M3	M4	OEC	#A
0	1	0	0	0	1	1	1	1	1	1	0.376	2190.8	222.18	1.726	2.41	2016
1	1	0	0	1	0	1	1	1	1	1	0.365	2709.5	219.25	1.616	2.32	21288
0	1	0	0	1	1	1	1	1	1	1	0.385	1930.1	220.38	1.752	2.43	33408
1	1	0	0	0	1	0	1	1	1	1	0.370	2328.0	220.25	1.585	2.32	2016
1	1	0	0	1	1	0	1	1	1	1	0.379	1958.8	218.47	1.608	2.34	33408
1	1	0	0	0	1	1	1	1	0	1	0.352	2851.1	220.85	1.519	2.26	1080
1	1	0	0	1	1	1	1	1	0	1	0.360	2369.9	218.88	1.559	2.28	18696
1	1	0	0	0	1	1	1	1	1	1	0.370	2771.8	219.83	1.610	2.33	534
1	1	0	0	1	1	1	1	1	1	1	0.378	2373.6	218.76	1.610	2.34	8010
0	1	0	1	0	0	0	1	0	0	1	0.376	2206.5	233.45	1.516	2.35	31
0	1	0	1	0	0	1	1	0	0	1	0.361	3127.3	225.82	1.565	2.31	115
1	1	0	1	0	0	0	1	0	0	1	0.378	2963.5	226.22	1.330	2.23	115
0	1	0	1	0	1	0	1	0	0	1	0.414	1471.3	227.93	1.531	2.40	245
0	1	0	1	1	0	0	1	0	0	1	0.388	1907.4	229.94	1.681	2.44	705
0	1	0	1	1	0	1	1	0	0	1	0.371	2850.9	222.44	1.703	2.39	2525

Continued on Next Page...

Table 22 – Continued

S1	S2	S3	S4	S5	S6	S7	S8	S9	S10	S11	M1	M2	M3	M4	OEC	#A
1	1	0	1	1	0	0	1	0	0	1	0.389	2409.1	222.83	1.422	2.28	2525
0	1	0	1	1	1	0	1	0	0	1	0.425	1357.0	224.82	1.673	2.48	5275
1	1	0	1	0	0	1	1	0	0	1	0.367	3646.4	221.77	1.429	2.24	170
0	1	0	1	0	1	1	1	0	0	1	0.394	2206.6	223.87	1.564	2.37	390
1	1	0	1	1	0	1	1	0	0	1	0.377	3123.9	218.65	1.503	2.28	3510
0	1	0	1	1	1	1	1	0	0	1	0.404	2066.4	221.20	1.672	2.43	7770
1	1	0	1	0	1	0	1	0	0	1	0.410	2121.6	224.21	1.374	2.30	390
1	1	0	1	1	1	0	1	0	0	1	0.420	1833.6	221.52	1.455	2.35	7770
0	1	0	1	0	0	0	1	1	0	1	0.386	1774.3	230.22	1.551	2.37	69
0	1	0	1	0	0	1	1	1	0	1	0.377	2461.7	225.40	1.635	2.38	115
1	1	0	1	0	0	0	1	1	0	1	0.382	2460.2	224.88	1.452	2.29	115
0	1	0	1	0	1	0	1	1	0	1	0.414	1319.3	227.01	1.559	2.41	261
0	1	0	1	1	0	0	1	1	0	1	0.398	1408.6	226.70	1.696	2.45	1531
0	1	0	1	1	0	1	1	1	0	1	0.386	2018.3	222.12	1.737	2.43	2397
1	1	0	1	1	0	0	1	1	0	1	0.392	1879.1	221.63	1.532	2.34	2397

Continued on Next Page...

Table 22 – Continued

S1	S2	S3	S4	S5	S6	S7	S8	S9	S10	S11	M1	M2	M3	M4	OEC	#A
0	1	0	1	1	1	0	1	1	0	1	0.425	1108.9	224.20	1.672	2.48	5259
0	1	0	1	0	0	0	1	0	1	1	0.417	1793.5	226.86	1.686	2.48	245
0	1	0	1	0	0	1	1	0	1	1	0.400	2611.2	223.28	1.699	2.44	445
1	1	0	1	0	0	0	1	0	1	1	0.410	2543.3	223.18	1.481	2.35	445
0	1	0	1	0	1	0	1	0	1	1	0.434	1394.3	223.92	1.697	2.51	683
0	1	0	1	1	0	0	1	0	1	1	0.428	1585.6	223.88	1.758	2.53	4779
0	1	0	1	1	0	1	1	0	1	1	0.410	2380.1	220.75	1.755	2.48	8211
1	1	0	1	1	0	0	1	0	1	1	0.421	2147.5	220.64	1.525	2.38	8211
0	1	0	1	1	1	0	1	0	1	1	0.445	1290.2	221.43	1.751	2.54	12453
1	1	0	1	0	1	1	1	0	0	1	0.395	2714.0	221.57	1.435	2.30	276
1	1	0	1	1	1	1	1	0	0	1	0.404	2397.0	219.48	1.484	2.33	4908
1	1	0	1	0	0	1	1	1	0	1	0.377	3025.4	222.06	1.577	2.34	78
0	1	0	1	0	1	1	1	1	0	1	0.402	1898.8	224.49	1.606	2.41	198
1	1	0	1	1	0	1	1	1	0	1	0.386	2376.4	219.35	1.617	2.36	1458
0	1	0	1	1	1	1	1	1	0	1	0.412	1605.7	222.30	1.670	2.45	3546

Continued on Next Page...

Table 22 – Continued

S1	S2	S3	S4	S5	S6	S7	S8	S9	S10	S11	M1	M2	M3	M4	OEC	#A
1	1	0	1	0	0	1	1	0	1	1	0.398	3221.1	220.83	1.521	2.34	336
0	1	0	1	0	1	1	1	0	1	1	0.415	2040.0	221.93	1.708	2.47	534
1	1	0	1	1	0	1	1	0	1	1	0.408	2816.5	218.90	1.549	2.37	5712
0	1	0	1	1	1	1	1	0	1	1	0.425	1911.7	220.04	1.739	2.50	8970
1	1	0	1	0	1	0	1	1	0	1	0.409	1911.7	224.15	1.455	2.34	198
1	1	0	1	1	1	0	1	1	0	1	0.418	1528.4	221.98	1.507	2.38	3546
1	1	0	1	0	1	0	1	0	1	1	0.426	2029.1	221.92	1.517	2.39	534
1	1	0	1	1	1	0	1	0	1	1	0.436	1796.4	220.03	1.544	2.42	8970
0	1	0	1	0	0	0	1	1	1	1	0.420	1576.7	226.94	1.697	2.49	261
0	1	0	1	0	0	1	1	1	1	1	0.409	2203.7	224.32	1.711	2.47	228
1	1	0	1	0	0	0	1	1	1	1	0.410	2217.4	223.54	1.553	2.39	228
0	1	0	1	0	1	0	1	1	1	1	0.432	1294.7	224.55	1.712	2.51	354
0	1	0	1	1	0	0	1	1	1	1	0.432	1302.4	224.22	1.757	2.53	4875
0	1	0	1	1	0	1	1	1	1	1	0.419	1858.3	222.29	1.746	2.50	3900
1	1	0	1	1	0	0	1	1	1	1	0.420	1777.6	221.53	1.583	2.42	3900

Continued on Next Page...

Table 22 – Continued

S1	S2	S3	S4	S5	S6	S7	S8	S9	S10	S11	M1	M2	M3	M4	OEC	#A
0	1	0	1	1	1	0	1	1	1	1	0.442	1113.5	222.58	1.744	2.54	5982
1	1	0	1	0	1	1	1	0	1	1	0.412	2579.1	220.45	1.547	2.38	150
1	1	0	1	1	1	1	1	0	1	1	0.421	2333.0	219.39	1.547	2.39	2250
1	1	0	1	0	0	1	1	1	1	1	0.403	2798.0	221.73	1.576	2.38	66
0	1	0	1	0	1	1	1	1	1	1	0.419	1810.3	223.24	1.722	2.49	108
1	1	0	1	1	0	1	1	1	1	1	0.412	2301.4	220.66	1.576	2.39	990
0	1	0	1	1	1	1	1	1	1	1	0.429	1568.7	222.17	1.722	2.50	1620
1	1	0	1	0	1	0	1	1	1	1	0.422	1854.2	222.70	1.574	2.42	108
1	1	0	1	1	1	0	1	1	1	1	0.431	1543.9	221.64	1.574	2.43	1620
0	1	0	0	0	1	1	1	0	0	0	0.377	2063.4	219.72	2.000	2.54	146
1	1	0	0	0	1	1	1	0	0	0	0.370	3007.0	217.48	1.700	2.37	460
0	1	0	0	1	1	1	1	0	0	0	0.388	2006.5	214.74	2.000	2.54	4030
1	1	0	0	1	1	1	1	0	0	0	0.378	2673.6	213.31	1.715	2.37	11380
0	1	0	0	0	1	1	1	1	0	0	0.361	1884.8	223.84	2.000	2.52	322
1	1	0	0	0	1	1	1	1	0	0	0.358	2656.9	221.25	1.762	2.39	496

Continued on Next Page...

Table 22 – Continued

S1	S2	S3	S4	S5	S6	S7	S8	S9	S10	S11	M1	M2	M3	M4	OEC	#A
0	1	0	0	1	1	1	1	1	0	0	0.369	1649.5	219.38	2.000	2.52	8142
1	1	0	0	1	1	1	1	1	0	0	0.365	2173.1	217.75	1.771	2.39	10896
0	1	0	0	0	1	1	1	0	1	0	0.406	2127.3	217.14	2.000	2.58	1250
1	1	0	0	0	1	1	1	0	1	0	0.393	2983.5	215.78	1.761	2.43	1838
0	1	0	0	1	1	1	1	0	1	0	0.416	2065.0	213.56	2.000	2.59	25870
1	1	0	0	1	1	1	1	0	1	0	0.402	2738.7	212.82	1.766	2.44	35154
0	1	0	0	0	1	1	1	1	1	0	0.386	1940.1	221.15	2.000	2.56	1305
1	1	0	0	0	1	1	1	1	1	0	0.377	2662.0	219.46	1.804	2.44	936
0	1	0	0	1	1	1	1	1	1	0	0.395	1738.5	217.99	2.000	2.56	25239
1	1	0	0	1	1	1	1	1	1	0	0.386	2283.0	217.18	1.807	2.45	16248
1	1	0	0	0	1	0	1	0	0	0	0.372	2371.1	216.21	1.630	2.33	146
1	1	0	0	1	1	0	1	0	0	0	0.382	2090.2	211.18	1.648	2.34	4030
1	1	0	0	0	1	0	1	1	0	0	0.354	2065.1	221.11	1.720	2.36	322
1	1	0	0	1	1	0	1	1	0	0	0.362	1668.7	216.69	1.732	2.36	8142
1	1	0	0	0	1	0	1	0	1	0	0.400	2374.3	214.38	1.720	2.42	1250

Continued on Next Page...

Table 22 – Continued

S1	S2	S3	S4	S5	S6	S7	S8	S9	S10	S11	M1	M2	M3	M4	OEC	#A
1	1	0	0	1	1	0	1	0	1	0	0.410	2151.9	210.85	1.727	2.43	25870
1	1	0	0	0	1	0	1	1	1	0	0.377	2109.4	218.76	1.782	2.42	1305
1	1	0	0	1	1	0	1	1	1	0	0.387	1791.6	215.66	1.786	2.43	25239
0	1	0	0	0	1	0	1	0	0	0	0.386	971.3	217.50	2.000	2.55	16
0	1	0	0	1	1	0	1	0	0	0	0.398	973.6	211.72	2.000	2.55	480
0	1	0	0	0	1	0	1	1	0	0	0.357	1007.4	224.15	2.000	2.52	92
0	1	0	0	1	1	0	1	1	0	0	0.367	890.5	218.87	2.000	2.52	2612
0	1	0	0	0	1	0	1	0	1	0	0.421	1173.4	215.48	2.000	2.60	368
0	1	0	0	1	1	0	1	0	1	0	0.431	1138.1	211.44	2.000	2.61	8096
0	1	0	0	0	1	0	1	1	1	0	0.390	1182.1	220.46	2.000	2.56	790
0	1	0	0	1	1	0	1	1	1	0	0.400	1055.1	216.66	2.000	2.57	16634
0	1	0	1	0	1	0	1	0	0	0	0.406	693.8	226.50	2.000	2.62	16
0	1	0	1	0	1	1	1	0	0	0	0.388	1809.2	223.38	2.000	2.57	65
1	1	0	1	0	1	0	1	0	0	0	0.409	1712.8	223.80	1.585	2.41	65
0	1	0	1	1	1	0	1	0	0	0	0.419	695.9	220.72	2.000	2.62	480

Continued on Next Page...

Table 22 – Continued

S1	S2	S3	S4	S5	S6	S7	S8	S9	S10	S11	M1	M2	M3	M4	OEC	#A
0	1	0	1	1	1	1	1	0	0	0	0.399	1767.5	218.43	2.000	2.57	1775
1	1	0	1	1	1	0	1	0	0	0	0.419	1510.5	218.83	1.601	2.41	1775
1	1	0	1	0	1	1	1	0	0	0	0.395	2503.9	221.79	1.664	2.41	110
1	1	0	1	1	1	1	1	0	0	0	0.403	2215.2	217.81	1.677	2.42	2610
0	1	0	1	0	1	0	1	1	0	0	0.413	803.3	226.74	2.000	2.63	38
0	1	0	1	0	1	1	1	1	0	0	0.403	1616.1	225.21	2.000	2.61	73
1	1	0	1	0	1	0	1	1	0	0	0.408	1622.0	224.62	1.712	2.47	73
0	1	0	1	1	1	0	1	1	0	0	0.424	694.1	221.50	2.000	2.63	1066
0	1	0	1	1	1	1	1	1	0	0	0.412	1364.1	220.93	2.000	2.61	1767
1	1	0	1	1	1	0	1	1	0	0	0.418	1277.8	220.42	1.722	2.48	1767
0	1	0	1	0	1	0	1	0	1	0	0.434	847.4	220.67	2.000	2.65	168
0	1	0	1	0	1	1	1	0	1	0	0.416	1755.7	220.36	2.000	2.61	311
1	1	0	1	0	1	0	1	0	1	0	0.428	1775.1	220.26	1.688	2.48	311
0	1	0	1	1	1	0	1	0	1	0	0.445	824.1	216.61	2.000	2.65	3624
0	1	0	1	1	1	1	1	0	1	0	0.426	1703.3	216.96	2.000	2.62	6201

Continued on Next Page...

Table 22 – Continued

S1	S2	S3	S4	S5	S6	S7	S8	S9	S10	S11	M1	M2	M3	M4	OEC	#A
1	1	0	1	1	1	0	1	0	1	0	0.438	1617.6	216.86	1.696	2.49	6201
1	1	0	1	0	1	1	1	1	0	0	0.402	2225.6	223.48	1.750	2.47	60
1	1	0	1	1	1	1	1	1	0	0	0.410	1757.7	220.54	1.757	2.48	1188
1	1	0	1	0	1	1	1	0	1	0	0.414	2457.2	219.91	1.729	2.47	240
1	1	0	1	1	1	1	1	0	1	0	0.424	2257.8	217.43	1.734	2.48	4272
0	1	0	1	0	1	0	1	1	1	0	0.434	934.8	222.84	2.000	2.65	182
0	1	0	1	0	1	1	1	1	1	0	0.423	1610.1	223.02	2.000	2.64	162
1	1	0	1	0	1	0	1	1	1	0	0.423	1663.6	222.21	1.778	2.52	162
0	1	0	1	1	1	0	1	1	1	0	0.445	823.9	219.21	2.000	2.66	3690
0	1	0	1	1	1	1	1	1	1	0	0.433	1409.2	220.38	2.000	2.64	2910
1	1	0	1	1	1	0	1	1	1	0	0.434	1391.8	219.60	1.781	2.53	2910
1	1	0	1	0	1	1	1	1	0	1	0.401	2394.3	222.59	1.519	2.35	54
1	1	0	1	1	1	1	1	1	0	1	0.410	1936.6	221.53	1.519	2.36	810
1	1	0	1	0	1	1	1	1	1	0	0.417	2214.1	222.13	1.792	2.52	48
1	1	0	1	1	1	1	1	1	1	0	0.426	1849.6	221.06	1.792	2.53	720

Continued on Next Page...

Table 22 – Continued

S1	S2	S3	S4	S5	S6	S7	S8	S9	S10	S11	M1	M2	M3	M4	OEC	#A
0	1	0	0	0	0	1	1	0	1	0	0.397	2731.6	219.93	2.000	2.57	184
1	1	0	0	0	0	1	1	0	1	0	0.384	3780.1	216.83	1.719	2.40	606
0	1	0	0	1	0	1	1	0	1	0	0.407	2626.3	215.80	2.000	2.58	4104
1	1	0	0	1	0	1	1	0	1	0	0.393	3388.2	213.17	1.725	2.40	12898
0	1	0	0	0	0	1	1	1	1	0	0.377	2382.8	223.54	2.000	2.55	395
1	1	0	0	0	0	1	1	1	1	0	0.370	3272.3	220.35	1.778	2.42	616
0	1	0	0	1	0	1	1	1	1	0	0.385	2095.6	219.59	2.000	2.55	8565
1	1	0	0	1	0	1	1	1	1	0	0.378	2728.1	217.09	1.782	2.42	12408
1	1	0	0	0	0	0	1	0	1	0	0.385	3088.7	215.87	1.652	2.36	184
1	1	0	0	1	0	0	1	0	1	0	0.395	2696.8	211.76	1.657	2.37	4104
1	1	0	0	0	0	0	1	1	1	0	0.363	2580.9	220.39	1.744	2.39	395
1	1	0	0	1	0	0	1	1	1	0	0.373	2112.0	216.53	1.747	2.39	8565
0	1	0	0	0	0	0	1	1	1	0	0.373	1387.8	224.30	2.000	2.55	92
0	1	0	0	1	0	0	1	1	1	0	0.383	1192.2	219.78	2.000	2.55	2116
0	1	0	1	0	0	0	1	0	1	0	0.410	1045.0	222.00	2.000	2.61	16

Continued on Next Page...

Table 22 – Continued

S1	S2	S3	S4	S5	S6	S7	S8	S9	S10	S11	M1	M2	M3	M4	OEC	#A
0	1	0	1	0	0	1	1	0	1	0	0.400	2387.4	220.74	2.000	2.58	84
1	1	0	1	0	0	0	1	0	1	0	0.412	2361.0	220.45	1.619	2.41	84
0	1	0	1	1	0	0	1	0	1	0	0.422	990.5	217.48	2.000	2.61	368
0	1	0	1	1	0	1	1	0	1	0	0.410	2300.1	216.64	2.000	2.59	1868
1	1	0	1	1	0	0	1	0	1	0	0.423	2057.3	216.39	1.623	2.42	1868
1	1	0	1	0	0	1	1	0	1	0	0.405	3278.4	219.73	1.685	2.43	146
1	1	0	1	1	0	1	1	0	1	0	0.415	2927.7	216.24	1.690	2.44	3054
0	1	0	1	0	0	0	1	1	1	0	0.422	1135.0	224.37	2.000	2.64	38
0	1	0	1	0	0	1	1	1	1	0	0.417	2040.5	223.67	2.000	2.63	91
1	1	0	1	0	0	0	1	1	1	0	0.415	2073.7	222.52	1.747	2.49	91
0	1	0	1	1	0	0	1	1	1	0	0.434	957.1	219.85	2.000	2.64	874
0	1	0	1	1	0	1	1	1	1	0	0.427	1742.8	219.85	2.000	2.63	1941
1	1	0	1	1	0	0	1	1	1	0	0.426	1662.8	218.82	1.748	2.50	1941
1	1	0	1	0	0	1	1	1	1	0	0.415	2852.9	221.87	1.764	2.50	72
1	1	0	1	1	0	1	1	1	1	0	0.425	2302.5	219.09	1.767	2.51	1368

Continued on Next Page...

Table 22 – Continued

S1	S2	S3	S4	S5	S6	S7	S8	S9	S10	S11	M1	M2	M3	M4	OEC	#A
0	1	0	0	0	0	0	1	0	1	0	0.410	1425.0	220.00	2.000	2.60	16
0	1	0	0	1	0	0	1	0	1	0	0.422	1350.7	215.48	2.000	2.60	368
0	1	0	0	0	0	0	0	0	0	1	0.354	3300.0	235.00	1.000	2.05	1
0	1	0	0	0	0	1	0	0	0	1	0.348	4321.9	222.61	1.000	2.00	31
1	1	0	0	0	0	0	0	0	0	1	0.346	4393.5	219.52	1.000	1.98	31
0	1	0	0	0	1	0	0	0	0	1	0.405	2147.6	219.25	1.000	2.09	63
0	1	0	0	1	0	0	0	0	0	1	0.368	2448.4	231.90	1.516	2.33	31
0	1	0	0	1	0	1	0	0	0	1	0.362	3778.0	219.52	1.516	2.27	961
1	1	0	0	1	0	0	0	0	0	1	0.359	3259.7	216.42	1.250	2.12	961
0	1	0	0	1	1	0	0	0	0	1	0.420	1885.3	216.30	1.512	2.36	1937
1	1	0	0	0	0	1	0	0	0	1	0.346	4850.7	215.50	1.000	1.96	180
0	1	0	0	0	1	1	0	0	0	1	0.386	2947.9	216.15	1.000	2.04	391
1	1	0	0	1	0	1	0	0	0	1	0.359	4048.4	212.40	1.330	2.14	5580
0	1	0	0	1	1	1	0	0	0	1	0.400	2731.6	213.45	1.495	2.30	11625
1	1	0	0	0	1	0	0	0	0	1	0.386	3075.6	213.91	1.000	2.03	391

Continued on Next Page...

Table 22 – Continued

S1	S2	S3	S4	S5	S6	S7	S8	S9	S10	S11	M1	M2	M3	M4	OEC	#A
1	1	0	0	1	1	0	0	0	0	1	0.400	2571.4	211.25	1.317	2.21	11625
0	1	0	0	0	0	0	0	1	0	1	0.342	2380.0	231.27	1.000	2.02	15
0	1	0	0	0	0	1	0	1	0	1	0.342	3260.7	223.33	1.000	1.99	115
1	1	0	0	0	0	0	0	1	0	1	0.333	3328.7	220.41	1.000	1.96	115
0	1	0	0	0	1	0	0	1	0	1	0.377	1797.9	220.98	1.000	2.04	245
0	1	0	0	1	0	0	0	1	0	1	0.355	1765.8	228.17	1.516	2.29	465
0	1	0	0	1	0	1	0	1	0	1	0.356	2675.1	220.23	1.516	2.26	3565
1	1	0	0	1	0	0	0	1	0	1	0.346	2469.7	217.31	1.346	2.14	3565
0	1	0	0	1	1	0	0	1	0	1	0.391	1497.7	218.21	1.500	2.31	7355
0	1	0	0	0	0	0	0	0	1	1	0.420	2423.8	224.84	1.508	2.39	63
0	1	0	0	0	0	1	0	0	1	1	0.400	3441.1	220.24	1.514	2.34	537
1	1	0	0	0	0	0	0	0	1	1	0.392	3508.6	217.35	1.328	2.22	537
0	1	0	0	0	1	0	0	0	1	1	0.435	1964.7	217.15	1.519	2.39	813
0	1	0	0	1	0	0	0	0	1	1	0.432	2004.2	221.30	1.677	2.48	1441
0	1	0	0	1	0	1	0	0	1	1	0.412	3038.5	216.86	1.673	2.43	11975

Continued on Next Page...

Table 22 – Continued

S1	S2	S3	S4	S5	S6	S7	S8	S9	S10	S11	M1	M2	M3	M4	OEC	#A
1	1	0	0	1	0	0	0	0	1	1	0.403	2814.8	213.99	1.436	2.28	11975
0	1	0	0	1	1	0	0	0	1	1	0.447	1752.1	213.84	1.673	2.48	17955
1	1	0	0	0	1	1	0	0	0	1	0.377	3559.1	212.59	1.000	2.01	960
1	1	0	0	1	1	1	0	0	0	1	0.389	3147.4	210.16	1.333	2.19	26880
1	1	0	0	0	0	1	0	1	0	1	0.338	3851.1	217.04	1.000	1.96	280
0	1	0	0	0	1	1	0	1	0	1	0.371	2430.3	218.52	1.000	2.02	675
1	1	0	0	1	0	1	0	1	0	1	0.351	3095.4	213.94	1.382	2.16	8680
0	1	0	0	1	1	1	0	1	0	1	0.384	2122.5	215.95	1.469	2.27	19085
1	1	0	0	0	0	1	0	0	1	1	0.385	4162.8	215.91	1.366	2.22	1530
0	1	0	0	0	1	1	0	0	1	1	0.415	2712.2	216.35	1.524	2.36	2385
1	1	0	0	1	0	1	0	0	1	1	0.396	3542.6	212.83	1.474	2.28	32710
0	1	0	0	1	1	1	0	0	1	1	0.426	2495.1	213.33	1.660	2.43	50015
1	1	0	0	0	1	0	0	1	0	1	0.365	2543.6	216.29	1.000	2.00	675
1	1	0	0	1	1	0	0	1	0	1	0.378	2065.2	213.78	1.347	2.19	19085
1	1	0	0	0	1	0	0	0	1	1	0.411	2835.5	214.23	1.370	2.26	2385

Continued on Next Page...

Table 22 – Continued

S1	S2	S3	S4	S5	S6	S7	S8	S9	S10	S11	M1	M2	M3	M4	OEC	#A
1	1	0	0	1	1	0	0	0	1	1	0.422	2434.6	211.24	1.473	2.33	50015
0	1	0	0	0	0	0	0	1	1	1	0.392	1928.6	227.26	1.531	2.36	245
0	1	0	0	0	0	1	0	1	1	1	0.384	2785.3	222.94	1.526	2.33	960
1	1	0	0	0	0	0	0	1	1	1	0.370	2825.5	220.07	1.392	2.22	960
0	1	0	0	0	1	0	0	1	1	1	0.405	1719.9	220.39	1.539	2.36	1465
0	1	0	0	1	0	0	0	1	1	1	0.404	1537.7	223.66	1.687	2.45	5515
0	1	0	0	1	0	1	0	1	1	1	0.395	2331.6	219.71	1.669	2.41	20640
1	1	0	0	1	0	0	0	1	1	1	0.381	2222.1	216.88	1.505	2.29	20640
0	1	0	0	1	1	0	0	1	1	1	0.417	1460.1	217.23	1.673	2.44	30935
1	1	0	0	0	1	1	0	0	1	1	0.399	3374.2	214.11	1.395	2.25	3166
1	1	0	0	1	1	1	0	0	1	1	0.410	3001.6	211.52	1.483	2.31	61698
1	1	0	0	0	0	1	0	1	1	1	0.370	3478.9	218.52	1.404	2.22	1388
0	1	0	0	0	1	1	0	1	1	1	0.395	2332.2	219.51	1.538	2.34	2222
1	1	0	0	1	0	1	0	1	1	1	0.381	2859.7	215.75	1.504	2.28	27924
0	1	0	0	1	1	1	0	1	1	1	0.406	2038.4	216.79	1.646	2.40	43506

Continued on Next Page...

Table 22 – Continued

S1	S2	S3	S4	S5	S6	S7	S8	S9	S10	S11	M1	M2	M3	M4	OEC	#A
1	1	0	0	0	1	0	0	1	1	1	0.387	2434.2	217.37	1.416	2.25	2222
1	1	0	0	1	1	0	0	1	1	1	0.398	2023.4	214.69	1.506	2.31	43506
1	1	0	0	0	1	1	0	1	0	1	0.363	2977.4	215.22	1.000	1.99	832
1	1	0	0	1	1	1	0	1	0	1	0.375	2531.5	212.88	1.324	2.17	21312
1	1	0	0	0	1	1	0	1	1	1	0.382	2928.0	217.12	1.427	2.25	1476
1	1	0	0	1	1	1	0	1	1	1	0.392	2514.1	215.04	1.486	2.29	25980
0	1	0	1	0	0	0	0	0	0	1	0.438	3000.0	235.00	1.000	2.21	1
0	1	0	1	0	0	1	0	0	0	1	0.387	4024.0	222.20	1.000	2.06	15
1	1	0	1	0	0	0	0	0	0	1	0.409	3520.0	222.73	1.000	2.11	15
0	1	0	1	0	1	0	0	0	0	1	0.461	1712.9	223.65	1.000	2.21	31
0	1	0	1	1	0	0	0	0	0	1	0.455	2225.8	231.90	1.516	2.49	31
0	1	0	1	1	0	1	0	0	0	1	0.401	3530.6	219.10	1.516	2.34	465
1	1	0	1	1	0	0	0	0	0	1	0.424	2611.6	219.64	1.241	2.24	465
0	1	0	1	1	1	0	0	0	0	1	0.478	1519.2	220.76	1.508	2.48	945
1	1	0	1	0	0	1	0	0	0	1	0.383	4174.2	217.14	1.000	2.04	50

Continued on Next Page...

Table 22 – Continued

S1	S2	S3	S4	S5	S6	S7	S8	S9	S10	S11	M1	M2	M3	M4	OEC	#A
0	1	0	1	0	1	1	0	0	0	1	0.418	2519.6	218.60	1.000	2.11	115
1	1	0	1	1	0	1	0	0	0	1	0.398	3492.4	214.04	1.320	2.21	1550
0	1	0	1	1	1	1	0	0	0	1	0.433	2359.3	216.04	1.481	2.37	3325
1	1	0	1	0	1	0	0	0	0	1	0.437	2312.6	219.00	1.000	2.15	115
1	1	0	1	1	1	0	0	0	0	1	0.453	1959.0	216.43	1.298	2.31	3325
0	1	0	1	0	0	0	0	1	0	1	0.422	2035.7	229.86	1.000	2.16	7
0	1	0	1	0	0	1	0	1	0	1	0.395	2813.7	222.03	1.000	2.08	31
1	1	0	1	0	0	0	0	1	0	1	0.398	2690.3	221.19	1.000	2.08	31
0	1	0	1	0	1	0	0	1	0	1	0.441	1433.9	222.62	1.000	2.17	69
0	1	0	1	1	0	0	0	1	0	1	0.438	1510.4	226.76	1.516	2.43	217
0	1	0	1	1	0	1	0	1	0	1	0.410	2281.3	218.94	1.516	2.35	961
1	1	0	1	1	0	0	0	1	0	1	0.413	1996.0	218.10	1.350	2.27	961
0	1	0	1	1	1	0	0	1	0	1	0.457	1198.6	219.97	1.489	2.43	2027
0	1	0	1	0	0	0	0	0	1	1	0.465	1935.5	225.71	1.516	2.48	31
0	1	0	1	0	0	1	0	0	1	1	0.428	2968.0	220.64	1.522	2.40	161

Continued on Next Page...

Table 22 – Continued

S1	S2	S3	S4	S5	S6	S7	S8	S9	S10	S11	M1	M2	M3	M4	OEC	#A
1	1	0	1	0	0	0	0	0	1	1	0.436	2736.9	220.22	1.323	2.31	161
0	1	0	1	0	1	0	0	0	1	1	0.473	1509.8	220.72	1.531	2.48	245
0	1	0	1	1	0	0	0	0	1	1	0.478	1600.0	222.11	1.681	2.57	705
0	1	0	1	1	0	1	0	0	1	1	0.441	2609.2	217.35	1.672	2.48	3519
1	1	0	1	1	0	0	0	0	1	1	0.449	2195.1	216.92	1.426	2.37	3519
0	1	0	1	1	1	0	0	0	1	1	0.485	1352.0	217.44	1.673	2.56	5275
1	1	0	1	0	1	1	0	0	0	1	0.410	2894.6	216.01	1.000	2.08	170
1	1	0	1	1	1	1	0	0	0	1	0.424	2585.0	213.72	1.301	2.25	4470
1	1	0	1	0	0	1	0	1	0	1	0.385	3216.5	217.26	1.000	2.04	42
0	1	0	1	0	1	1	0	1	0	1	0.417	1998.0	219.50	1.000	2.11	115
1	1	0	1	1	0	1	0	1	0	1	0.400	2568.7	214.17	1.381	2.25	1302
0	1	0	1	1	1	1	0	1	0	1	0.432	1738.6	217.09	1.438	2.35	3069
1	1	0	1	0	0	1	0	0	1	1	0.416	3527.7	217.91	1.357	2.28	280
0	1	0	1	0	1	1	0	0	1	1	0.440	2241.8	218.91	1.535	2.42	445
1	1	0	1	1	0	1	0	0	1	1	0.429	2992.4	215.05	1.454	2.34	5736

Continued on Next Page...

Table 22 – Continued

S1	S2	S3	S4	S5	S6	S7	S8	S9	S10	S11	M1	M2	M3	M4	OEC	#A
0	1	0	1	1	1	1	0	0	1	1	0.452	2065.1	216.09	1.650	2.49	8883
1	1	0	1	0	1	0	0	1	0	1	0.423	1967.7	218.98	1.000	2.12	115
1	1	0	1	1	1	0	0	1	0	1	0.438	1609.0	216.62	1.323	2.30	3069
1	1	0	1	0	1	0	0	0	1	1	0.451	2154.1	218.80	1.364	2.35	445
1	1	0	1	1	1	0	0	0	1	1	0.463	1860.8	215.97	1.454	2.41	8883
0	1	0	1	0	0	0	0	1	1	1	0.451	1541.3	227.17	1.551	2.48	69
0	1	0	1	0	0	1	0	1	1	1	0.430	2333.0	222.77	1.535	2.42	170
1	1	0	1	0	0	0	0	1	1	1	0.424	2251.5	221.39	1.400	2.33	170
0	1	0	1	0	1	0	0	1	1	1	0.457	1362.0	222.30	1.559	2.47	261
0	1	0	1	1	0	0	0	1	1	1	0.464	1218.6	223.54	1.696	2.56	1531
0	1	0	1	1	0	1	0	1	1	1	0.443	1923.8	219.77	1.662	2.49	3510
1	1	0	1	1	0	0	0	1	1	1	0.436	1764.4	218.40	1.505	2.39	3510
0	1	0	1	1	1	0	0	1	1	1	0.469	1148.9	219.30	1.672	2.54	5259
1	1	0	1	0	1	1	0	0	1	1	0.428	2759.3	217.63	1.387	2.32	336
1	1	0	1	1	1	1	0	0	1	1	0.440	2467.4	215.47	1.448	2.36	6000

Continued on Next Page...

Table 22 – Continued

S1	S2	S3	S4	S5	S6	S7	S8	S9	S10	S11	M1	M2	M3	M4	OEC	#A
1	1	0	1	0	0	1	0	1	1	1	0.414	2950.5	219.45	1.399	2.30	138
0	1	0	1	0	1	1	0	1	1	1	0.437	1914.1	221.07	1.553	2.43	228
1	1	0	1	1	0	1	0	1	1	1	0.427	2408.3	217.18	1.474	2.36	2550
0	1	0	1	1	1	1	0	1	1	1	0.448	1658.5	218.81	1.626	2.48	4092
1	1	0	1	0	1	0	0	1	1	1	0.437	1916.5	220.29	1.421	2.36	228
1	1	0	1	1	1	0	0	1	1	1	0.448	1591.5	218.04	1.485	2.40	4092
1	1	0	1	0	1	1	0	1	0	1	0.408	2393.0	217.24	1.000	2.08	78
1	1	0	1	1	1	1	0	1	0	1	0.421	2033.3	215.15	1.257	2.23	1746
1	1	0	1	0	1	1	0	1	1	1	0.423	2415.1	219.48	1.424	2.33	66
1	1	0	1	1	1	1	0	1	1	1	0.432	2078.7	218.42	1.424	2.35	990
0	1	0	0	1	1	1	0	0	0	0	0.424	2525.8	203.65	2.000	2.56	496
1	1	0	0	1	1	1	0	0	0	0	0.406	3021.4	203.33	1.639	2.35	2880
0	1	0	0	1	1	1	0	1	0	0	0.400	1788.4	209.95	2.000	2.54	1840
1	1	0	0	1	1	1	0	1	0	0	0.387	2284.8	208.51	1.739	2.38	4480
0	1	0	0	1	1	1	0	0	1	0	0.450	2295.8	206.52	2.000	2.62	8060

Continued on Next Page...

Table 22 – Continued

S1	S2	S3	S4	S5	S6	S7	S8	S9	S10	S11	M1	M2	M3	M4	OEC	#A
1	1	0	0	1	1	1	0	0	1	0	0.427	2976.5	206.94	1.718	2.44	21610
0	1	0	0	1	1	1	0	1	1	0	0.423	1810.5	212.62	2.000	2.59	13815
1	1	0	0	1	1	1	0	1	1	0	0.404	2394.7	212.48	1.784	2.45	17964
0	1	0	0	0	1	1	0	0	1	0	0.439	2385.7	212.54	2.000	2.62	276
1	1	0	0	0	1	1	0	0	1	0	0.416	3276.3	212.71	1.709	2.44	790
0	1	0	0	0	1	1	0	1	1	0	0.411	2044.3	218.50	2.000	2.60	505
1	1	0	0	0	1	1	0	1	1	0	0.392	2789.8	217.88	1.775	2.45	724
0	1	0	1	1	1	1	0	0	0	0	0.437	2118.0	208.00	2.000	2.60	240
1	1	0	1	1	1	1	0	0	0	0	0.431	2389.1	208.38	1.620	2.40	800
0	1	0	1	1	1	1	0	1	0	0	0.441	1441.2	212.16	2.000	2.63	496
1	1	0	1	1	1	1	0	1	0	0	0.430	1807.0	211.55	1.738	2.47	672
0	1	0	1	1	1	1	0	0	1	0	0.460	1846.5	210.38	2.000	2.65	2364
1	1	0	1	1	1	1	0	0	1	0	0.448	2379.1	212.04	1.699	2.49	3726
0	1	0	1	1	1	1	0	1	1	0	0.458	1449.0	215.52	2.000	2.67	2325
1	1	0	1	1	1	1	0	1	1	0	0.442	1930.9	216.91	1.775	2.53	1560

Continued on Next Page...

Table 22 – Continued

S1	S2	S3	S4	S5	S6	S7	S8	S9	S10	S11	M1	M2	M3	M4	OEC	#A
0	1	0	1	0	1	1	0	0	1	0	0.448	1930.7	216.36	2.000	2.65	84
1	1	0	1	0	1	1	0	0	1	0	0.435	2620.5	217.59	1.685	2.48	146
0	1	0	1	0	1	1	0	1	1	0	0.446	1668.5	221.12	2.000	2.67	91
1	1	0	1	0	1	1	0	1	1	0	0.429	2272.8	221.49	1.764	2.52	72
0	1	0	0	1	0	1	0	0	1	0	0.442	3200.8	207.81	2.000	2.61	496
1	1	0	0	1	0	1	0	0	1	0	0.418	3809.9	206.34	1.646	2.38	4030
0	1	0	0	1	0	1	0	1	1	0	0.418	2171.5	214.45	2.000	2.59	2015
1	1	0	0	1	0	1	0	1	1	0	0.398	2856.0	211.70	1.750	2.42	6820
0	1	0	0	0	0	1	0	0	1	0	0.433	3375.0	214.00	2.000	2.62	16
1	1	0	0	0	0	1	0	0	1	0	0.409	4334.8	212.54	1.646	2.39	130
0	1	0	0	0	0	1	0	1	1	0	0.409	2533.8	220.65	2.000	2.60	65
1	1	0	0	0	0	1	0	1	1	0	0.390	3460.5	217.89	1.750	2.43	220
0	1	0	1	1	0	1	0	0	1	0	0.447	2667.3	208.81	2.000	2.62	248
1	1	0	1	1	0	1	0	0	1	0	0.439	3135.4	209.60	1.632	2.43	1178
0	1	0	1	1	0	1	0	1	1	0	0.458	1726.7	214.96	2.000	2.67	589

Continued on Next Page...

Table 22 – Continued

S1	S2	S3	S4	S5	S6	S7	S8	S9	S10	S11	M1	M2	M3	M4	OEC	#A
1	1	0	1	1	0	1	0	1	1	0	0.443	2356.3	213.89	1.750	2.51	1116
0	1	0	1	0	0	1	0	0	1	0	0.437	2812.5	215.00	2.000	2.63	8
1	1	0	1	0	0	1	0	0	1	0	0.430	3574.7	215.79	1.632	2.43	38
0	1	0	1	0	0	1	0	1	1	0	0.448	2055.8	221.16	2.000	2.67	19
1	1	0	1	0	0	1	0	1	1	0	0.433	2903.3	220.08	1.750	2.52	36
1	1	0	0	1	1	0	0	0	0	0	0.427	2320.3	200.55	1.484	2.30	496
1	1	0	0	1	1	0	0	1	0	0	0.391	1716.1	207.03	1.670	2.35	1840
1	1	0	0	1	1	0	0	0	1	0	0.444	2267.8	203.69	1.647	2.42	8060
1	1	0	0	1	1	0	0	1	1	0	0.409	1810.2	209.88	1.755	2.44	13815
1	1	0	0	0	1	0	0	0	1	0	0.434	2578.2	209.70	1.638	2.42	276
1	1	0	0	0	1	0	0	1	1	0	0.398	2176.4	215.73	1.745	2.43	505
1	1	0	1	1	1	0	0	0	0	0	0.466	1576.4	208.53	1.467	2.39	240
1	1	0	1	1	1	0	0	1	0	0	0.445	1280.5	211.32	1.677	2.47	496
1	1	0	1	1	1	0	0	0	1	0	0.471	1644.5	210.01	1.635	2.49	2364
1	1	0	1	1	1	0	0	1	1	0	0.454	1386.1	214.30	1.762	2.54	2325

Continued on Next Page...

Table 22 – Continued

S1	S2	S3	S4	S5	S6	S7	S8	S9	S10	S11	M1	M2	M3	M4	OEC	#A
1	1	0	1	0	1	0	0	0	1	0	0.461	1858.2	216.07	1.619	2.49	84
1	1	0	1	0	1	0	0	1	1	0	0.442	1675.2	219.97	1.747	2.53	91
1	1	0	0	1	0	0	0	0	1	0	0.420	2851.4	202.81	1.500	2.30	496
1	1	0	0	1	0	0	0	1	1	0	0.386	2005.5	209.87	1.708	2.37	2015
1	1	0	0	0	0	0	0	0	1	0	0.411	3480.0	209.00	1.500	2.31	16
1	1	0	0	0	0	0	0	1	1	0	0.377	2555.1	216.06	1.708	2.38	65
1	1	0	1	1	0	0	0	0	1	0	0.447	2064.8	207.31	1.500	2.37	248
1	1	0	1	1	0	0	0	1	1	0	0.434	1516.3	212.28	1.737	2.48	589
1	1	0	1	0	0	0	0	0	1	0	0.438	2520.0	213.50	1.500	2.37	8
1	1	0	1	0	0	0	0	1	1	0	0.425	1945.3	218.47	1.737	2.49	19
0	1	0	0	1	1	0	0	0	0	0	0.478	900.0	199.00	2.000	2.64	16
0	1	0	0	1	1	0	0	1	0	0	0.413	837.6	208.07	2.000	2.56	240
0	1	0	0	1	1	0	0	0	1	0	0.487	1069.7	201.85	2.000	2.67	976
0	1	0	0	1	1	0	0	1	1	0	0.437	1013.8	209.36	2.000	2.61	3790
0	1	0	0	0	1	0	0	0	1	0	0.477	1125.0	208.00	2.000	2.68	32

Continued on Next Page...

Table 22 – Continued

S1	S2	S3	S4	S5	S6	S7	S8	S9	S10	S11	M1	M2	M3	M4	OEC	#A
0	1	0	0	0	1	0	0	1	1	0	0.426	1162.4	215.48	2.000	2.61	130
0	1	0	1	1	1	0	0	0	0	0	0.504	630.0	208.00	2.000	2.73	16
0	1	0	1	1	1	0	0	1	0	0	0.471	653.1	211.43	2.000	2.68	112
0	1	0	1	1	1	0	0	0	1	0	0.500	772.8	207.28	2.000	2.72	480
0	1	0	1	1	1	0	0	1	1	0	0.480	796.3	212.37	2.000	2.70	1066
0	1	0	1	0	1	0	0	0	1	0	0.489	810.0	213.50	2.000	2.72	16
0	1	0	1	0	1	0	0	1	1	0	0.469	916.6	218.47	2.000	2.70	38

REFERENCES

[1] 111 th United States Congress, *Weapon Systems Acquisition Reform Act of 2009*. United States Congress, 2009.

[2] Ahuja, R. K., Kumar, A., Jha, K. C., and Orlin, J. B., "Exact and heuristic algorithms for the weapon target assignment problem," *Operations Research*, vol. 55, no. 6, pp. 1136–1146, 2007.

[3] Alexander, P., "Rosetta: Standardization at the system level," *Computer*, vol. 42, no. 1, pp. 108 – 110, 2009.

[4] Alghamdi, A. S., "Evaluating Defense Architecture Frameworks for C4I System Using Analytic Hierarchy Process," *Journal of Computer Science*, vol. 5, no. 12, pp. 1075–1081, 2009.

[5] Amazon Web Services, "Amazon elastic compute cloud." Accessed Mar. 9th 2011. `http://aws.amazon.com/ec2/`.

[6] Andrews, R. A., "An overview of acquisition logistics," *Defense Acquisition University*, 2007.

[7] Bar-Yam, Y., *Dynamics of Complex Systems*. Reading, MA: Perseus Books, 1997.

[8] Bassi, L., Secchi, C., Bonfe, M., and Fantuzzi, C., "A SysML-Based Methodology for Manufacturing Machinery Modeling and Design," *Transactions on Mechatronics*, vol. 16, no. 6, pp. 1049 – 1062, 2011.

[9] Bell, A. E., "Death by UML Fever," *ACM Queue*, vol. 2, pp. 72–80, March 2004.

[10] Birkler, J., "Untying Gulliver: Taking Risks to Acquire Novel Weapon Systems," *RAND Occasional Paper*, 2009. RAND OP-268-OSD.

[11] Bixby, R. E., "Solving real-world linear programs: A decade and more of progress," *Operations Research*, vol. 50, pp. 3–15, Jan-Feb 2002.

[12] Bolkcom, C., "Military Suppression of Enemy Air Defenses (SEAD): Assessing Future Needs," *CRS Report for Congress*, January 2005.

[13] Bouvier, E., Cohen, E., and Najman, L., "From crowd simulation to airbag deployment: Particle systems, a new paradigm for simulation," *Journal of Electronic Imaging*, vol. 6, pp. 94–107, January 1997.

[14] Box, G. E. P. and Draper, N. R., *Empirical Model-Building and Response Surfaces*. John Wiley & Sons, Inc., 1987.

[15] Broy, M., Gleirscher, M., Merenda, S., Wild, D., Kluge, P., and Krenzer, W., "Toward a Holistic and Standardized Automotive Architecture Description," *Computer*, vol. 42, no. 12, pp. 98–101, 2009.

[16] C4ISR Architecture Working Group, *C4ISR Architecture Framework Version 2.0*. DoD, 1997.

[17] Cabrera, R., "Mission-Level Systems Engineering," in *Enterprise Architecture Conference*, DoD Enterprise Architecture Conference: Delivering Warfighting Capability, (Hampton, VA), April 11-15 2011.

[18] Carr, D. B., Littlefield, R. J., Nicholson, W. L., and Littlefield, J. S., "Scatterplot matrix techniques for large n," *Journal of the American Statistical Association*, vol. 82, no. 398, pp. pp. 424–436, 1987.

[19] Chairman of the Joint Chiefs of Staff, *Joint Capabilities Integration and Development System*, March 2009. Instruction CJCSI 3170.01G.

[20] Charette, R., "What's Wrong With Weapons Acquisitions?," *Spectrum, IEEE*, vol. 45, pp. 33 –39, November 2008.

[21] Chen, C. C.-Y. and Das, S. K., "Breadth-first traversal of trees and integer sorting in parallel," *Information Processing Letters*, vol. 41, pp. 39–49, January 1992.

[22] Chen, D. and Stroup, W., "General system theory: Toward a conceptual framework for science and technology education for all," *Journal of Science Education and Technology*, vol. 2, no. 3, pp. 447–459, 1993.

[23] Cheung, T.-Y., "Graph traversal techniques and the maximum flow problem in distributed computation," *IEEE Transactions on Software Engineering*, vol. SE-9, no. 4, pp. 504–512, 1983.

[24] CJCS, *Universal Joint Task Manual*. DoD, 2008. Chairman of the Joint Chiefs of Staff Manual 3500.04E.

[25] CJCS, *Manual for the Operation of the Joint Capabilities Integration and Development System*. Chairman of the Joint Chiefs of Staff, 2009. Updated July 31, 2009.

[26] Cohen, R., Katzir, L., and Raz, D., "An efficient approximation for the generalized assignment problem," *Information Processing Letters*, vol. 100, pp. 162–166, November 2006.

[27] Colombo, P., Khendek, F., and Lavazza, L., "Bridging the gap between requirements and design: An approach based on Problem Frames and SysML," *Journal of Systems and Software*, vol. 85, pp. 717–745, March 2012.

[28] Comittee on Pre-Milestone A SE, *Pre-Milestone A and Early-Phase Systems Engineering: A Retrospective Review and Benefits for Future Air Force Systems Acquistion*. National Academies Press, 2008.

[29] Committee on Naval Analytical Capabilities and Improving Capabilities-Based Planning, "Naval analytical capabilities: Improving capabilities-based planning," tech. rep., National Research Council of the National Academies, 2005. The National Academies Press.

[30] Complex, "Merriam-Webster Online Dictionary," 2010. Retrieved May 24, 2010, from `http://www.merriam-webster.com/dictionary/complex`.

[31] Complicated, "Merriam-Webster Online Dictionary," 2010. Retrieved May 24, 2010, from `http://www.merriam-webster.com/dictionary/complicated`.

[32] Corps, U. M., *MCWP 3-22.2 Suppression of Enemy Air Defenses (SEAD)*. U.S. Marine Corps, 2001.

[33] DARPA, *META*. Tactical Technology Office, 2009.

[34] DARPA, *Abstraction Based Complexity Management*. DARPA, April 2010. DARPA-BAA-10-59 Appendix F.

[35] DARPA, *META-II*. Tactical Technology Office, 2010. DARPA-BAA-10-59.

[36] Davis, P. K., *Analytic Architecture for capabilities-based Planning, Mission-System Analysis, and Transformation*. RAND, 2002. RAND MR-1513-OSD.

[37] Davis, P. K. and Dreyer, P., "RAND's Portfolio Analysis Tool (PAT): Theory, Methods, and Reference Manual," tech. rep., RAND, 2009.

[38] Dean, J. and Ghemawat, S., "Mapreduce: Simplified data processing on large clusters," *OSDI'04: Sixth Symposium on Operating System Design and Implementation, San Francisco, CA.*, December 2004.

[39] Defense Science Board, *21st Century Strategic Technolgoy Vectors*, vol. I. Office of the Under Secretary of Defense For Acqusition, Technology, and Logistics, February 2007.

[40] Department of Defense, *Department of Defense Instruction Number 5000.02: Operation of the Defense Acquisition System*, December 2008.

[41] Despotou, G., Alexander, R., and Hall-May, M., "Key Concepts and Characteristics of Systems of Systems," *DARP - HIRTS Strand 2*, 2003.

[42] DeVries, D. and Westphal, M., "What's Ahead for Architectures in the DoD?," in *Enterprise Architecture Conference*, DoD Enterprise Architecture Conference: Delivering Warfighting Capability, (Hampton, VA), April 11-15 2011.

[43] Dickerson, C. E. and Mavris, D. N., *Architecture and Principles of Systems Engineering*. CRC Press, 2010.

[44] Dieter, G. E., *Engineering Design: A materials and Processing Approach*. McGraw Hill, 3rd ed., 2000.

[45] DoD, "Defense knowledge online." website: `http://www.us.army.mil/suite/page/454707`.

[46] DoD, "Quadrennial defense review report," *Department of Defense Publication*, September 2001.

[47] DoD, "Capability portfolio management," *Department of Defense Directive*, September 2008. Department of Defense Directive 7045.20.

[48] DoD, "DoDAF v2.0 Volume 1: Introduction, Overview, and Concepts: Manager's Guide," May 2009.

[49] DoD, "Dodaf v2.0 volume 2: Architectural data and models : Architect's guide," May 2009.

[50] DoD, *Defense Acquisition Guidebook*. Defense Acquisition University, 2010. Website. Accessed on May 28^{th}, 2010. `https://dag.dau.mil/`.

[51] DoD, "Quadrennial defense review report," *Department of Defense Publication*, February 2010.

[52] DoD Architecture Framework Working Group, *DoD Architecture Framework Version 1.0*, vol. 1. DoD, August 2003.

[53] DoD Architecture Framework Working Group, *DoD Architecture Framework Version 1.5*, vol. 1. DoD, April 2007.

[54] DoD DCIO, "Dodaf journal." website: `http://cio-nii.defense.gov/sites/dodaf20/journal_exp3.html`.

[55] DoDAF 2.0 Work Group, "DoDAF v2.0 Way Ahead," in *Enterprise Architecture Conference*, DoD Enterprise Architecture Conference: Unity of Effort – Readiness for Action – Mission Success, (San Antonio, TX), May 10-14 2010.

[56] Domercant, J. C., *ARC-VM: An Architecture Real Options Complexity-Based Valuation Methodology for Military Systems-of-Systems Acquisitions*. PhD thesis, Georgia Institute of Technology, 2011.

[57] Domercant, J. C. and Mavris, D. N., "Measuring the Architectural Complexity of Military Systems-of-Systems," in *2011 IEEE Aerospace Conference*, IEEE Aerospace Conference (Big Sky, MT), pp. 1–16, March 5-12 2011. Paper No. 1649.

[58] Dori, D., *Object-process methodology : A Holistics Systems Paradigm*. Springer, 2002.

[59] dos Santos Soares, M., Vrancken, J., and Verbraeck, A., "User requirements modeling and analysis of software-intensive systems," *Journal of Systems & Software*, vol. 84, no. 2, pp. 328–339, 2011.

[60] Dreyer, P. and Davis, P. K., "A portfolio-analysis tool for missile defense (pat-md): Methodoogy and user's manual," Tech. Rep. TR-262, RAND Corporation, 2005.

[61] Duke, D., Sims, D. E., and Pharmer, J., "Reimagining workload task analysis: Applications to training system design," *Defense Acquisition Review Journal*, vol. 18, pp. 428 – 448, October 2011.

[62] Enterprise Architecture and Standards Directorate, *OMG Systems Engineering UPDM*, September 2009. Brief to OMG System Engineering 2009.

[63] Fang, W., He, B., Luo, Q., and Govindaraju, N. K., "Mars: Accelerating MapReduce with Graphics Processors," *IEEE Transactions on Parallel and Distributed Systems*, vol. 22, no. 4, pp. 608 – 620, 2011.

[64] Fei, X. and Lu, S., "A dataflow-based scientific workflow composition framework," *IEEE TRANSACTIONS ON SERVICES COMPUTING*, vol. 5, pp. 45–58, January - March 2012.

[65] Ferguson, J., "Military electronic systems for command and control," *IEEE Transactions on Military Electronics*, vol. 9, no. 2, pp. 80 – 87, 1965.

[66] Fleischer, L., Goemans, M. X., Mirrokni, V. S., and Sviridenko, M., "Tight approximation algorithms for maximum general assignment problem," *SODA*, pp. 611–620, 2006.

[67] Fonoage, M., Cardei, I., and Shankar, R., "Mechanisms for requirements driven component selection and design automation," *Systems Journal*, vol. 4, no. 3, pp. 396 – 403, 2010.

[68] for Standards, D. I. S. A. C., *Technical Architecture Framwork for Information Management Version 3.0*. DoD, 1996.

[69] Framework, "Merriam-Webster Online Dictionary," 2010. Retrieved May 24, 2010, from `http://www.merriam-webster.com/dictionary/framework`.

[70] Fulghum, D. A., "Melding Ops and Intel," *Aviation Week and Space Technology*, pp. 53–54, April 26 2010.

[71] Gell-Mann, M., "What is complexity?," *Complexity*, vol. 1, no. 1, 1995.

[72] Graves, H. and Bijan, Y., "Using formal methods with SysML in aerospace design and engineering," *Annals of Mathematics and Artificial Intelligence*, vol. 63, no. 1, pp. 53–102, 2011.

[73] Greene, H. and Mendoza, R., "Lessons Learned from Developing the ABCS 6.4 Solution," *DEFENSE ACQUISITION REVIEW JOURNAL*, vol. 12, pp. 195– 215, April - July 2005.

[74] Griendling, K., *ARCHITECT: The Architecture-based Technology Evaluation and Capability Tradeoff Method*. PhD thesis, Georgia Institute of Technology, December 2011.

[75] Griendling, K. and Mavris, D., "A process for systems of systems architecting," in *8th AIAA Aerospace Sciences Meeting Including the New Horizons Forum and Aerospace Exposition, Orlando, Florida*, Jan. 4-7 2010. AIAA-2010-786.

[76] Hause, M., "UPDM – Unifed Progfile for DoDAF/MoDAF," in *Integrated Enterprise Architecture 2009*, Artisan Software Tools, 2009.

[77] Hause, M., Thom, F., and Moore, A., "Inside SysML," *Electronics Systems and Software*, vol. 3, no. 3, pp. 20–25, 2005.

[78] IEEE, "IEEE 1471-2000 Recommended Practice for Architecture Description of Software-Intensive Systems," September 2000.

[79] IEEE Standards Board, "IEEE Standard Glossary of Software Engineering Terminology," 1990.

[80] ISO/IEC/IEEE, "Systems and software engineering – architecture description." ISO/IEC/IEEE 42010:2011(E), Dec 2011.

[81] Izquierdo, J. L. C. and Molina, J. G., "An architecture-driven modernization tool for calculating metrics," *IEEE Software*, vol. 27, no. 4, pp. 37 – 43, 2010.

[82] JMP, "Version 9," 1989-2011. SAS Institute Inc. Cary, NC.

[83] Johnson, N. F., *Two's Company, Three is Complexity: A simple guide to the science of all sciences.* Oneworld Publications, 2007.

[84] Joint Chiefs of Staff, *Joint Publication 1-02: Department of Defense Dictionary of Military and Associated Terms*. DoD, Apr. 2001 As amended through Oct. 2009.

[85] Joint Staff, *JTTP for Joint Suppression of Enemy Air Defenses (J-SEAD)*. Joint Chiefs of Staff, 1995.

[86] Joint Staff, "Joint Publication 3-05: Special Operations." Joint Publication 3-05, April 2011.

[87] Kaiser, D., "Physics and feynman's diagrams," *American Scientist*, vol. 93, pp. 156–165, March-April 2005.

[88] Karsai, G., Maroti, M., Ledeczi, A., Gray, J., and Sztipanovits, J., "Composition and cloning in modeling and meta-modeling," *IEEE Transactions on Control Systems Technology*, vol. 12, no. 2, pp. 263 – 278, 2004.

[89] Kelley, R. E., "U.S. Army Special Forces Unconventional Warfare Doctrine: Engine of Change or Relic of the Past?," Master's thesis, Naval War College, 2000.

[90] Kent, G. A., Ochmanek, D., Spirtas, M., and Pirnie, B. R., *Thinking About America's Defense: An Analytical Memoir*. RAND, 2008. RAND OP-223-AF.

[91] Kerzhner, A. A. and Paredis, C. J. J., "Using domain specific languages to capture design synthesis knowledge for model-based systems engineering," *ASME Conference Proceedings*, vol. 2009, no. 48999, pp. 1399–1409, 2009.

[92] Klein, H. A., "Computational modeling in complex multinational domains," *IEEE Intelligent Systems*, vol. 24, no. 1, pp. 84–88, 2009.

[93] Lagerström, R., Johnson, P., and Höök, D., "Architecture analysis of enterprise systems modifiability – models, analysis, and validation," *Journal of Systems & Software*, vol. 83, no. 8, pp. 1387–1403, 2010.

[94] Lee, S. and Johnson, P., "DoDAF v2.0 In Action: Search and Rescue (SAR) Example: User Experience and Analysis," in *Enterprise Architecture Conference*, DoD Enterprise Architecture Conference: Delivering Warfighting Capability, (Hampton, VA), April 11-15 2011.

[95] Levin, C., "Summary of the weapon systems acquisition reform act of 2009," *Website*, February 24^th 2009. Accessed on May 25^th at `http://levin.senate.gov/newsroom/release.cfm?id=308525`.

[96] Lopez, D. M. and Blobel, B. G., "A development framework for semantically interoperable health information systems," *International Journal of Medical Informatics*, vol. 78, no. 2, pp. 83 – 103, 2009.

[97] Mahulkar, V., McKay, S., Adams, D. E., and Chaturvedi, A. R., "System-of-systems modeling and simulation of a ship environment with wireless and intelligent maintenance technologies," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 39, no. 6, pp. 1255–1270, 2009.

[98] Maier, M., Emery, D., Hilliard, R., Kerner, J., and Kruchten, P., "Frequently Asked Questions: ISO/IEC/IEEE 42010." website. `http://www.iso-architecture.org/ieee-1471/faq.html`. Accessed 1/20/12.

[99] Maier, M. W., "Architecting principles for systems-of-systems," *System Engineering*, vol. 1, pp. 267–284, 1998.

[100] Makki, S., "Efficient distributed breadth-first search algorithm," *Computer Communications*, vol. 19, pp. 628–636, 1996.

[101] Makki, S. and Havas, G., "Distributed algorithm for depth-first search," *Information Processing Letters*, vol. 60, pp. 7–12, 1996.

[102] Marks, R. and Joint Architecture Working Group, "Enterprise Architecture in the Intelligence Community: The Joint Architecture Working Group (JAWG)," in *Enterprise Architecture Conference*, DoD Enterprise Architecture Conference: Delivering Warfighting Capability, (Hampton, VA), April 11-15 2011.

[103] McCarthy, J., "Lisp - notes on its past and future," *Conference on LISP and Functional Programming*, vol. Proceedings of the 1980 ACM conference on LISP and functional programming, 1980.

[104] McGarvey, R. G., Tripp, R. S., Rue, R., Lang, T., Sollinger, J. M., Conner, W. A., and Luangkesorn, L., *Global Combat Support Basing: Robust Prepositioning Strategies for Air Force War Reserve Materiel*. RAND, 2010. RAND MG-902-AF.

[105] Mercer, B., "Enabling executable architecture by improving the foundations of dod architecting," in *Collaborative Technologies and Systems, 2008. CTS 2008. International Symposium on*, pp. 558 –559, 19-23 2008.

[106] Methodology, "Merriam-webster online dictionary," 2010. Retrieved May 24, 2010, from `http://www.merriam-webster.com/dictionary/methodology`.

[107] Milicev, D., "Automatic Model Transformations Using Extended UML Object Diagrams in Modeling Environments," *IEEE Transactions on Software Engineering*, vol. 28, no. 4, pp. 413 – 431, 2002.

[108] Mittal, S., "Extending DoDAF to Allow Integrated DEVS-Based Modeling and Simulation," *Journal of Defense Modeling and Simulation: Applications,Methodology, Technology*, vol. 3, pp. 95–123, April 2006.

[109] Monch, L., Lendermann, P., McGinnis, L. F., and Schirrmann, A., "A survey of challenges in modelling and decision-making for discrete event logistics systems," *Computers in Industry*, vol. 62, no. 6, pp. 557–567, 2011.

[110] Morihata, A., Matsuzaki, K., Hu, Z., and Takeichi, M., "The third homomorphism theorem on trees: Downward and upward lead to divide-and-conquer," in *POPL 2009*, ACM, January 2009.

[111] NASA, *NASA System Engineering Handbook*. NASA STI, 2007. NASA/SP-2007-6105 Rev 1.

[112] Northrop Grumman, "X-47B UCAS." Website. last accessed on 1/30/12. http://www.as.northropgrumman.com/products/nucasx47b/index.html.

[113] Obama, B., "National security strategy," *White House*, May 2010.

[114] Object Management Group, "OMG Systems Modeling language." website, May 2010. Acccessed on May 31st 2010 at http://www.omgsysml.org/.

[115] Object Management Group (OMG), "Documents Associated with UML Version 2.3." website, May 2010. Accessed on May 31st 2010 at http://www.omg.org/spec/UML/2.3/.

[116] Office of the Deputy Under Secretary of Defense for Acquisition and Technology, Systems and Software Engineering, *Systems Engineering Guide for Systems of Systems*. Washington, DC: DoD, August 2008. Version 1.0.

[117] Osmundson, J. S., Gottfried, R., Kum, C. Y., Boon, L. H., Lian, L. W., Patrick, P. S. W., , and Thye, T. C., "Process modeling: A systems engineering tool for analyzing complex systems," *Systems Engineering*, vol. 7, pp. 320 – 337, December 2004.

[118] Paul K. Davis, R. D. S. and Beck, J., *Portfolio-Analysis Methods for Assessing Capability Options*. Santa Monica, CA.: RAND Corporation, 2008.

[119] Perlis, A. J., "Special feature: Epigrams on programming," *SIGPLAN Not.*, vol. 17@MISCframework, author = Framework, title = Merriam-Webster Online Dictionary, year = 2010, note = Retrieved May 24, 2010, from http://www.merriam-webster.com/dictionary/framework, owner = jiacobucci, timestamp = 2010.05.24 , no. 9, pp. 7–13, 1982.

[120] President's Council of Advisors on Science and Technology, "Designing a digital future: Federally funded research and development in networking and information technology," tech. rep., Executive Office of the President, December 2010. Report to the President and Congress.

[121] Rechtin, E. and Maier, M. W., *The Art of Systems Architecting*. CRC Press, 2nd ed., 2001.

[122] Ren, C. H., Busch, S., and Prebble, M., "Improving the initiation of acquisition activities for automated information systems," *Defense Acquisition Review Journal*, vol. 17, pp. 418 – 435, October 2010.

[123] Reveron, D. S. and Cook, J. L., "Developing strategists: Translating national strategy into theater strategy," *Joint Forces Quarterly*, pp. 21–28, October 4th quarter 2009.

[124] Richards, M. A. and Campbell, D. P., "Rapidly reconfigurable high performance computing cluster," tech. rep., Georgia Tech Research Institute, July 2005.

[125] Sage, A. P., *Methodological Considerations in the Design of large Scale Systems Engineering Processes*, vol. 7 of *Studies in Management Science and Systems*, ch. 5 in Large Scale Systems, pp. 99–141. North-Holland Publishing Company, 1982.

[126] Sage, A. P. and Cuppan, C. D., "On the systems engineering and management of systems of systems and federations of systems," *Information Knowledge Systems Management*, vol. 2, pp. 325–345, 2001.

[127] Schaaf, R. E. V., DeLaurentis, D. A., and Abraham, D. M., "Multi-objective optimization models for improved decision-support in humanitarian infrastructure project selection problems," *IEEE Systems Journal*, vol. 2, no. 4, pp. 536 – 547, 2008.

[128] Sevinc, S., "Extending common lisp object system for discrete event modeling and simulation," in *Winter Simulation Conference*, pp. 204–206, 1991.

[129] Siel, C. and Kuncel, J., "Systems engineering needs of the dod architecture framework," in *NDIA SE Conference 2009*, National Defense Industrial Association, October 2009. NDIA SE Conference 2009 San Diego, CA.

[130] Silvela, J. and Portillo, J., "Breadth-first search and its application to image processing problems," *IEEE Transactions on Image Processing*, vol. 10, pp. 1194–1199, August 2001.

[131] Spinellis, D., "Faking it," *IEEE Software*, vol. 28, no. 5, pp. 96 – 96, 2011.

[132] Steele, G., "Organizing functional code for parallel execution: or, foldl and foldr considered slightly harmful," *Sun Microsystems*, 2009.

[133] Strauss, M., Rock, B. S., Zeidner, L., Desai, N., and Reeve, H., "Application of a technology screening methodology for rotorcraft alternative power systems," in *48th AIAA Aerospace Sciences Meeting Including the New Horizons Forum and Aerospace Exposition*, January 2010. AIAA-2010-1505.

[134] SysML, "SysML – Open Source Specification Project." website, May 2010. Accessed on May 31st at `http://www.sysml.org/`.

[135] System Engineering Handbook Working Group INCOSE, *Systems Engineering Handbook: A Guide for System Life Cycle Processes and Activities*. INCOSE, 3.2 ed., January 2010. Edited by Cecilia Haskins.

[136] Taylor, R. N., Medvidovic, N., and Dashofy, E. M., *Software Architecture: Foundation, Theory, and Practice*. Wiley, January 2009.

[137] Thaler, D. E., *Strategies to Tasks: A Framework for Linking Means and Ends*. RAND, 1993.

[138] United States Navy Fact File, "Aircraft Carriers - CVN." Website, Jan 2012. http://www.navy.mil/navydata/fact_print.asp?cid=4200&tid=200&ct=4&page=1.

[139] United States Navy Fact File, "Destroyers - DDG." Website, Jan 2012. http://www.navy.mil/navydata/fact_display.asp?cid=4200&tid=900&ct=4.

[140] United States Navy Fact File, "E-2 Hawkeye early warning and control aircraft." Website, Jan 2012. http://www.navy.mil/navydata/fact_display.asp?cid=1100&tid=700&ct=1.

[141] United States Navy Fact File, "E-6B Mercury airborne command post." Website, Jan 2012. http://www.navy.mil/navydata/fact_display.asp?cid=1100&tid=900&ct=1.

[142] United States Navy Fact File, "F/A-18 Hornet strike fighter." Website, Jan 2012. http://www.navy.mil/navydata/fact_display.asp?cid=1100&tid=1200&ct=1.

[143] U.S. ARMY Fact Files, "Abrams." Website, Jan 2012. http://www.army.mil/factfiles/equipment/tracked/abrams.html.

[144] U.S. ARMY Fact Files, "Apache longbow." Website, Jan 2012. http://www.army.mil/factfiles/equipment/aircraft/apache.html.

[145] US Department of State, "Strategic arms limitation treaty i (salt i)," May 1972. http://www.state.gov/www/global/arms/treaties/salt1.html.

[146] U.S. DoD, "DoDAF Architecture Framework Version 2.02." Website, August 2010.

[147] van Deursen, A., Klint, P., and Visser, J., "Domain-Specific Languages: An Annotated Bibliography," *SIGPLAN Not.*, vol. 35, no. 6, pp. 26–36, 2000.

[148] Vandiver, R., "Architecture's Role in Capability Development: Architecting - A Campaign Quality Army with a Joint & Expeditionary Mindset," in *Enterprise Architecture Conference*, DoD Enterprise Architecture Conference: Delivering Warfighting Capability, (Hampton, VA), April 11-15 2011.

[149] Warmer, J. and Kleppe, A., *The Object Constraint Language: Getting your Models Ready for MDA*. Addison-Wesley, 2003.

[150] Wayson, M. L., "Dod architecture framework version 2.0 release overview," June 2009. 12 June 2009. Enterprise Architecture and Standards Directorate.

[151] Wayson, M. L., "DoDAF v2.0 Update," in *Enterprise Architecture Conference*, DoD Enterprise Architecture Conference: Unity of Effort – Readiness for Action – Mission Success, (San Antonio, TX), May 10-14 2010.

[152] Willard, B., "UML for Systems Engineering," *Computer Standards & Interfaces*, vol. 29, no. 1, pp. 69–81, 2007.