SYSTEMS ENGINEERING
Research Center

Development of 3-Year Roadmap to Transform the Discipline of Systems Engineering

Technical Report SERC-2010-TR-006-1

March 31, 2010

Principal Investigator: Dr. Jon Wade, Stevens Institute of Technology

Co-Principal Investigator: Dr. Azad Madni, University of Southern California

Team Members:

Pennsylvania State University: Dr. Colin Neill

Stevens Institute of Technology: Dr. Robert Cloutier, Dr. Richard Turner, Peter Korfiatis, Anne Carrigy,

With Contributions by:

University of Southern California: Dr. Barry Boehm,

Stevens Institute of Technology: Stanislaw Tarchalski

Contract Number: H98230-08-D-0171

Task Order: 0002, DO1, RT 010

Report No. SERC-2010-TR-006-1

The Systems Engineering Research Center (SERC) is a federally funded University Affiliated Research Center managed by Stevens Institute of Technology.

This material is based upon work supported, in whole or in part, by the U.S. Department of Defense through the Office of the Assistant Secretary of Defense for Research and Engineering (ASD(R&E)) under Contract H98230-08-D-0171 (Task Order 0002, DO1, RT10).

EXECUTIVE SUMMARY

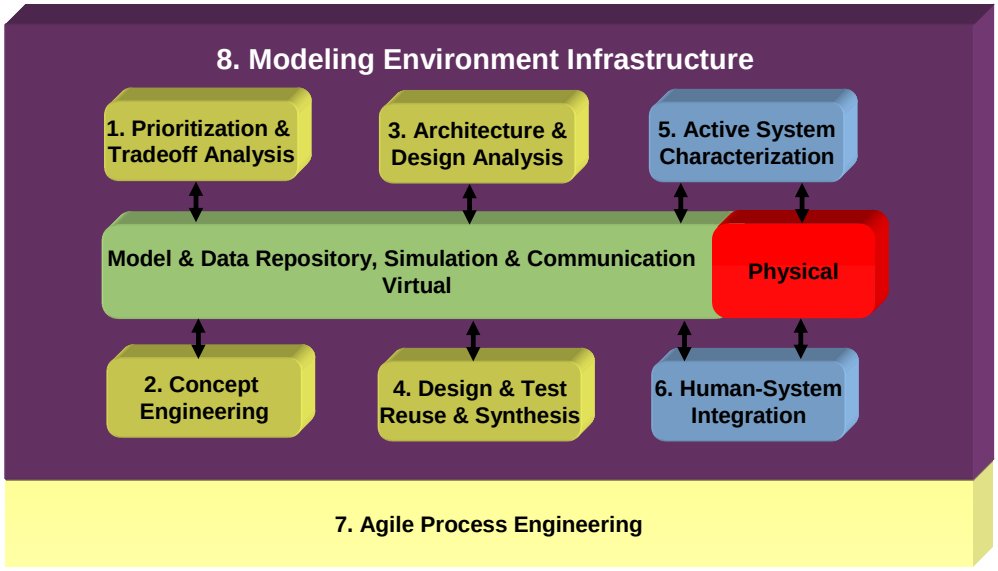
As systems continue to grow in size and complexity, it has become clear that existing Systems Engineering (SE) methods, processes and tools are becoming increasingly inadequate. The SET project is intended to identify the gaps and bring about the necessary transformation in Systems Engineering to satisfy the needs of the complex system's life cycle. Accomplishing this transformation requires a fundamental rethinking of current SE practices. The SET project is focused on first principles and stripping away non-essential activities while being cognizant of recent trends in SE. A number of trends collectively accelerate this challenge. Growing system complexity and criticality raise vulnerability. The ascendancy of software as the preferred solution continues in the face of significant gaps in our ability to understand, validate and manage large evolving software ecosystems. The increasing speed of technological change, the rapid evolution of threats, and the decreasing schedules for development all lead to the sense that time itself is compressing. New systems envisioned by the defense and intelligence communities reflect, embrace and reinforce these trends.

The SET project identified specific characteristics of the SE transformation that are embodied in the following attributes:

- **Agile:** Allowing for quality, timely development with an incomplete and changing set of system requirements.
- **Integrated:** Part of the main development process and not an additional set of discretionary tasks.
- **Efficient:** Providing the greatest amount of benefits with the minimal number of steps and least amount of effort.
- **Leveraged:** Enabling exponential capability growth through the leveraging of computational and information technologies, and prior systems experience.
- **Extensible:** Providing the ability to expand and enhance capabilities for future growth without having to make major changes in the infrastructure.
- **Deployable:** Enabling widespread impact through workforce education and broad application.

Our analysis of existing Systems Engineering methods, processes and tools has identified significant gaps and a set of eight research thrusts to begin addressing those gaps. These research thrusts are the core components of an integrated, modular road map toward SE transformation. A workshop held with the sponsor confirmed the relevance of these research thrusts and provided the necessary input that resulted in their refinement and the creation of the overall framework.

The resulting SE research framework is shown in the figure below. Central to this framework is a model and data repository, simulation and communication substrate that provides the ability for the various tool sets and capabilities to synchronize and interoperate. Each of the research areas may address systems composed of arbitrary combinations of hardware, software, human agents, and governance systems. The systems being developed may be a complex system of systems, a standalone platform or a rapid response action.



SET Research Area Framework

The **Prioritization & Tradeoff Analysis** module provides the capability to input the particular factors relating to the relative value and priority of high-level capabilities of the system under development. The **Concept Engineering** module provides an interactive, collaborative, multimedia environment to multiple stake holders, along with a library of concept modules, and Reuse and Synthesis capabilities to quickly construct concepts of operation and other high-level abstract models of the system under development. The **Architecture and Design Analysis** module provides the system architect and human operator with the ability to develop and optimize an architecture and design which supports the conceptual view while providing an optimal solution based on the Prioritization & Tradeoff Analysis models described earlier. **Design & Test Reuse and Synthesis** provides the means, by leveraging existing assets and utilizing computational capabilities, to rapidly translate high level abstractions into lower level ones. These capabilities can be used across the entire range of design and test abstractions from concept to implementation. **Active System Characterization** has the role of providing feedback between the virtual and physical system domains. This module constantly monitors the actively deployed system and feeds back this information into the model and data repository ensuring that that this information is up to date in near real time. **Human-System Integration**, true to its name, is integrated

throughout the system lifecycle activities to ensure that the human considerations are accounted for and modeled with the end goal of optimizing the entire system, not just the technical and human subsystems and components. **Agile Process Engineering** provides the processes and governance to enable productive parallel development in each of the aforementioned areas. Finally, the **Modeling Environment Infrastructure** provides the plumbing that supports the other tools.

Each research area satisfies the following criteria:

- Critical to the transformation of SE
- Furthers the sponsor's mission
- Requires multidisciplinary research which is not currently being done
- Appropriate scope & scale for an academic research program
- Supports a 3-year or longer roadmap of research
- Expected to have measureable impact

Each research area has capabilities that leverage the current state of the art in computation, visualization, communication and information technologies. Future advances in these areas will increase the capabilities of the technologies developed in these research areas, thus keeping Systems Engineering "on the curve".

The road map presented in this study provides a modular, integrated and extensible framework for transforming system engineering. These research efforts are integrated, such that together they provide value that is greater than the sum of their parts, yet remain modular such that each area can proceed and provide value independently. These characteristics allow the funding of the SET roadmap to be extremely flexible with respect to amount, time and source for each research area or combination of areas. Taken separately, these research areas have the potential to significantly advance the state of the art of Systems Engineering. Taken together, they have the potential to transform Systems Engineering.

This page intentionally left blank

TABLE OF CONTENTS

FIGURES AND TABLES

1 PURPOSE

This report documents the results of an eight month Systems Engineering Research Center (SERC) research task. The task created a 3-year research roadmap to transform Systems Engineering into a discipline capable of addressing current and emerging critical system challenges.

1.1 PROBLEM STATEMENT

Traditional Systems Engineering (SE) is not adequate to meet the emerging challenges posed by critical system trends. This inadequacy threatens SE relevancy to current and future system development. Linear improvements in SE have not kept up with the exponentially growing system functional, performance, safety and criticality requirements enabled by rapidly evolving hardware, software and communications technologies. Compounding this issue are decreasing development horizons, constantly changing threat and operational environments, and relatively slow improvements in human capabilities.

The current practices of Systems Engineering were to a large degree created during the post-WWII era and are based on the Systems Principles and Theory developed in the 1930's through 1950's. These methods evolved early in the NASA space race of the 1960's and 70's and are characterized by the design of the Apollo missions. However, those systems represent architecture and technology that is now half a century old. Today's contemporary mobile telephone has far more software and computing capability than the Apollo spacecraft. It is safe to say that traditional SE Methods, Processes and Tools (MPT) were developed for less complex, hardware dominated systems, and were accommodated by relatively stable, long planning cycles.

> "Systems Engineering has traditionally been aimed at driving certainty into developments and reducing the risk to the success of the final product. This approach continues to be valuable in life-critical situations (e.g., space shuttle) and projects for which 'trying again' is fiscally prohibitive (e.g., satellite launch)."
>
> "The IT systems environment - coupled with the culture and life-experiences of today's developers - could render the traditional SE "V-model" approach irrelevant. Requiring elaborate documentation, detailed requirements definition, risk avoidance, and long term plans are all counter-intuitive to the nature of the technologist working in today's IT domain." [Barnabe, 2009].

Furthermore, when systems have tens of millions of lines of software code, it is physically impossible to develop and then test all the possible paths with current reductionist and deterministic practices and tools.

1.2 CRITICAL SYSTEMS TRENDS

The exponential increase in computing capability and storage, coupled with the exponential growth of network bandwidth, has unleashed global networked computing and is having tremendous ramifications in every aspect of our social, economic and political lives. The following are some critical trends that are pushing current Systems Engineering practices into obsolescence in many critical domains and applications.

The first trend is that systems are becoming increasingly more complex in both number of components and interconnectivity. Decreasing numbers of complex systems are "stand-alone", but are instead connected to other complex systems. Not only is there an exponential growth in connectivity and interdependencies, but many of these are hidden. What once might have been a local issue is now a globally networked challenge. The examples are many. Systems of systems, and in particular Network Centric Services, have become the norm rather than the exception. Successful engineering of these systems may require understanding the technical, social, political, economic, behavioral and environmental implications to develop a system that best serves the relevant stakeholders. Quite often these are complex adaptive System of Systems in which direct control may not be possible such that influence and the creation of a reward system to guide self-organization become the rule for the system "designer". Compounding this challenge is the need to interface with legacy systems of all shapes and forms, which is noted as the fifth trend.

The second trend is our increasing societal day-to-day dependency on technologically advanced and powerful, yet unpredictable System of Systems (SoS). The resulting impact is that these systems are no longer a matter of convenience, but instead are necessary for our daily existence. There are numerous examples of how the application of the Internet and web technology has changed our perception of how work should be accomplished – i.e. in a collaborative, distributed, evolvable manner. There is a B2B web that has taken advantage of the communications medium and created extremely complex interdependent business processes and rules. There is the rise of cyber-physical systems, including everything from smart automobiles to environmental controls, to net-centric warfare. There are social networking sites and massive multi-player games. This new paradigm of interacting systems, services and users has fundamentally changed the way systems are conceived, developed, deployed, managed, and retired. It has been the driving force behind the creation of the System of Systems. The notion of Systems of Systems was not created, but rather evolved organically as a technological response to the desires and needs of customers.

The third trend, which compounds the impact of the first two trends, is that a combination of customer expectations and competitive demands has greatly compressed the development and deployment lifecycles of products and services. The result is that while the complexity and criticality of new systems is increasing exponentially, the time to develop and deploy them is decreasing. It is important to differentiate between the development and the evolution of the distributed IT network and infrastructure, and the evolution of application layer functionality. The architecture and infrastructure of the former might have a much longer life cycle than the applications which it supports.

The fourth trend is related to security. Our increasing dependence on networked systems has greatly increased their value as a target while at the same time the increased complexity and interconnectedness increases their vulnerability. Security has to be of central importance in the design and deployment of systems, particularly when they themselves are composed of systems which are unreliable and independently evolving.

The fifth trend, which relates to the first trend of complexity, is that so-called "greenfield development" (development that is completely new, with no legacy issues to address) is becoming the exception rather than the rule, particularly in netcentric systems which must increasingly work with legacy systems. In addition, these legacy systems are often ill-planned and unsuited for their future missions. This is not necessarily the result of poor planning, architecting, design and execution, but rather often is the result that their future usage was unforeseen at the time of their design. The extension of a system's service life well beyond the original plan due to the significantly increasing cost and time for a replacement exacerbates the problem further. Examples of this include the now projected life of the B-52 aircraft to be 90+ years, the Aegis Combat system is expected to be 50 years instead of the original 30 years, and the existence of legacy software (e.g., Cobol programs) that is still in existence well beyond their anticipated end of life. The third trend of time compression amplifies this issue, and accelerates the aging process of the legacy system infrastructure.

The final trend is found in the workforce called upon to conceive, create and manage these complex systems – a workforce that has changed over time as much as the technological environment around it. Perhaps as a result of having grown up in a networked age with almost instantaneous feedback, our technical workforce is increasingly more interactive and experiential, while being more comfortable with change and a lack of comprehensive knowledge about a system. This new workforce often is more concerned with how things behave, rather than how the work [Wilcox, 2010]. One result of this trend is that systems engineers and management are being trained with subject matter that may not be relevant to the challenges that they face in the field. Those who are formally trained in traditional Systems Engineering may find themselves in a fundamental mismatch with customers and markets because of the compressed notions of time and the newer methods, processes and tools. Who is willing to perform, or pay for a thorough requirements analysis, when it is likely to be outdated

(or at least perceived to be outdated) before it is completed? There are also the challenges of a geographically, functionally and culturally distributed workforce.

1.3 OBJECTIVES

Program Goal: Transform the discipline of SE to meet the emerging challenges and increase its relevancy. The first step is to create an initial roadmap outlining the critical early research that is necessary for this transformation. To be successful, this roadmap should result in a series of funded Research Topics to begin this transformation.

1.4 VISION FOR SYSTEMS ENGINEERING

We envision a transformed Systems Engineering ability that consistently enables rapid, efficient delivery of continuously evolving capabilities while staying ahead of increasingly complex mission requirements and advancing technology capabilities. SE is a seamless part of system conception, development and sustainment through flexible, integrated infrastructure (methods, processes and tools) that is adapted to the specific needs of the environment. The size and number of text based artifacts are minimized, consistency of system representations is assured, and effective low-overhead communication is ubiquitous. Systems engineers are able to focus on thought-based SE tasks rather than mechanics. The time between need recognition and fielded capability is acceptable to the stakeholders.

Key Characteristics of Transformed SE
- Seamlessly integrated into life cycle
- Adjustable to specific needs
- Makes best use of human agents
- Supports asynchronous development
- Automated wherever possible
- Integrated (no sneaker-net)
- Knowledge based, continually evolving
- Supports analysis and decision making with automated data mining of artifacts
- Focuses on the interfaces

1.5 BENEFITS

The transformation of Systems Engineering provides significant benefits in three key areas:

1. **Challenges -** Addresses the challenges created by the critical systems trends: reduces the effects of complexity through abstraction and automation; mitigates dependency through higher quality; provides new concepts, tools and techniques to increase the competency of systems engineers and accelerate their development
2. **Relevance -** Enhances the relevance and increases the application of Systems Engineering to a wider spectrum of system development projects: effectively

integrates Systems Engineering into increasingly software-driven systems; simplifies adaptation to different project and environmental contexts; provides flexibility and agility throughout the development lifecycle; increases the probability Systems Engineering keeps up with the technology and application curve

3. **Efficiency -** Reduces the cost and schedule overhead associated with applying Systems Engineering: effectively shortens development cycles; supports rapid deployment and incremental fielding; applies advancing technologies to increase Systems Engineering efficiency

1.6 RESEARCH APPROACH

The research approach is to understand the current challenges facing Systems Engineering. Then, the state of the art in Systems Engineering, both in traditional and non-traditional uses, will be reviewed to determine the gaps between existing and desired capabilities. Approaches which have been used to stay on the curve in developing these technologies will be reviewed and examined for potential use in Systems Engineering. Based on this, a vision will be created to show how these challenges might be addressed. An integrated and modular roadmap of innovation will be created which can support this vision. This roadmap will then be reviewed in a workshop with the sponsor to review, refine and ensure that it is responsive to their needs. From this, a 3-year roadmap will be created to transform Systems Engineering. While this concludes this research topic, this should be seen as the beginning rather than the end as this should result in the creation of a set of Research Topic proposals whose funding will start the transformation process.

The primary means of transforming Systems Engineering will be to use the same elements that are creating the systems challenges: namely technology and human capabilities. The capabilities to be leveraged are the rapid advances in computation, visualization, communication and information technology. Computation performance has been approximately doubling every 24 months for the past 30+ years. This capability will be required to support simulation based Systems Engineering approaches in a number of areas including functional simulation, verification and validation; design analysis and synthesis, and data-mining operations. Visualization capabilities have exceeded Moore's Law rates in the recent past due to huge market demand in games and entertainment. These visualization capabilities are a critical element to enable humans to better understand hugely complex systems and make appropriate decisions regarding their capabilities, architecture, design and implementation. Information technology, particularly the ability to analyze data and present information in a coherent way, is a necessary tool for navigating the huge amounts of data that are created during the development or use of a system, particularly one that has been instrumented for feedback. Finally, technology is required to provide effective communication among

diverse set of stakeholders, architects, developers and support personnel both synchronously and asynchronously. While the capabilities of individual humans are not evolving rapidly, our ability to most effectively leverage these assets with technology is critical. In addition, we need to provide the means to effectively form and operate in cooperative and competitive organizations. All of this must be achieved if Systems Engineering is to ride the same curve that is driving the complexity of the systems which they are developing.

It should be noted that this research directly leverages the results of the "Evaluation of Systems Engineering Methods, Processes and Tools on Department of Defense and Intelligence Community Programs" [Turner, 2009a; Turner, 2009b] and "Investigation of a Graphical CONOPS Development Environment for Agile Systems Engineering" [Cloutier, 2009] SERC research topics. In fact, these two research topics were conducted in parallel with this research project, sharing findings and research efforts throughout.

2 RESEARCH FRAMEWORK

In an attempt to transform SE, it is best to look at the fundamental principles of the discipline to determine the first order goals for the profession. In this process, it should be possible to create a set of definitions and a framework upon which the current methodologies can be analyzed, compared and reviewed for shortcomings with relationship to the greatest needs within the practicing community. Rather than starting with a fully mature set of SE practices and attempting to optimize them piecemeal, it is likely to be more productive to start with the essential requirements and build from there. To create transformational change requires starting with the fundamentals which are described below.

2.1 DEFINITIONS

The following are the current working set of definitions used in this research [Turner, 2009b].

Systems Engineering – Systems Engineering is an interdisciplinary approach and means to enable the realization of successful systems [INCOSE, 2004].

Clockspeeds – The rate at which something evolves or changes. This can be applied to the rates at which industries, organizations, processes or products evolve [Fine, 1998].

Method (M) – A Method is a collection of inter-related processes, practices, artifacts, agents, resources and tools. A method is essentially a "recipe." It can be thought of as the application of inter-related processes, practices and tools wherein different agents use resources to create and apply artifacts to a class of problems.

Process (P) – A process is a logical sequence of steps (tasks) intended to achieve an objective. The objective achieved may be abstract (e.g. "negotiate among multiple stakeholders") and/or a composite of multiple individual goals (e.g. "Deliver a fixed-date, variable-scope system"). Performance of a step is often the responsibility of an agent, which may be a human, a device, or a software system. Performing the step may consume resources and require access to various kinds of artifacts in order to execute. Execution of a step will generally produce more artifacts. The structure of a process enables several levels of aggregation (i.e. sub-processes) to allow understanding and analysis of the process at multiple levels of abstraction in support of decision-making.

Tool (T) – A tool automates or partially automates one or more steps within a process and thereby enhances process performance efficiency.

Environment (E) - Comprises the surroundings, the external objects, and conditions or factors that influence the actions of any system entity or component, or interaction between these. The conditions can be associated with a variety of considerations (e.g., social, cultural, personal, physical, organizational, functional). The environment integrates and supports the tools and methods used on projects/programs.

A **useful** MPT is defined as one that is:

- *Relevant to the application environment:* applicable to some subset of systems within the target environment.
- *Repeatable:* sufficiently well defined that implementation is possible in a different context.
- *Likely to have significant impact:* can materially improve Systems Engineering practice in the application environment.

A **viable** MPT is successfully implementable in the target organization given appropriate and reasonable tailoring.

2.2 SYSTEMS ENGINEERING FRAMEWORK

Systems development, or just about any development, consists of the following four major activities as shown in Figure 1:

- **Value[1] -** This includes understanding an environmental context, understanding the factors which influence the creation of value and discerning how value may be created within it. Quite often, this is the domain of executive leadership within an organization. In some organizations this function is resident in marketing, in other places it may be in sales or engineering. To be successful, whoever makes these decisions should understand the total value proposition which includes customer needs, not just customer "wants", and how satisfying them brings value to the organization.
- **Conceive -** This includes the creation of a conceptual or abstract design which creates value. Architects and marketing may be involved in this activity.
- **Develop -** This includes the design, manufacture and whatever else is necessary before the system can be used. This activity usually includes engineering and manufacturing.
- **Use -** This includes the distribution, deployment, use, maintenance, and eventually retirement of the system. This activity includes sales and service and, of course, the end user.

[1] "Value" is used as a verb in this framework as are the other descriptors "Conceive", "Develop" and "Use". In this case, one is placing value on the various attributes of the environment and the outcomes that can be achieved through a deployed system.

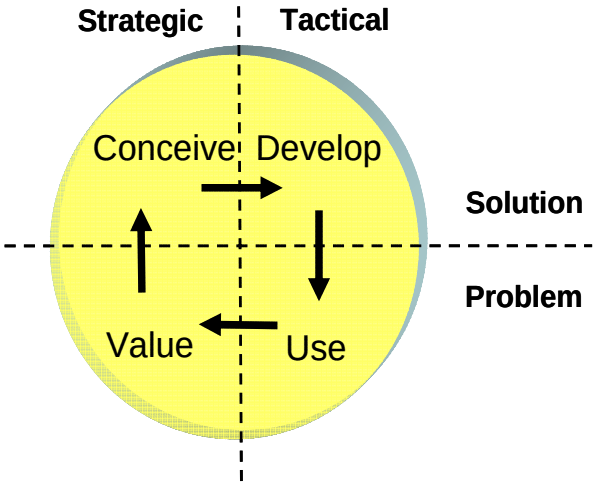


Figure 1: System Life Cycle Activities

The lower activities can be classified as being associated with the problem or application space, while the upper activities are related to the solution. The left-hand activities tend to be high level and abstract, while the right-hand activities lower-level and more tangible.

All of these activities must be considered in the system life cycle. All of these should be in the realm of the system engineer. Traditionally, these phases are executed in sequence with oral or textual documentation being used to communicate across the interfaces.

2.3 VALUE, PROJECT & ENVIRONMENT ATTRIBUTES

In general, there are no single "best practices" independent of the value and particulars of a project and the environment in which it is being developed and deployed. It is necessary to create a framework by which the most appropriate practices can be selected based on these criteria.

The following is a set of attributes which can be used to characterize an SE application to help determine the relevancy of an MPT. These attributes can be divided into the following three categories:

1. Stakeholder Value Attributes,
2. Project Attributes and
3. Environment Attributes.

For an MPT to be deemed appropriate, there should be an acceptable match in each of these categories.

Stakeholder Value: The first category relates to the overall aggregated utility for the entire system. For Intelligence Community or Military Systems, this might be the value of new capabilities delivered to the field. For commercial systems, this might be represented by some combination of ROI, market share growth, strategic advantage and customer satisfaction. These attributes include the value of schedule, cost, features, capacity, reliability/availability, maintainability, upgradeability, etc. This category determines the alignment of the stakeholder values and the potential strengths of an MPT.

Project: The second category relates to the project itself. These include the attributes of project size, complexity, duration, stability of requirements, etc. In some sense, this could create a set of relative risk/cost levels for each of the value attributes noted above. This, along with the alignment of the value attributes, determines the potential benefit of the MPT.

Environment: The final category relates to the environment in which the project is taking place. These attributes might relate to the ability to share knowledge between those who are responsible for the Valuation, Conception, Development & Use of the product/service, as well as flexibility of the organization to change its operation and processes (along with its customers and supply chain), and its human and financial resources. This category can be used to expose barriers that might make a particular MPT inappropriate for a particular organization or environment.

The following are a set of attributes in each of these three categories (the attributes that are followed by "*" are described in [Boehm, 2003]):

- **Stakeholder Value:**
  - Emergent solutions vs. project predictability
  - Rapid value creation
  - System Criticality*
  - Maintainability, Upgradeability & Extensibility
- **Project:**
  - Size & Complexity: required number of personnel*
  - Dynamism: % requirements change per month*
  - Legacy: complexity of legacy system integration issues
- **Environment:**
  - Developers*:
    - % per Cockburn level
    - Understanding of basis of project value
  - Customers: dedicated, collocated CRACK[2] performers
  - Level of Trust
  - Communication Capability
  - Culture*: thriving on Chaos or thriving on Order

[2] Collaborative, Representative, Authorized, Committed, Knowledgeable [Boehm, 2002].

3 CURRENT PRACTICE

The transformation of Systems Engineering is not so much about filling the gaps in current approaches, but rather creating a new paradigm which transcends these gaps. While it is important to understand the current state of the art to determine what has been done and is currently possible, this should only serve as a reference point and not bias the future vision such that it becomes a patchwork of point solutions to address current shortcomings. To provide a broad context for SE, this section reviews current practices in traditional environments characterized by government, military and aerospace industries, as well as non-traditional environments such as the computer industry and commercial software engineering. This section is not intended to present all the current practices that are being used today, but rather provides a high-level overview of these practices. The gaps in these practices are identified through reference to the SERC MPT Project [Turner, 2009b] and a review of the impact of value, project and environment attributes. Finally, descriptions of some recent advances and visionary proposals are described.

3.1 TRADITIONAL SYSTEMS ENGINEERING

The following describes the attributes, history and recent developments and advances in traditional Systems Engineering.

3.1.1 ATTRIBUTES

Traditional SE processes involve early and comprehensive identification of goals and requirements, concept of operations, and rigorous requirements-driven design and test. SE took root and developed building large, complex, mission critical, generally self-contained systems, where requirements changed slowly and a high value was placed on project predictability. As such, the major focus was on risk reduction and cost control to achieve a specified mission. These attributes rewarded a sequential process based on mission requirements. As all design and test was driven by and traceable to requirements, it was believed that nothing would be done unnecessarily and everything that was necessary would be accomplished. This process has served the profession well for systems and environments with similar attributes.

3.1.2 HISTORY

The term Systems Engineering traces back to the early 1940's [Schlager, 1956], saw major application in World War II [Fagen, 1978], and was first taught in its current form in the 1950's [Hall, 1962]. During these formative years, systems were primarily composed of electro-mechanical systems and largely devoid of software or microcircuits. Many computers of the 1940's were analog, passenger cars until the later 1960's were devoid of software, and the public phone system was supported by electromechanical crossbar switches well into the 1970's [Turner, 2009c]. To a great degree, one could physically see the system and its elements. The complexity of systems was limited to a certain degree by how quickly humans could physically construct them. Indeed, magnetic core memory was entirely constructed by human hands until it was superseded by integrated circuit technology (RAM) in the 1970's [Burger, 2009]. Systems Engineering has matured over the past half century often with the adoption and refinement of a number of related development processes which were originally created for software development. One of the first was the waterfall model [Royce, 1970] which was designed in 1969 to provide a structured, repeatable development process for software. This model promotes understanding requirements upfront before beginning the design and coding process. The spiral model [Boehm, 1986] was developed to provide for risk management in an iterative development process. The Vee Model which supports top-down and bottom-up processes was developed in the late 1980's and published in 1991 [Forsberg, 1991] and continues to be refined and elaborated upon [Forsberg, 2005; INCOSE, 2007]. The Vee was created to address system development issues of decomposition, definition, integration, verification and validation. The left side of the "V" represents the decomposition of requirements and creation of system specifications, and the right side of the "V" represents the integration of parts and their verification.

As time passed, the processes were refined to continue to reduce risk and unpredictability. Market forces rewarded reduction in risk and uncertainty in cycle time more than the reduction of cycle time to deployment. As issues were encountered in the system lifecycle, additional checks and balanced were added to the process. Since the market forces did not reward increased efficiency, but rather tended to work on a cost plus basis, substantial investments were not made in technology, tools and training. Hence, the processes tended to remain text-based with a focus on the creation of documentation artifacts rather than design artifacts. There were also few incentives to reduce process complexity and requirements as there are generally few personal or organizational rewards for tailoring a process down, but rather an increase in personal risk such that when something fails (as it is likely to do) the failure will be attributed to the person who approved the reduction in process oversight. This evolutionary process has resulted in Systems Engineering processes which many see as providing mainly an oversight function, rather enhancing the productivity of the systems team. The DoD5000 process is an example of the end product of this evolutionary process. The net effect on system delivery had been quite negative. For example, the time for the

DoD to procure a major system is several times longer than it was 40 to 50 years ago [NRC, 2008].

3.2 NON-TRADITIONAL SYSTEMS ENGINEERING

The following describes the attributes in non-traditional SE approaches. Then, a short history of the approaches, the impact of Moore's Law (the root of many of the critical systems trends described earlier) and the response from the computer system industry is discussed. The transformation of the processes and tools used in this industry to maintain Moore's predicted growth can serve as a model for what might be done for SE. While created in a specific domain, these MPTs have been successful in the development of systems of exponentially increasing complexity and capabilities over a sustained period of time.

3.2.1 ATTRIBUTES

Systems Engineering in non-traditional areas has been driven by a very different set of attributes than its traditional counterpart. In particular, risk reduction is generally replaced by a focus on opportunity optimization in which the objective is the maximization of profitability within an acceptable level of risk. Within a competitive marketplace, requirements may change quickly resulting in the need for agility. Risk is seen as a necessary evil which can be traded off against potential opportunities. In addition, over time these systems have become far less self-contained and more generally depend upon and live within a much larger ecosystem. As a result, these system developments have needed to support a range of different clockspeeds or rates of change.

The following example describes the development practices in one such system domain, that of computer systems, which has not only kept pace with technological change, but has in fact driven it and made the impact of Moore's Law possible.

3.2.2 HISTORY: SYSTEMS ENGINEERING IN THE COMPUTER INDUSTRY

Moore's Law is an economic pronouncement which states that the density of transistors at minimum cost per device will double every 24 months [Moore, 1965]. When Gordon Moore first described his Law in 1965, the low-cost point was 50 devices per integrated circuit. In 2008, Intel manufactured and sold microprocessors with 2 Billion devices. These transistor trends are shown in Figure 2. It is expected that Moore's Law will hold for at least the next 10 years [Gelsinger, 2008]. In addition to Moore's Law for switching devices, there are parallel "Laws" for Compute Performance (doubling every 24 months), Hard Disk Storage Cost (Kryder's Law - doubling every 24 months) [Walter, 2005], Network Capacity (Butter's Law of Photonics - doubling every 9

months) [Tehrani, 2000], and User Bandwidth (Nielson's Law - doubling every 21 months) [Nielsen, 1998].

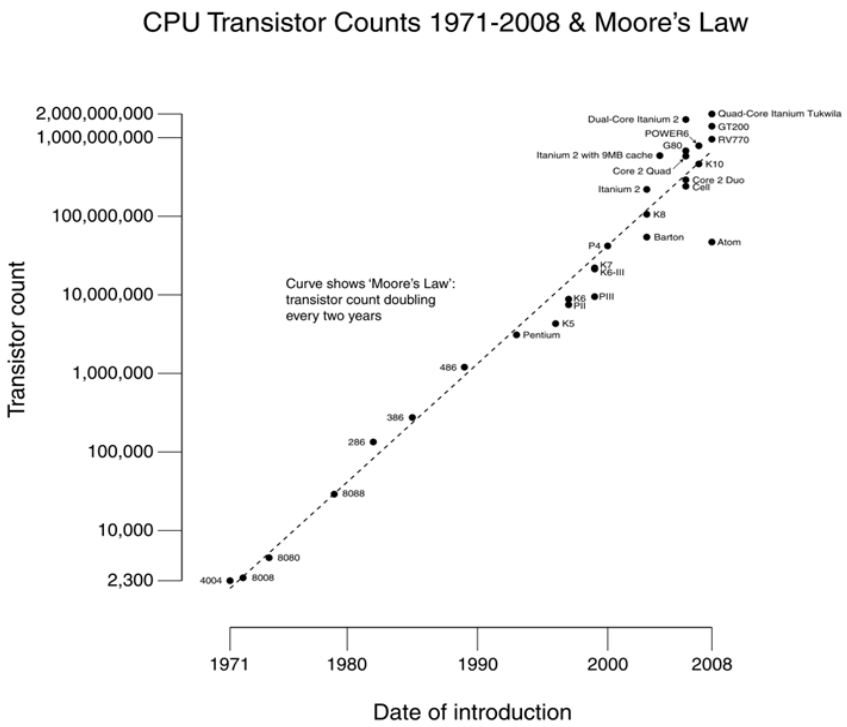


Figure 2: CPU Transistor Counts (Source: [Wgsimon, 2008])

The development of integrated circuits went through a necessary transformation in the late 1970's to sustain the delivery of technology on this exponential curve.

> "Until 1979, IC design was done by specialists who understood every aspect of the design from semiconductor fabrication, transistor characteristics, all the way up to small blocks of a maybe a thousand gates which was the limit of chip fabrication in that era. In the late 1970s this 'tall thin man' approach started to break down. Design was getting too complex for people who understood the process to do it, and the process was getting sufficiently complex to become the realm of its own specialists. Everything changed thirty years ago with the publication of Mead and Conway's book 'Introduction to VLSI systems.' Mead and Conway separated design from manufacturing by creating simplified design rules for layout, and a simplified timing model suitable for digital design. No longer was it necessary to understand every nuance of the fabrication process, no longer was it necessary to consider every transistor as an analog device. The most important aspect of this is that it meant that computer scientists could design digital chips since they no longer needed deep electrical engineering knowledge."
>
> "Design became the province of computer scientists who understood enough about layout, enough about timing, enough about architecture and enough about

test to successfully create state-of-the-art chips. Indeed, they could do so more effectively than the electrical engineers since chips were getting to be too large to do entirely by hand, and computer scientists already knew how to deal with complexity. They also started to create the first EDA (Electronic Design Automation) tools, simple layout editors, simple simulators, rudimentary design rule checkers, because their natural instincts were never to do anything by hand if you could create a program to automate it. Mead and Conway's book created a cohort of IC-literate computer scientists who went on to populate the CAD groups of the semiconductor companies and, eventually, the EDA industry once it got going."

"To see how big a difference it made, look at analog design versus digital design today. Analog design is largely done today the way digital design was done until Mead and Conway: deeply expert designers with the raw process models, raw design rules and polygons. The next big change would be the invention of Verilog and RTL synthesis that meant that computer scientists could design complex chips with almost no knowledge of how chips worked, what a transistor was, how a chip was made. This new layer meant that front end designers and back end designers were different people with different skill-sets. We seem to be on the cusp of another such layer with ESL tools starting to become much more widely used, allowing designers with very little hardware knowledge at all to create complex systems. The layer above that is software, already well-understood and with its own culture and tools." [Mclellan, 2009]

The development of these tools and processes was made possible by the development of an ecosystem which has sustained the exponential growth of Moore's Law over a period of 35 years due to the virtuous cycle shown in Figure 3 below.

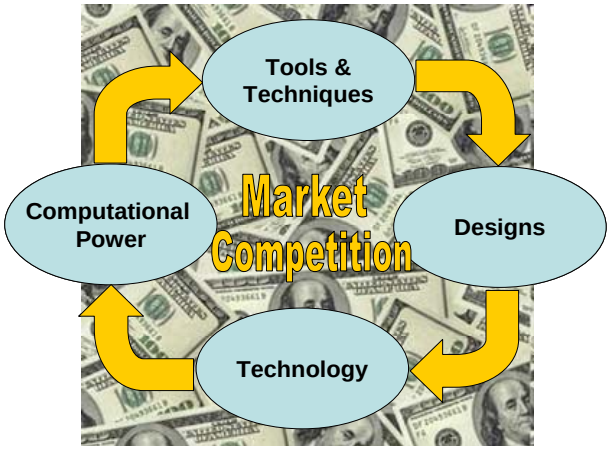


Figure 3: Virtuous Cycle Driving Moore's Law

First, the roadmap for future advancement in integrated circuit technology is well understood by the users of this technology and designs are created which do not target

the current state of the art, but rather the future state of the art that will be available when the designs are complete. Since these advancements are predictable and there is generally open communication between the silicon vendors and designers (particularly in integrated vertical vendors which supply both the means of fabrication and designs), this cooperation results in a great deal of synergy with corrective actions taking place on both the design and technology fronts. These designs are then implemented in the emerging technology which results in new computer systems which drive computational power per constant cost. This increase in computational power can then be used to support a new generation of development tools and techniques which in turn support the next generation of designs. Thus, advances in hardware and software technology not only are the result of this cycle, but they enable it.

The profits generated from the current generation of products are then invested to make the next generation possible. In the microprocessor and high-end server business, it is not uncommon that 20% or more of engineering resources are committed to the development of software tools necessary to enable this process and allow the company to stay on the curve. Companies which have delayed product introductions fail to reap the profits which are concentrated at the front end of a new product introduction and are likely not to have the resources to stay on the technology curve. In general, this either means that the company will go out of business, or move away from the development of core technology towards a less technology intensive business model. In this economic environment, only the strong survive. However, this virtuous cycle was enabled by government investment in technology development in its initial phases and at critical transition points, exemplified by the formation of SEMATECH in 1986, and with cooperation, or perhaps more accurately “coopetition”, between many of the vendors [Bonvillian, 2004].

A comparison can be made between the state of the integrated circuit development in the late 1970’s and Systems Engineering today as shown in Table 1. While the complexity, criticality, time scale and technology challenges may be similar for these two cases, the economic drivers, product requirements, scope, types of practitioners and culture are very different which tend to impede change in Systems Engineering.

Table 1: Comparison between VSLI 1970's and SE today

	VLSI circa 1970's	Traditional SE Today
Economics	Driven by TTM[3] & need to stay on the technology "curve"; Tradeoff Optimality for TTM	Driven by "cost-plus", little reward for investment in technology & training Optimize for lowest cost to satisfy contract
Products	Product families	One of a kind
Scope	Narrow: Computer Science, Electronics & Physics	Broad: Human Factors, Socio-Economic-Technical
Complexity	Becoming Unmanageable without Automation	
Criticality	Mission & Life Critical	
Time Scale	Product releases expected every 9-18 months	
Technology	Driven by Moore's Law & Software	
Practitioners	Computer Scientists, Electrical Engineers, Physicists	Multi-disciplinary, Technical & Non-technical
Culture	Technology Driven Opportunistic Automate	Process Driven Risk Averse Staff up

3.2.3 DEVELOPMENT PRACTICES

Current computer systems employ a Service Oriented Architecture in which standard electro-mechanical and software interfaces allow components and systems to evolve independently while still supporting their composition into integrated operational systems. To a large degree, the model of vertically integrated computer companies is gone. The two exceptions to this rule are IBM and Sun Microsystems. For their mainframe computers, IBM develops the applications, operating system, computer hardware and microprocessors (off the shelf standard memory and disk drive components are used). Sun Microsystems develops Java-based application middleware, the operating system, computer hardware and processors. Recently Oracle Corp. has acquired Sun Microsystems which has added database applications to this vertical aggregation. Non-critical applications generally are supported by non-vertically integrated platforms using the Wintel (Windows and Intel) or Lintel (Linux and Intel) model. In these platforms, Intel or AMD processors, chip sets and the Windows or Linux operating system are used to support applications. Web based applications are often further supported by the open source software LAMP stack composed of Linux, Apache HTTP Server, MySQL and PHP, Python or Perl scripting languages.

[3] Time to Market (TTM)

It should also be noted that at introduction most computer systems are generally not on the Moore's Law curve, but rather tend to fall above it for a new generation of servers, and then gradually fall below it with processor speed bumps and cost reductions. This discussion is focused on the development of a new generation of servers which most closely follows the development of a new system.

The development cycle of new server systems generally requires five years from conception to delivery, results in a server which is sold for five years, and then is maintained for an additional five years. Thus, a 15-year timeframe must be envisioned at the initiation of one of these programs. To support a continual supply of products, there may be three developments active at any one time: the currently supported product, a product just getting ready for introduction, and the start of the next-next generation product.

At the initiation of one of these product developments, a short Product Concept document is usually created to outline the business case and determine the product timeframe, high-level performance, capabilities, price, cost and resulting return on investment (ROI). Once this analysis is completed, a high-level Product Requirements Document is usually written to outline key capabilities. In parallel with this effort, a range of potential technologies are identified to enable these capabilities. In fact, several technologies may be determined to be viable and are designed into the product with a final technology choice made later in the development cycle. There usually is an ongoing process to determine how Moore's Law can be maintained for each component in the system. Relationships between the internal technology team and external technology vendors are critical. In parallel with this effort, processor developers work on potential computational and memory pipelines to support the required performance capabilities and use the available transistors enabled by the new silicon fabrication technology. These investigations have been enabled through performance modeling using instruction and address traces from existing applications and projections on the composition of future applications.

A number of rapid conceptual prototyping activities are initiated incorporating the targeted technologies and processor concepts to bridge the communication gap between the marketing and engineering organizations. Unlike traditional Systems Engineering programs, this is not done through text-based requirements documents, but rather is done through visual models and/or simulations. It is challenging work as the technology projections are fuzzy, the market requirements in 5 years are questionable, and the high-level concepts do not contain a great detail of engineering analysis. It is here where human capability is critical. Breakthrough products generally are the result of well-communicating teams led by a visionary leader who understands how value is created in the marketplace, can see the integrated possibility of the system, and understands how best to balance technical and product risk and opportunity.

It is generally at this stage in the product development that a full team is ramped up and project funding is put in place. Cost, performance and reliability/availability models are created and monitored throughout the development process to ensure that the product can be developed and deliver the targeted ROI. If it is seen during the development process that this is no longer the case, the project may be changed in a significant way or canceled. Competing projects may be created with the expectation that some of them will be canceled. The notion of Value Based Engineering is not seen as something that it optional or novel, but rather is central to the system conception, development and deployment process.

With the ability to support billions of transistors in a single integrated circuit die, the development of a microprocessor in many ways takes on most of the attributes of the development of a complete computer hardware system. The development of a processor takes place simultaneously in a number of different domains. First, there is the architectural domain which creates the representation of the processor as seen by the software and external server system. It should be noted that this architecture may and should exist over several generations of system development. For example, continuity in the processor instruction set is critical for the multi-generational support of applications.[4] In addition, there are architectural models that are instrumented with performance information that can be used to characterize their performance on existing and future applications. It should be noted that existing processors are generally richly equipped with a multitude of performance counters that can be used in live computer systems to provide information for future designs. Architectural models are created and supported which are provided to software and system developers.

The second area is the logical design of the processor. These designs are represented in behavioral models and often interchangeable high-level descriptions of the actual design. This modular approach provides the capability to create verification test suites that can be executed at the maximal rate while adjusting the fidelity of the representation of the processor to the desired degree. In addition, execution of these models can be used to validate the logical and performance operation of the architectural models. Design is done at the very highest possible level. In many cases, a few lines of design code can be synthesized to create hundreds of thousands or millions of transistors along with the mask level polygons necessary to physically implement them. It is here where design processes and tools have made the use of the capabilities of Moore's Law possible. Devices at the transistor and interconnect level are characterized and modeled. These devices are then used to create standard cell libraries and memory arrays which are likewise modeled and characterized. These are then made available to synthesis tools which follow sets of rules to determine how they can best be used to support the complex logical constructs specified by high-level languages. In

[4] Note the failure of HP and Intel in the development of the new Itanium instruction set. It is ironic that AMD exploited this misadventure by valuing and supporting the i86 instruction set more effectively than its originator.

addition, there are tools which ensure that the billions of polygons are drawn correctly through automated rule checks, that timing constraints are met through each of the billions of possible signal paths and that reliability is ensured. With a device of this complexity, it is assumed that there will be faults both in manufacture and in normal use. For signaling, error correction codes are put in place. For faulty transistors, these may be mapped out and redundant devices may be used. In addition, the integrity of data must be preserved, so none of these failures can be allowed to propagate back to the final data store. Tools are available to create tests that can ensure that any manufacturing or operational faults are detected and that appropriate action is taken. Given the enormity of the design, there may be a huge number of false positives in the design rule tests, but these are accepted and fixed as to prove them false is usually not a pragmatic approach.

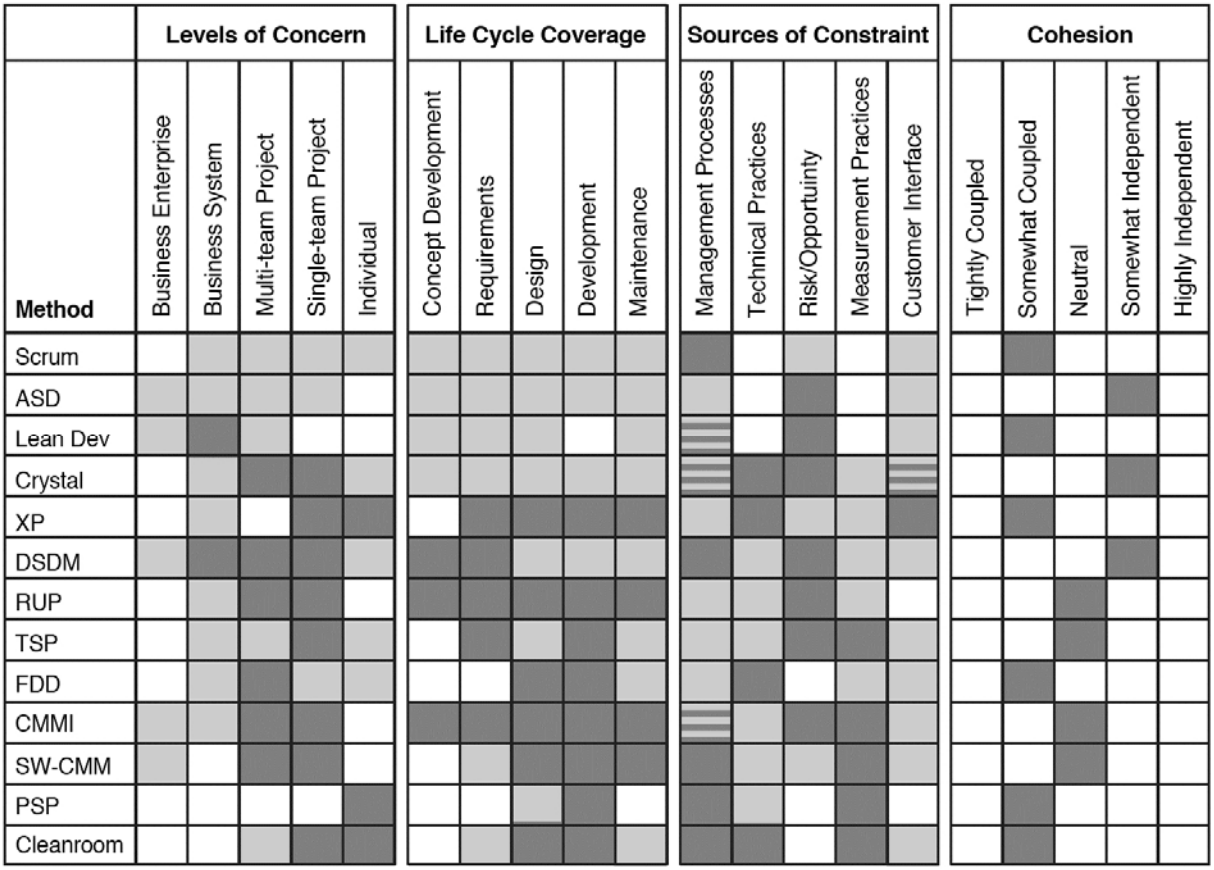
Finally, there is the physical design of the processor which includes floor planning, clock distribution (time of flight is a significant fraction of the total clock period), electrical, mechanical and thermal design. This is an area which involves state of the art modeling technology and tools, many of which are physics based. This modeling allows for interactive design and often results in the building of virtual or sometimes physical prototypes which are instrumented and measured to validate the modeling process.

It should be obvious from the above description, that server and processor design does not follow the usual SE process of sequential, requirements driven design. While the time frame for development may be long, and the need for perfection is high (design level flaws could easily result in catastrophic corporate failure), the penalties for being late to market or for not adapting quickly to changing market requirements can be just as high.

3.3 SOFTWARE ENGINEERING

A substantial amount of development has taken place in the past ten years in the area of agile Software Engineering. As has been noted earlier, some of the current standards in SE methods, processes and tools, were first developed for software development and then were migrated over to the more general practice of SE. While many of the approaches may be transferrable, efforts need to be made to ensure that they have the breadth and the generality necessary for SE. In “Balancing Agility and Discipline” [Boehm, 2003] a comprehensive list of agile software methods with attributes are described and summarized in
Table 2 (dark grey indicates full applicability, light grey partial applicability and stripes mean a range of values). In addition, the book describes “home grounds” for agile software processes, which serve as a guide to determining the applicability of each method to a given environment.

Table 2: Agile Software Methods (Source: [Boehm, 2003])



3.4 IDENTIFIED GAPS

Each of the areas described earlier have their own home grounds of applicability as shown in

Table 3. For example, traditional Systems Engineering approaches work well for programs with relatively static requirements and require mission-critical quality levels. The methods used in the computer industry enable these systems to stay on the technology curve, and drive the curve of technological innovation and capabilities, yet are rather domain specific. Software engineering agile methods can produce rapid results, yet are often limited in their scalability. However, none of these methods consistently take into consideration the human element in the operation of the system nor address the long-term impact of the system on its environment. While none of these methods directly provide solutions to the challenges facing Systems Engineering in the 21st century, there certainly are elements in each of these that are relevant to the problem at hand.

Table 3: Comparison of SE Approaches

	Industry	Optimal	Leverage	Shortfalls
Traditional SE	Defense & Aerospace	Large, mission critical	Formal Reviews, Text based documentation	Inflexible, overwhelmed by complexity
Agile Processes	Commercial Software	Small, opportunistic	Small, integrated teams, constant communication	Inability to scale, mission critical issues
Automated	Electronics: Computers, Communications & Entertainment	Large, mission critical	Technology, tool automation	Domain specific

Given the breadth and depth of the challenge, it may be difficult to determine where to start to develop a solution. The approach that is taken here is to identify the gaps which have the greatest impact on the success of Systems Engineering and then provide solutions to address these. Important focus areas are those in which advances can reduce risk and/or reduce the time, effort and cost required to achieve a desired outcome. Three different approaches are described below to identify these gaps. The first is to review the work that has been done on the SERC "Evaluation of Systems Engineering Methods, Processes and Tools" task [Turner, 2009a] and summarize the major gaps identified in this work. The second approach is to review how the stakeholder value, project and environment attributes of a system impact the ability to do agile development. Third, there is a short review of some of the areas which provide opportunities to improve the capabilities of rapidly deploying systems. While these may not have been identified as gaps by the former two means of analysis, they are nevertheless areas which may be rewarding for innovation. The results from each of these approaches are then summarized as a list of critical gaps to be addressed by the transformation of Systems Engineering.

3.4.1 Results from MPT Project

During 2008 and 2009, the SERC conducted research under the topic "Evaluation of Systems Engineering Methods, Processes and Tools (MPTs)." The results, published in two reports [Turner, 2009a; Turner, 2009b] provide insight into the current use of MPTs by systems engineers operating in environments similar to the sponsor, identify gaps between MPT capability and SE needs, and pilot ways to better evaluate MPTs and to show their interrelationships.

Interviews of sponsor engineers and managers, determined four key challenges:

1. **Requirements** - Changing requirements priorities and/or emerging requirements
2. **Stakeholders** – Obtaining useful stakeholder input and dealing with conflicting stakeholder requirements
3. **Sustainment** – Conflicts between developing new capabilities and supporting a currently deployed system
4. **Integration** – Integrating independently evolving components into a larger interoperable system

The MPT project surveyed 116 engineers with significant systems and software engineering experience in agile development across a wide range of commercial and defense industries to investigate the MPTs used to address these key challenges. As part of the survey, respondents were asked to identify the three or four Systems Engineering areas which they believed to be in need of new MPTs. Of the 102 respondents for this question, the top five areas identified were: decision management; stakeholder requirements definition; requirements analysis; architecture design; and, life cycle model management. Table 4 illustrates the results.

Table 4: Critical areas identified as needing new MPTs (Source: [Turner, 2009a])

Answer Options	Response Percent	Response Count
Decision Management	36.3%	37
Stakeholder Requirements Definition	35.3%	36
Requirements Analysis	32.4%	33
Architectural Design	30.4%	31
Life Cycle Model Management	27.5%	28
Risk Management	24.5%	25
Integration	23.5%	24
Acquisition	19.6%	20
Project Planning	18.6%	19
Project Assessment and Control	18.6%	19
Verification	17.6%	18
Measurement	16.7%	17
Configuration Management	15.7%	16
Validation	15.7%	16
Project Portfolio Management	11.8%	12
Implementation	10.8%	11
Quality Management	9.8%	10
Information Management	8.8%	9
Infrastructure Management	6.9%	7

Answer Options	Response Percent	Response Count
Human Resource Management	5.9%	6
Operation	3.9%	4
Maintenance	3.9%	4
Disposal	2.0%	2
Supply	1.0%	1
Transition	1.0%	1

The MPT research developed an approach to illustrate how various MPTs related to each other, to the key challenges, and to a number of themes identified within the survey responses. The approach, referred to as a bridge diagram, is illustrated in Figure 4.

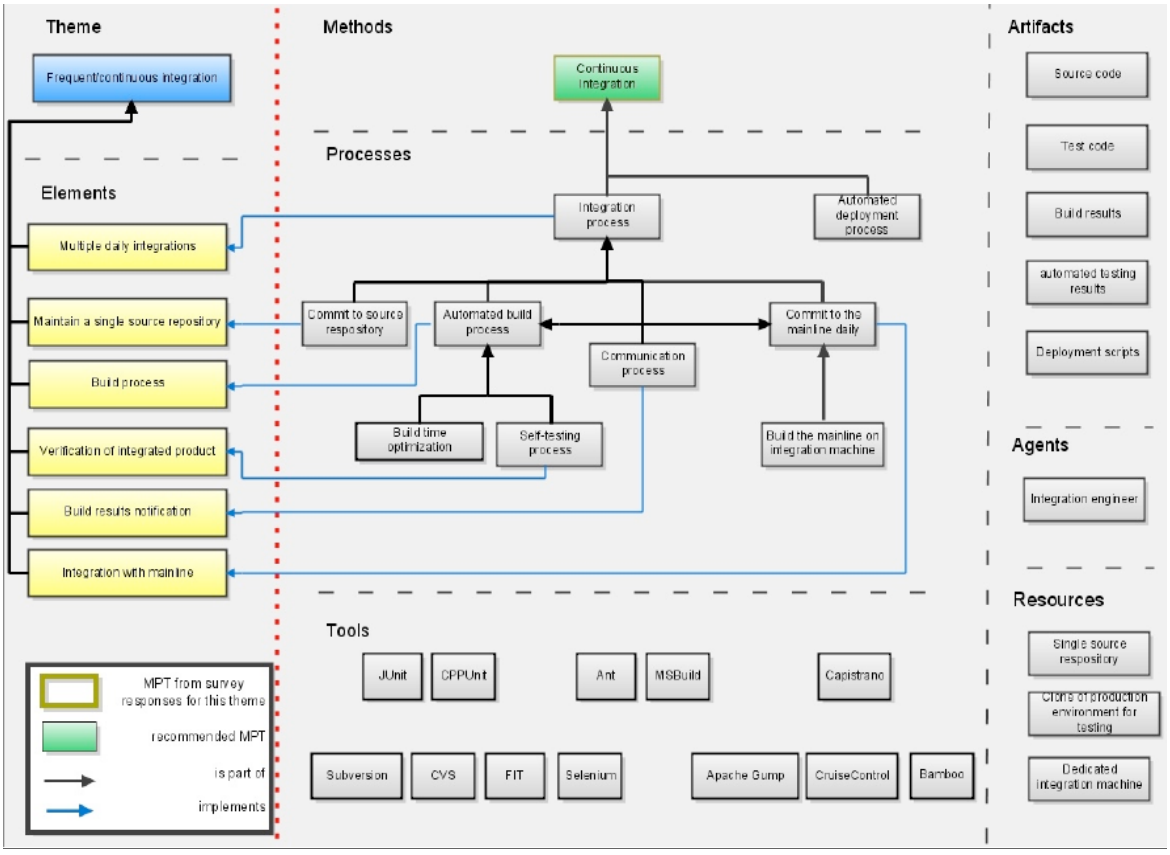


Figure 4: Bridge Diagram for Continuous Integration (Source: [Turner, 2009b])

The MPT team also successfully applied micro-process modeling techniques to several MPTs to determine the practicality of using more formal methods for the *in vitro*

analysis and improvement of MPTs. An example of one of the models is shown in Figure 5.

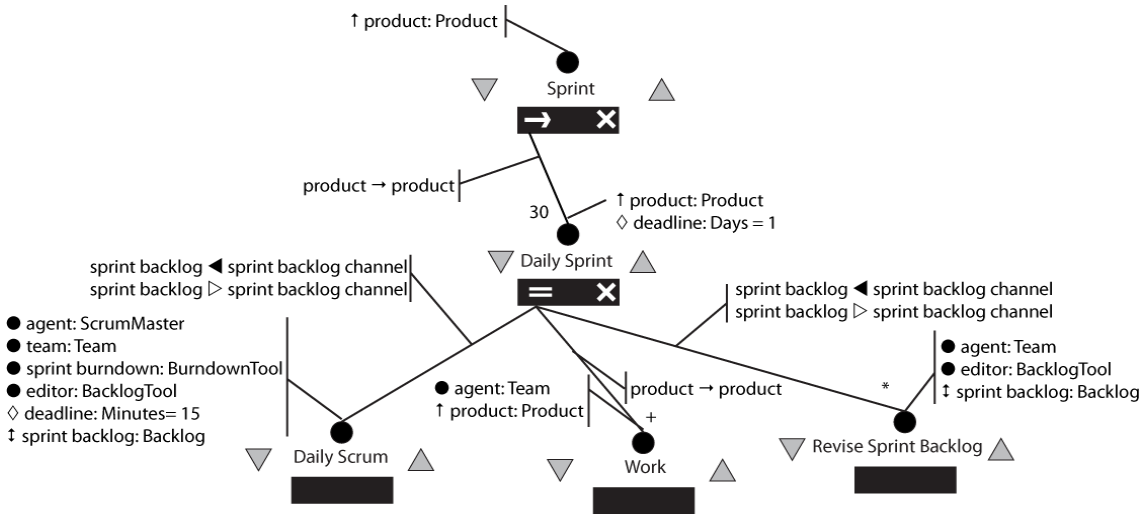


Figure 5: Elaboration of the Sprint step (Source: [Turner, 2007b])

The work accomplished by the MPT team establishes some of the essential groundwork necessary for transforming SE. It supports the understanding of key gaps in current Systems Engineering capability, identifies the impact of the critical trends on practitioners, and provides a way to describe MPTs both hierarchically in relationships and formally in an executable manner.

3.4.2 Review of Attribute Impact

In this analysis, each of the attributes in the areas of Value, Project and Environment are pushed beyond the home ground for agile development and the resulting dominant failure mechanisms are identified as the gaps which need to be addressed. An information processing paradigm will be used to represent the necessary work flow which results in the transformation of information through the four activities of Value, Conceive, Develop and Use. If there were only a single person understanding the value proposition, creating a concept design, developing and then using the system, the communication challenges would be minimized. However, while the use of a number of people with a broad range of specialized skills greatly increases the scale of what can be accomplished within a given timeframe, it also creates a communication challenge. In the case of parallel computation, increasing the number of processors reduces the total time spent computing, but increases the communication overhead. At some point, the communication overhead becomes dominant and the fallacies of the mythical man month are made evident. The gaps noted below are classified as being communication or computation related. The computation related gaps are those in which the complexity of the problem is beyond a human's capabilities to successfully cope with it. Finally, there is a challenge in latency reduction which generally drives the use of

automation to reduce the amount of human effort that needs to be expended to accomplish the task in question. Thus, the methods, processes and tools provide communication, computation (for analysis and decision making), and automation to provide agile development and rapid fielding capability.

3.4.2.1 Value

Emergent solutions vs. Project predictability: There is a clear tradeoff between the desire to allow for spontaneous innovation in a program and having a predictable outcome. Clearly, plan-driven approaches are best suited to foster predictable results, but these may well not be the results that create the most value. However, it may be possible to create an environment and supply the methods and tools which reliably provide innovative results, although it is not known a priori what these results may be. In general, the greater the potential reward, the greater the project risk. This is likely to be very unsettling to those who are uncomfortable with uncertainty. There are two challenges here. The first is the ability to determine the value of the risk and reward factors. The second is the ability to successfully communicate the current status of the development and the risk to project predictability. It is necessary to satisfy both of them if one is to be able to rationally determine how to balance these opposing forces. The former is the area of greatest gap in tool availability.

Rapid value creation: An iterative approach, with timesteps on the scale of what is considered to be "rapid" and development of sufficient degree to constitute value, is clearly necessary to achieve this goal. The primary challenge is to reduce the latency through a life cycle including all the work necessary to deliver a system at the required quality level. The reduction of latency will require improved communication, analysis and automation capability. Achieving this requires a complete tool suite supporting all three of these capabilities.

System Criticality: While plan-driven approaches have traditionally been employed to reduce risk in project failure through a hierarchy of review cycles, there are many ways of achieving this goal by less human and time intensive means. Rather than providing validation and verification at the end of a design cycle, it should be applied throughout the development process. Techniques such as Test Driven Design (TDD) and Continuous Integration are critical to achieving high quality systems within a minimum amount of time. Analysis tools, such as code coverage tools along with determination of the potential impact in the exposed areas, are necessary to assist in making the appropriate risk vs. time to deployment decisions.

Maintainability, Upgradeability & Extensibility: These attributes can generally only be achieved through the development of a suitable architecture, and support and upgrade plan. The use of a Service Oriented Architecture or Product Line infrastructure can assist in this effort such that the desired services can be deployed without having to independently create a new infrastructure. The need for such an architecture is independent of the type of developmental methods that are used to create it. Tools can

assist in the process of creating such an architecture and upgrade plans, but there is no substitute for its existence.

3.4.2.2 Project

Size & Complexity: Size and complexity generally determine the number of personnel that are necessary to staff on a project. This has a direct impact on the amount of communication that needs to take place in the project which directly affects the ability for a method to scale. This is a major limiting factor for agile methods that rely on informal information exchange between capable individuals. For example, it has been noted that the method of Scrums is generally limited to teams of ten individuals or less and that there is no conclusive evidence that "Scrums of Scrums" works effectively. [Turner, 2009a]. The major challenge is one of communication. In addition, analysis and automation capabilities can be used to reduce the effective scale of the project, reducing the number of personnel employed and thus the communication overhead.

Dynamism: % requirements change per month. This is one of the most significant factors noted in the industry surveys mentioned above. There are a number of contributing factors that enable requirements churn to disable projects. The first issue is that the change in requirements needs to accurately reflect what is necessary to create the desired additional value in the deployed system. This requires that there is a means to accurately interpret the needs of the customers and transform these into a system concept along with the related requirements. Often times the customer may not know what they want until they see or experience it. The use of rapid prototyping, either virtual or physical, may be necessary to accomplish this goal.

The second issue is that changing requirements result in the need for communication and stabilization time such that everyone on the project is made aware of and understand the impact of the new requirements or direction. If the time between requirements change is less than the communication and stabilization time, then the project will cease to make forward progress.[5] There are two ways in which this problem can be addressed. The first is to ensure that the communication and stabilization time is well understood and changes are not made more frequently than can be handled by the system. This requires appropriate policies and skills in project management. More significantly, the communication and stabilization time needs to be decreased. This can be achieved through a variety of processes and tools. For example, only those who need to be notified of the changes should be notified. This reduces churn to the absolutely minimal amount. Next the changes need to be conveyed in the most meaningful way to each person on the program so that they can quickly determine its impact and act accordingly. Again, tools can be used to accomplish this. This is an area in which

[5] In this case, it may well be better that the development team does not listen to the desired changes and makes forward progress rather than fall into a state of learned helplessness in which productive work ceases.

model-based capabilities with automatic notification and advanced visualization are generally far superior to text based requirements methods.

Legacy: Systems with substantial legacy constraints often impose great challenges in the area of system integration and ultimately realized system performance and value creation. These are challenges regardless of the type of development method that is being used. However, it may well be that there are rapid changes in the legacy environment outside of one's control which effectively results in dynamism in requirements. In this case, instrumentation of the legacy system may be necessary to properly characterize it to provide the information necessary to determine the requirements for the deployed system to deliver the desired value. Model based development may be required to facilitate the translation of this information into the development process.

In addition, flexibility in the systems operational environment which includes the interfaces, application modularity, and human interfaces and governance might well dictate the rate at which the system can be changed and still achieve its desired effects. These issues tend to reduce the agility of the overall system. Some of this can be mitigated with improved user training tools and facilities. Interactive simulations of the actual use experience are generally superior to text based documents.

3.4.2.3 Environment

Developers: The overall competency level of the developers and their understanding of the basis of project value have a large impact on their ability to work independently. Often agile methods depend on informal communications between a small number of developers which can cause this to be a major limiter on the applicability of these methods with less capable teams. Less capable teams, all else being equal, are generally less agile, but this can be partially compensated for with appropriate processes and tools. For example, while text based documentation may not be effective to represent complex systems, model based representations might provide the desired degree of information more intuitively to the user.

Customers: Of particular importance are dedicated, collocated CRACK (Collaborative, Representative, Authorized, Committed, Knowledgeable) performers [Boehm, 2003]. Understanding the value proposition for a system is critical to its success. In agile development processes which depend upon informal communication, it is critical that these are dedicated, collocated individuals such that communication can be continuous through the development process as the work is being determined on a very fine grained basis. However, it should be possible to increase the granularity of this work specification up to the size of that actually being deployed. This will require more formalized means of communication. Again, it is critical that the intent of the changes is accurately communicated to each member of the development team and validated with the customer. While face to face communication is obviously best, technology (such as high-definition tele-presence systems) can be used to enhance the capabilities of remote

communications. Model based systems which can produce multi-modal, multi-sensory feedback are most useful for this. However, the competency of the customer or the proxy for the customer to convey the needs of the system is a critical factor for which no amount of methods, tools or technology can provide a substitute. However, this is a limiting factor for both plan-driven or agile development methods.

Level of trust: Trust is a vital component of any high performing team. In fact, it may be seen as the foundational element [Lencioni, 2002]. Trust and empowerment are especially important in agile developments which rely upon informal means of communication and personal initiative. However, one could argue that transparent processes which enable rapid feedback are able to increase trust as this can be based upon verifiable fact. Tools and processes can be created which facilitate the rewards for teamwork and remove uncertainty in the state of deliverables which reduces the level of mistrust. Trust is generally the precursor to empowerment. Ultimately lack of trust and empowerment will have negative impacts on any development, whether plan-driven or agile.

Communication Capability: To some extent this determines the size of the effective team and includes the ability to communicate within and between marketing, architecture, development, test, service, sales and customers. Communication depends upon trust or else it is not possible to have the necessary constructive conflict [Lencioni, 2002]. Many agile methods depend on informal communication which involves voluntary communication between a variety of parties. Unless the cost of communication is low, it is unlikely to happen and the agile process will not be effective. Rather than depending upon informal communication, it is possible with processes and tools to create the means of structured communication which is a byproduct of the development process.

Culture: Does the culture thrive on chaos or order? It is generally believed that cultures which thrive on chaos are well suited to agile development and those that do not need to work within a plan-driven environment. However, one can certainly find plan-driven environments that are chaotic and agile ones that are disciplined. Agile environments in which methods and tools are employed to give personnel instant feedback on their progress produce fewer surprises and thus can be very orderly environments. Plan-driven environments which deploy a waterfall process which results in integration surprises that have been months or years in the making can be extremely gut wrenching and chaotic. Agile environments generally require that people are thoughtful, innovative and are open to change. However, these are generally the characteristics of productive workers in any discipline. It should be noted that an organization will need to be willing to accept change if they are to take advantage of the results of a transformation of Systems Engineering.

3.4.3 Additional Areas of Opportunity

In addition to the areas mentioned above, there are other opportunities to improve the capability to rapidly deploy effective systems. Many of the areas noted above relate to the ability to communicate, analyze and automate development activities to improve the quality of the life cycle while compressing its duration and cost. However, more can be done. For example, the flexibility of the system to adapt to future change is critical to enable the rapid addition of new features and capabilities once the system is deployed. In addition, the system could be engineered to be intelligent such that these changes can be made to be self-adaptive. Likewise, the system needs to be developed taking advantage of emerging technologies and subsystems, while determining when to obsolete existing ones. It is critical to determine how best to ensure that the system provides the necessary levels of service availability and security, while retaining its flexibility. Finally, it is necessary to appropriately incorporate the human element into the system ensuring that the strengths of human capabilities are leveraged and integrated with the strengths of technology.

3.4.4 Summary of Gap & Opportunity Areas

The following is a compilation of the gap areas described above:

- **System requirements** - creation, validation, prioritization, resolution of conflicting requirement, managing changing and emerging requirements & decision making; in particular the creation of a collaborative environment that facilitates tradeoff resolution and creation of a mutually understandable description of the desired system concept
- **Low-Overhead Communication** – the ability to provide the essential communication to keep a large organization synchronized throughout a system lifecycle with a minimum amount of overhead to provide scalable agility
- **Architectural Design Support** – processes and tools to support the development of an architecture which can support the attributes of maintainability, upgradeability (flexibility) and extensibility, along with reliability, availability, security and other emergent properties
- **Risk/Opportunity Management** – tools which can assist in the assessment of program risk and value creation to allow for the proper tradeoffs between these competing goals based on the capabilities of the organization and the challenges of the system under development
- **Verification & Validation** – an integrated set of processes and tools which can provide verification and validation throughout the lifecycle process
- **Legacy Integration** – the capability to monitor and characterize the current legacy system to ensure that the addition of new applications and services have the desired capabilities, and the ability to integrate independently evolving components into a larger interoperable system

- **Human Aware/Self-Adaptive** – the capability to optimize the use of humans in the system to take advantage of self-adaptive human capabilities

The following is a compilation of the opportunity areas described above:

- **Complexity Handling Capabilities** – tools and techniques which leverage technological advances in computation, visualization, information technology and communication to provide Systems Engineering with the capability to manage ever increasing system complexity thus keeping SE on the curve
- **Cycle Time Reduction** – a suite of processes and tools, including those noted above, which can increase the quality of the systems while compressing latency through the life cycle; these include tools which not only accelerate new development, but also eliminate unnecessary work by facilitating reuse and providing correct by design construction

3.5 Recent Developments and Future Directions

It is evident to many that the future of SE relies upon the leverage of computational and visualization technology to augment the skills of human designers. While model and simulation based technologies are not a panacea, they are seen by many as the only practical means by which to address the emerging challenges to the discipline. Numerous recent visioning efforts by INCOSE [Friedenthal, 2007; Crisp, 2007], DARPA [Eremenko, 2009], DDR&E [Carlini, 2009; SSCI, 2009] and Lockheed Martin [Watson, 2009] have come to the conclusion that Model Based Systems Engineering (MBSE) is the future of SE.

3.5.1 Current State of MBSE

High-level modeling at the system level has been enabled with the development of new languages such as SysML and other new tools. Some of the new methods and tools include [Estefan, 2008]:

- IBM Telelogic Harmony – SE
- INCOSE Object-Oriented Systems Engineering Method (OOSEM)
- IBM Rational Unified Process for Systems Engineering (RUP SE) for Model-Driven Systems Development (MDSD)
- VITECH Model-Based System Engineering (MBSE)
- JPL State Analysis(SA)
- DORI Object-Process Methodology (OPM)

While there are a number of available methods, many of them are based to some degree on the aforementioned waterfall, spiral or “Vee” models. This is not unexpected as these

represent the predominant methods for traditional Systems Engineering, which is the major market for these tools.

MBSE is clearly in its infancy as the use of it in SE is generally the exception rather than the rule. At the 2010 INCOSE International Workshop, the state of the MBSE practice was put into context. Over the last two years there has been an increased awareness of the practice of MBSE, there has been a notable increasing level of interest from academia, and the MBSE tools and practices have continued to mature. The MBSE initiative is very much a grassroots movement and is one that is being met with great enthusiasm across industry and academia. It is still very much an emerging practice and there is a great deal of ongoing experimentation in the field. However, some larger scale applications have been successful. It is thought that the current state of MBSE is analogous to the early stages of MCAD/ECAD [Friedenthal, 2010a].

Looking back as far as 2006, practitioners leading the adoption of MBSE were already taking advantage of the availability of advanced automated methods and techniques [INCOSE, 2007]. For example, acquirers were developing black-box models to define the capabilities of their systems, and then having suppliers respond in kind with their own detailed white-box models supporting these requirements. The acquisition process thus involved the determination of the model and proposal which best satisfied the modeled requirements.

The growing community of MBSE practitioners has demonstrated a significant amount of excellent work that is ongoing in pockets across different domains. Some examples that have been cited include:

- The Computational Research and Engineering Research Tools and Environments (CREATE) program has invested significant resources to create computational engineering tools to support the design of air vehicles, ships, and RF antennas [Carlini, 2009].
- The DoD training community has taken the leadership role in adopting emerging modeling and simulation (M&S) tools in the areas of games, and virtual, mixed, and augmented reality. For example, the DARPA RealWorld capability allows the user to create an operationally relevant 'game' representing an actual geophysical location in no more than four steps. The resulting virtual environments are extremely realistic and have been extensively used for training, mission planning, and rehearsals [Carlini, 2009].
- Physics-based simulation, applied to engineering design, can have substantive impacts on rapid capability fielding programs by reducing the number of test-build-test cycles required to converge on a workable solution. This approach was used by the Department of Defense with P-3 sensor integration, and also by Goodyear in the design of their tires [Carlini, 2009].
- NASA is implementing a Simulation Based Acquisition (SBA) program to facilitate the realization of the Vision for Space Exploration. As part of their SBA program, modeling and simulation (M&S) of systems and their environments has

been identified as a critical component in order to foster better informed, more timely, and more defensible decisions throughout the acquisition life cycle [NASA, 2005].

- A special edition on Model Based Systems Engineering was published in the December 2009 edition of INSIGHT, reinforcing the increased level of industry and academic activity in the practice of MBSE.

3.5.2 INCOSE VISION 2020 AND MBSE

The INCOSE Systems Engineering Vision Working Group developed a 2020 vision for the practice of Systems Engineering, providing a roadmap to guide practitioners and researchers [INCOSE, 2007]. They identified Model-Based Systems Engineering as the future of Systems Engineering given the "continued evolution of complex, intelligent, global systems that exceed the ability of the humans who design them to comprehend and control all aspects of the systems they are creating". In order to respond to this growing need, new tools need to be identified and developed. Virtual development tools will reduce the need for physical prototypes and assist in the design, engineering, implementation, test and evaluation, and operational support of the system being created. Workflow management tools will assist in collaboration and knowledge sharing efforts. The key characteristics of MBSE in 2020 include [INCOSE, 2007]:

- Domain-specific modeling languages and visualization that enable the systems engineer to focus on modeling of the user domain
- Modeling standards based on a firm mathematical foundation that support high fidelity simulation and real-world representations
- Extensive reuse of model libraries, taxonomies and design patterns
- Standards that support integration and management across a distributed model repository
- Highly reliable and secure data exchange via published interfaces.

As part of the 2020 vision, the INCOSE Systems Engineering Vision Working Group values the concept of cross-domain model integration as depicted in Figure 6. It is envisioned that the integrated capabilities shown will dramatically increase the application of MBSE to support domains such as marketing research, decision analysis, integration with biological system models, and environmental impact analysis.

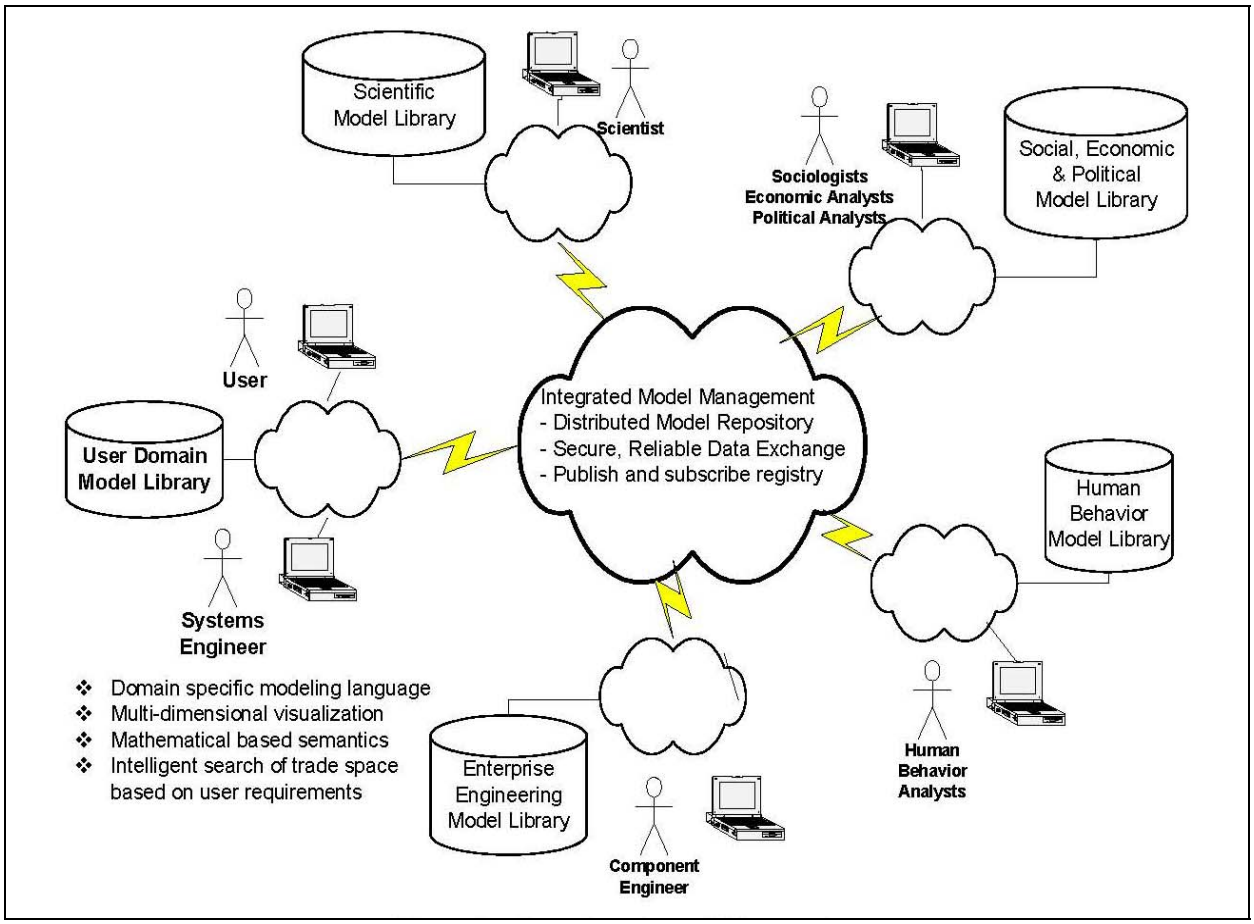


Figure 6: Cross Domain Model Integration (Source: [INCOSE, 2007])

Significant advances have been made in the adoption of Model-Based Systems Engineering. In order to facilitate these advances, an MBSE working group initiative was pioneered within INCOSE during the 2007 International Workshop [Dee, 2009; Friedenthal, 2007; Friedenthal, 2010a; Friedenthal, 2010b]. The purpose of the MBSE initiative is to integrate the four pillars of Systems Engineering, that is: the requirements view, the behavior view, the architecture view, and the parametrics view. To achieve this goal, the MBSE initiative will [Dee, 2009; Friedenthal, 2010b]:

- Evaluate new/existing methodologies & languages (SysML, UML, etc.)
- Identify areas for improvement in the current state of practice
- Promote collaboration and knowledge sharing across the wider Systems Engineering community
- Advocate appropriate methodologies
- Promote, advance, and institutionalize the practice of MBSE to attain the MBSE 2020 Vision through broad industry and academic involvement in: research; standards; processes, practices, and methods; tools and technology; and, outreach, training and education

The MBSE initiative is organized into two main areas: technical activities, and challenge teams. Technical activities involve projects that aim to investigate and develop the technical aspects of modeling, irrespective of the application domain, whereas, challenge teams involve independent projects that investigate and demonstrate the use of MBSE across application domains [Dee, 2009].

3.5.3 DARPA META PROGRAM

The goal of the DARPA META program is to dramatically shorten the design, integration and verification time for complex defense systems. The META program is focused on "using model-based design methods for cyber-physical systems far more complex and heterogeneous than those to which such methods are applied today; to combine these methods with a rigorous deployment of hierarchical abstractions throughout the system architecture; to optimize system design with respect to an observable, quantitative measure of complexity for the entire cyber-physical systems; and to apply probabilistic formal methods to the system verification problem, thereby dramatically reducing the need for expensive real-world testing and design iteration." [Eremenko, 2009]

The following are the META research focus areas [Eremenko, 2009]:

- Develop a practical, observable metric of complexity for cyber-physical systems to enable cyber-vs-physical implementation trades and to improve parameterization of cost and schedule;
- Develop a quantitative metric of adaptability associated with a given system architecture that can support trade-offs between adaptability, complexity, performance, cost, schedule, risk, and other system attributes;
- Develop a structured design flow employing hierarchical abstraction and model-based composition of electromechanical and software components;
- Develop a component and manufacturing model library for a given airborne or ground vehicle systems domain through extensive characterization of desirable and spurious interactions, dynamics, and properties of all constituent components down to the numbered part level; develop context models to reflect various operational environments;
- Develop a verification flow that generates probabilistic "certificates of correctness" for the entire cyber-physical system based on stochastic formal methods, scaling linearly with problem size;
- Apply the above framework and toolset to design, manufacture, integrate, and verify an air and/or ground vehicle of substantial complexity 5X faster than with a conventional design/build/test approach.

META makes comparisons between defense aerospace programs and those for the automobile and integrated circuit industries over the past 40-50 years. The belief is that

the integrated circuit industry is developing chips with exponentially more complexity in constant design time (18-40 months) and the automobile industry is doing the same while reducing the development time from approximately 5 years to 1.5 years today as shown in Figure 7.

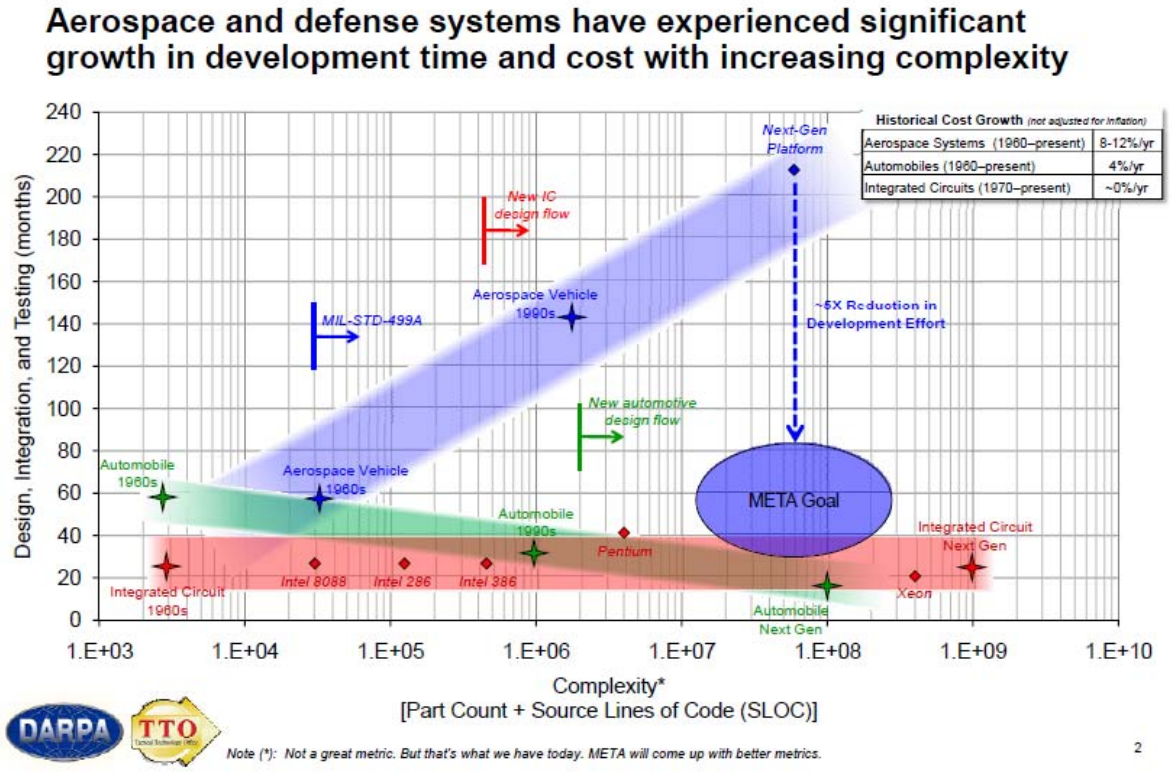


Figure 7: Development Times for Aerospace, Integrated Circuits and Automobiles (Source [Eremenko, 2009])

The capabilities shown for the automobile and integrated circuit industries are enabled by the fact that they exhibit a great deal of reuse in their design. The development times for all new microprocessors and automobile platforms are actually longer by a factor of 2-3 times or more than is shown above. The development of a microprocessor with a new instruction set could take 5-10 years to deploy. For example, the Itanium processor was conceived by HP in 1989, who partnered with Intel in 1994 for development. By 1997 it was determined that it was going to be more difficult than expected. The first Itanium was released in 2001, 12 years after it was initially conceived and 7 years after initial development with disappointing performance and sales. The general success rates for the commercial sector tend to be on products that are evolutionary advances or updates on existing product lines. Achieving the desired 5x reduction in development time is likely to require similar approaches in defense systems.

3.5.4 LOCKHEED MARTIN SSI RESEARCH AGENDA

Lockheed Martin Systems and Software Research recently developed a research agenda for their Systems and Software Initiative (SSI) [Watson, 2009]. The SSI is motivated by the increasing complexity of current software-intensive systems that already strain the human limits of understanding. It is argued that today's systems have too many requirements, too many components, too much complexity, and involve too many people for humans to be able to comprehend the systems, and as a result, new tools and technology are needed. More specifically, the focus of the SSI research agenda is on MBSE and collaborative development environments. Their vision is that: "models will form the baseline and be at the center of all development activities" [Watson, 2009] as shown in Figure 8.

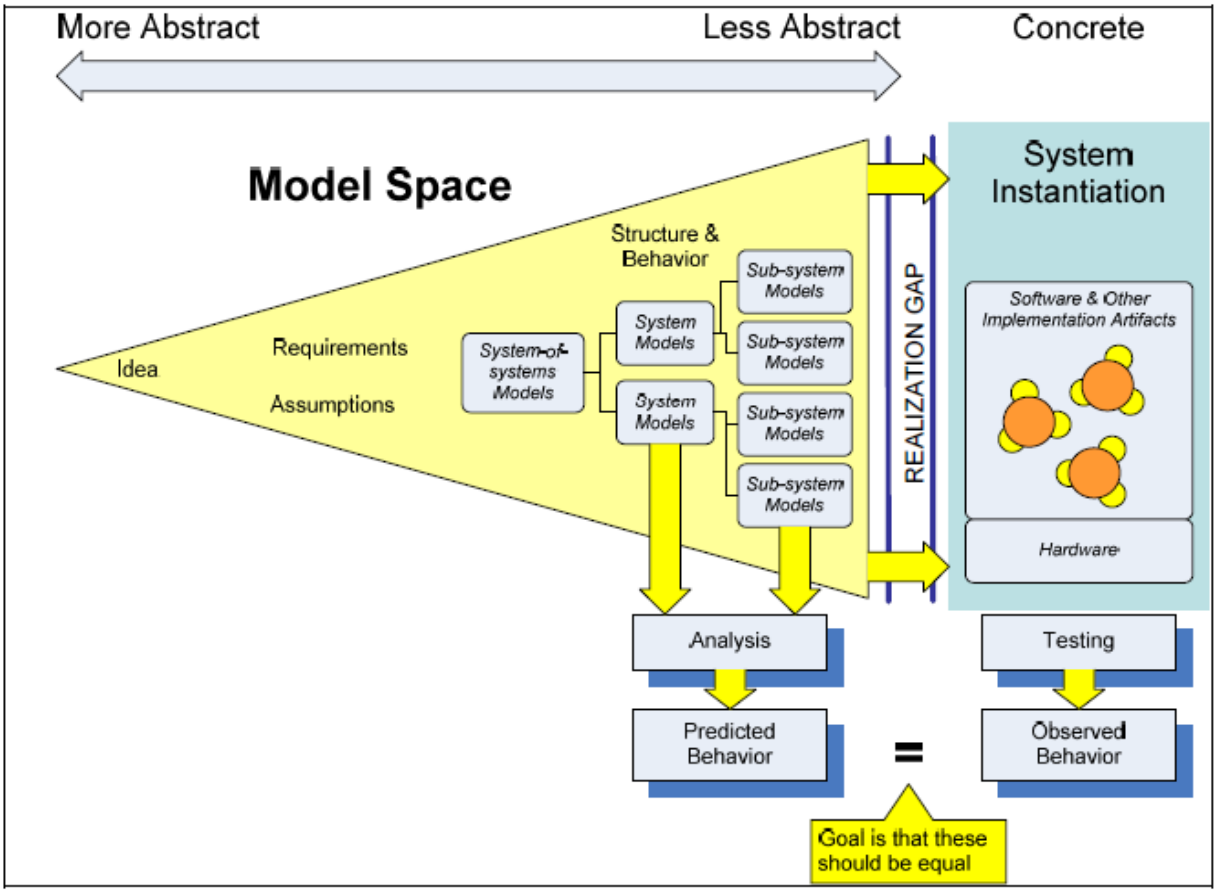


Figure 8: Visualization of System Models (Source: [Watson, 2009])

Research surrounding this vision is partitioned into the following categories:

1. Managing the model space continuum – supporting various domains and levels of modeling abstractions
2. Extending the model space – broadening the use of models
3. Crossing the realization gap - bridging the gap between model and implementation
4. Model exploitation – addressing ways to use and analyze models

3.5.5 DDR&E Technology Tools for Rapid Fielding Study

In 2009, a 60-day DDR&E Technology Tools for Rapid Fielding Study was conducted with the goal to identify "technological opportunities to significantly decrease the development time and increase the operational effectiveness of rapidly fielded capabilities" [Lemnios, 2009].

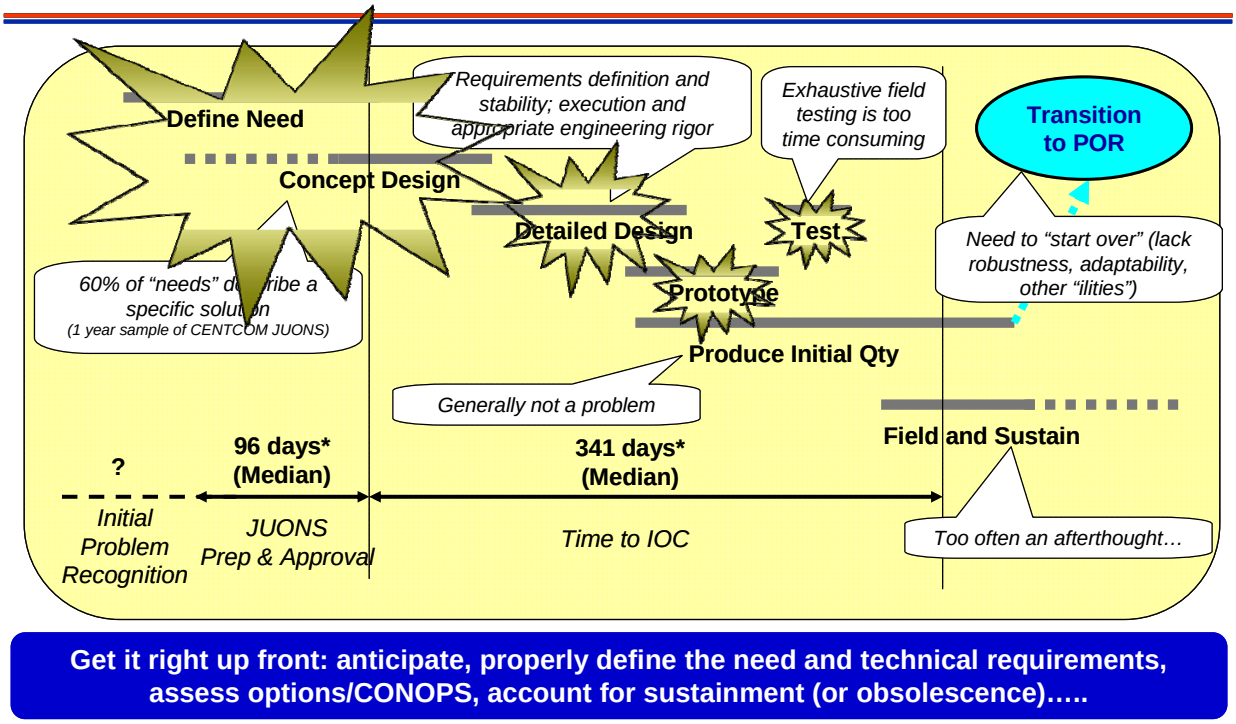


Figure 9: Leverage Points for Rapid Fielding (Source: [Carlini, 2009])

The primary findings as shown in Figure 9 were [Carlini, 2009]:

- Significant opportunities exist to develop and deploy technologies to strengthen the Department's ability to conduct rapid capability fielding
    - However, non-technical challenges (e.g. cultural, budgetary, contracting, etc) must be simultaneously addressed
- Greatest leverage in the "front end" of the life cycle
    - Concept Engineering: Rapidly elucidating the need, exploring solutions, developing CONOPs, and deriving requirements for materiel solutions
        - Virtual environments and rapid physical prototyping are linchpin technologies
- Opportunities exist to increase design, test, and production efficiencies
    - Examples include physics-based M&S to reduce testing and model-based engineering and manufacturing approaches

In addition to this work, the Systems and Software Consortium conducted a workshop in October 2009 with thirteen industry participants from eight different companies to make detailed recommendations on appropriate technologies for Rapid Fielding of defense oriented systems. The following recommendations were made [SSCI, 2009]:

- Encourage the use of modeling and simulation tools to expedite the development of life cycle activities
- Promote the creation and maintenance of common repositories (and common infrastructures) to achieve consistency, promote information sharing, and encourage stakeholder interactions
- Remove stovepipes and other barriers to information sharing and allow the open source community to participate
- Incentivize tool integration
- Support demonstration labs to encourage engineering advancements
- Enable greater (and richer) interaction between developers and user community to promote greater understanding of capability needs and system tradeoffs
- Acknowledge the benefits of automated (or auto-assisted) development environments for expediting the development process
- Encourage the implementation of iterative/incremental life cycles that provide on-going feedback and continuous system validation and verification
- Prioritize system attributes such as flexibility and robust designs
- Leverage the advances in virtual reality engineering to bridge the gap between reality and expectation

4 PROPOSED AREAS OF INNOVATION

Transforming SE requires understanding the gaps between traditional MPT capabilities and the needs of future systems. However, simply incrementally evolving existing SE concepts is unlikely to dramatically increase its relevance and put it on the technology curve. True transformation requires creative thinking and significant innovation. This section characterizes a new paradigm for SE. It presents innovation objectives and philosophy, articulates an innovation concept, and recommends critical areas of research to enable the necessary paradigm shift.

4.1 NEW PARADIGM FOR SYSTEMS ENGINEERING

Meeting the challenges presented by the critical system trends, addressing the gaps identified in the previous section, and transforming SE into a successful, relevant and timely discipline, requires a new SE paradigm. Within this new paradigm, SE must be:

- **Agile:** Allowing for quality, timely development with an incomplete and changing set of system requirements.
- **Integrated:** Part of the main development process and not an additional set of discretionary tasks.
- **Efficient:** Providing the greatest amount of benefits with the minimal number of steps and least amount of effort.
- **Leveraged:** Enabling exponential capability growth through the leveraging of computational, visualization, communication and information technologies, and prior systems experience.
- **Extensible:** Providing the ability to expand and enhance capabilities for future growth without having to make major changes in the infrastructure.
- **Deployable:** Enabling widespread impact through workforce education and broad application.

And yet, it must be sufficiently rigorous to ensure our systems are thoroughly engineered and will work as intended, satisfying the stakeholder's needs and vision.

Some particular areas of systems science need to be developed to address the emerging systems challenges. These areas include the architecture, design, and sustainment of:

- dependable systems - which includes security, availability, reliability and resilience
- evolving and self-adaptive systems – which are flexible and can efficiently and effectively be externally adapted or self-adapt to address changing environment and mission needs

- enterprise, systems of systems – which include the governance support and means of influence to manage systems that cannot be directly controlled.

The initial goal of this project is the development of a long-term roadmap to support the transformation of SE Practice, and will focus on how SE is carried out through advancements in methods, processes and tools (MPTs), rather than extending systems science and related knowledge or the accelerated development of the Systems Engineering workforce.

4.2 Innovation Objectives and Philosophy

As noted an earlier in this report, in general there are no single "best practices" independent of the value and particulars of a project and the environment in which it is being developed and deployed. However, rather than develop a set of Methods, Processes and Tools that are tailored to a specific profile, the roadmap proposals from this research will be made to be generally applicable to Systems Engineering projects of all types, but with a particular focus on addressing the aforementioned six critical systems trends for relatively critical, complex systems which are not sufficiently addressed by current System Engineering practices.

The intention of this research is to do for SE what Carver and Meade were able to do for integrated circuit design. The methodologies they developed have stood the test of time and have supported IC design for over 30 years and appear to be sufficient for the next 10 years of Moore's Law scale growth. The intended transformation of SE should be extensible over a similar period of time.

The approach taken is to focus research efforts on innovations that provide non-linear advances over the state of the art as incremental improvements are not believed to be adequate to address the emerging challenges. Thus, this research focuses on leverage in two major areas:

1. the inherent capabilities of computational, visualization, communication and information technology
2. the unique capabilities of the human mind

As noted earlier, for the past 30-40 years, and for the foreseeable future, the technologies in the first area have been improving at an exponential rate, whereas the capabilities of the human mind have been relatively unchanging. Advances in the second, human side of the equation have been largely achieved through specialization—allowing the application of more human minds to a problem, and the development of tools which extend human capabilities.

There are a number of differences between the capabilities of humans and technology, but the major differences are that computational technology is unsurpassed in its ability

to multitask and operate with blinding speed with great precision and accuracy within well-bounded contexts. Humans are poor at multitasking and performing quick, accurate calculations, but do extremely well in loosely bounded problems making decisions based on imprecise information. Staying on the technology curve requires the leverage and integration of the underlying technologies driving systems complexity into the SE process. Given the complexity of the systems of interest, a necessary approach is to instrument them, collect the information and use technology to enable humans to understand their emergent behavior. Computational and information technology should carry as much of the Value, Conceive, Develop and Use load as possible. Effective use of technology is an essential means of tightening the feedback and reducing the delay through this loop of activities. Human interaction should be limited as much as possible to areas in which only human knowledge and capabilities provide unique value. In this space, technology should be used as a tool to assist in the human creative and decision making process.

Rather than start with a predefined process and then focus our efforts on how tools might support these, we will look at how to optimize the combined capabilities of man and machine and develop supporting processes based on sound SE principles. These processes may then be integrated into larger existing processes or methods. It may well be that this results in disruption, but that is often the nature of technical breakthroughs. It is quite extraordinary that the current SE practices have evolved over the past half century without any significant disruptions.

In summary, we will create an integrated, yet modular set of innovations which will enable the optimal use of technical and human capabilities to improve general SE practices, while focusing on the emerging critical systems challenges. Reuse, leverage, and sustained tool development are critical elements for putting SE on the curve.

4.3 INNOVATION CONCEPT

As discussed in Section 3, the goal of SE is to ensure that systems are developed which can sustainably create value. In traditional SE practices, the value proposition is developed and communicated through static written documents, if at all, usually to a separate set of people who then create a concept of operations as shown in Figure 1. This is generally also done through a static set of written documents. This information is then passed on to another group of people who are responsible for developing the systems through the creation of an architecture, design and implementation. Again, the communication is often document based and typically accomplished through requirements and requirements traceability. Finally, the system is deployed and used. Human interaction with the system provides a huge amount of additional complexity and uncertainty in the operation of the system and its ability to create value. Determination of whether or not the appropriate level of value is created closes the loop, which impacts subsequent activities. While specialization allows many more minds to

work on the system, thus amplifying human capabilities, it also results in great challenges in communication and the development of a shared mental model of what is attempting to be accomplished.

4.3.1 OVERVIEW

The proposed concept, as shown in Figure 10, is one that we call Interactive Model-Centric System Engineering. In many ways, this approach is inspired by the automation and extensive use of computation, visualization, information and communication technologies used in the electronics industry as outlined in Section 3.2.3. The critical attributes are that each activity of the lifecycle, and communication between these activities, are accomplished with an optimal mix of technology and human capability, through the interactive use of a consistent set of data and models, with visualization that is optimized for the particular user (see Section 4.3.2). The use of graphical models has the potential to provide consistent meaning to all the stakeholders and bridge these gaps.

The three key elements are:

- **Interactive** - Interactive, iterative design, execution and re-design
- **Modeling** - The representation of information that can be processed by computation for analysis and on-the-fly simulation
- **Multi-Modal/Sensorial Visualization** - The visual representation of information, personalized to the needs of the user

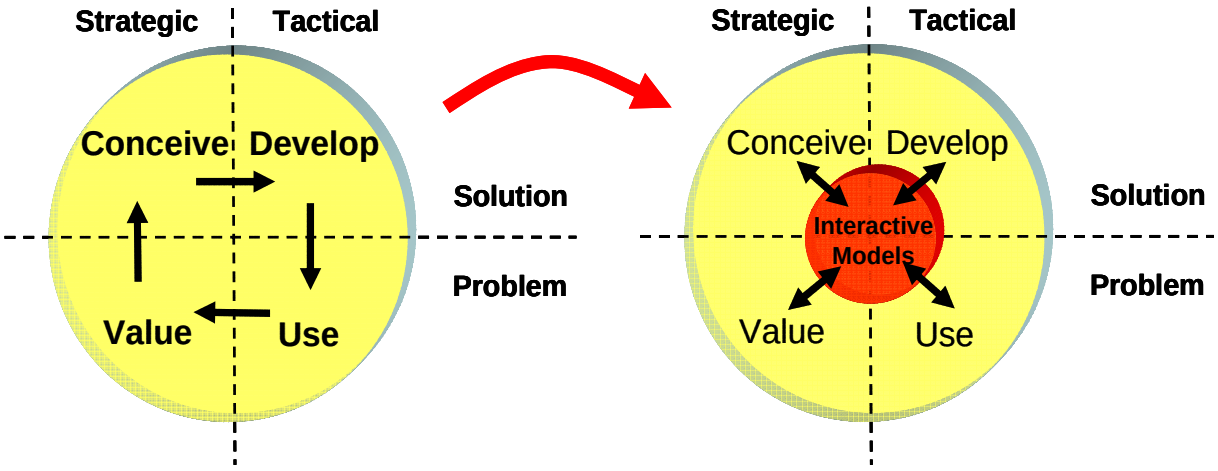


Figure 10: Transformation from Document to Model Driven

The adoption of an interactive model-centric system lifecycle model provides for the following necessary transformations:

- Document based → Interactive Model based
- Linear → Opportunistic
- Sequential → Consistent
- Human Avoided → Human Integrated

Tools and technology, which include models and simulation, can be used to:

- Facilitate understanding & decision making
- Improve development efficiency
- Automate processes

At the front end of the lifecycle, much of the effort is in the area of very abstract, multi-dimensional analysis and decision making across multiple domains. This is the area in which the human mind can be productively applied if it can be given the appropriate means of viewing the critical attributes. This is likely to be too broad of a space for design automation. However, it is an area in which visualization, and other multi-media, multi-modal means of presenting information can greatly improve the ability of human trade space analysis and decision making. As the system representation is transformed and refined through the lifecycle into less abstract and more concrete information, increasing design efficiency becomes a major focus. Finally, as these representations start to fit into established patterns and technology, automation can be applied.

In the four activity life cycle process, much of the human contributions to add value occur at the front-end of the process, in the strategic areas of value recognition and concept creation. In these areas, tools can be used as a means to capture ideas and concepts, and translate these into representations which can be recognized by a variety of stakeholders and contributors. As such, tools can be used to facilitate the iterative creation of shared mental models, and provide a means by which to evaluate their behaviors and attributes to facilitate decision making. Tools can be used to assist in the creation and analysis of architecture to help transform complexity into visualizations of emergent properties. Tools can provide developers with the ability to ensure that their designs can be successfully integrated to form a system that is in compliance with and supports the envisioned value proposition. These tools can be used to automate the development of system verification and validation. Finally, these tools can assist in the validating the desired usage models and support the training of personnel necessary for deployment.

It is critical that the models used in this lifecycle are capable of supporting the required level of detail and fidelity for each of their users, while being integrated for use through the entire life cycle. Limitations in their representational detail, views (for different users) and ability to be integrated and updated for consistency will greatly restrict their

value. Thus, these are required characteristics. While all of the capabilities need not appear at once, the overall value of the tools is increased as additional tools are added to the suite.

4.3.2 ELEMENTS

The critical elements for innovation and research necessary to transform Systems Engineering to meet the emerging systems challenges are shown in Figure 11. These elements reduce the time required to go from idea to deployment, and increase the operational effectiveness by providing closer coupling between the system solution and its usage thereby increasing the impact of the deployed system.

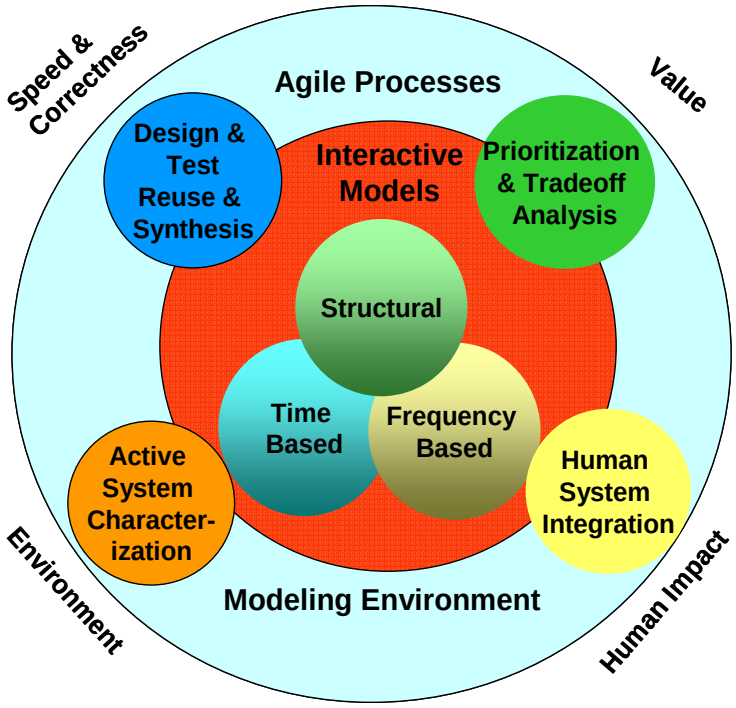


Figure 11: Elements of Innovation

At the core of this concept is the existence of interactive models. These models can be used to represent a system over the entire lifecycle, with a particular emphasis on providing the means to support a single, consistent system view to all involved stakeholders, including developers, service & support, and users. These models are intended to be both hierarchical and integrated such that various portions of the system may be modeled at different levels of precision. These will consist of state-based, capability-based and structural models. The three different types are described below.

4.3.2.1 State-Based Models

The first approach is to use state based simulations to determine that the system has the correct functional behavior. This requires a set of models that can be executed sequentially to show causal, temporal relationships. This is the most common use of executable models and is the object of study for most mathematical treatments of Model-Based Systems Engineering [Wymore, 2004]. These models can be used to not only ensure correct functional behavior, but they can also be used to perform stress testing and fault recovery through actual or synthetic loads, and interface analysis and testing.

4.3.2.2 Capability-Based Models

Analysis can also be done in the "frequency" domain (capability-based), rather than the time domain (state-based) used for functional analysis. In this domain, the frequency at which certain types of activities occur is characterized, and then analyzed to determine how the system responds to an ensemble of these inputs in the aggregate. In this way, time and frequency domain analysis has an analog to that used in understanding communication and signal processing systems. This type of analysis can be used to understand the performance and reliability characteristics of the system.

4.3.2.3 Structural Models

Another type of models are structure based which can be used to analyze the system with respect to its interconnectivity and aggregate behavior, independent of sequential functional behavior. For example, structural analysis can be used to better understand the dependencies of the system without the need for stimulus. An analog to this in integrated circuit design is the use of static timing analysis which looks at all potential signaling paths within the circuit independently of actual circuit stimulus, or in software the analysis of execution paths. Such analysis can be used to determine the relative flexibility of the system for change, and where and how change can most efficiently be made. A number of other system attributes can be analyzed using this approach.

For the model-based methodologies to work effectively, it is critical that the models used for each of these views are kept consistent throughout the system's lifecycle. In addition, users of these models need to be informed of changes that have been made which impact their view of the system.

4.4 Critical Areas of Research

There are a wide range of potential research topics to support the seven critical elements described earlier. To reduce this to a manageable set, the following criteria were used to determine which were most appropriate for research within the SERC:

- Critical to the transformation of SE by addressing identified gaps
- Furthers the sponsor's mission

- Requires multidisciplinary research which is not currently being done
- Appropriate scope and scale for an academic research program
- Supports a 3-year or longer roadmap of research
- Anticipated to have measureable impact

In addition, the critical elements necessary for the transformation of SE noted above are integrated into a set of clustered areas of innovation which are described below. It is expected that the research results from each of these modular clusters also can be integrated together to provide a wider set of capabilities.

The resulting eight SET research focus areas and their relationships to the 4-stage lifecycle are shown in

Figure **12.**

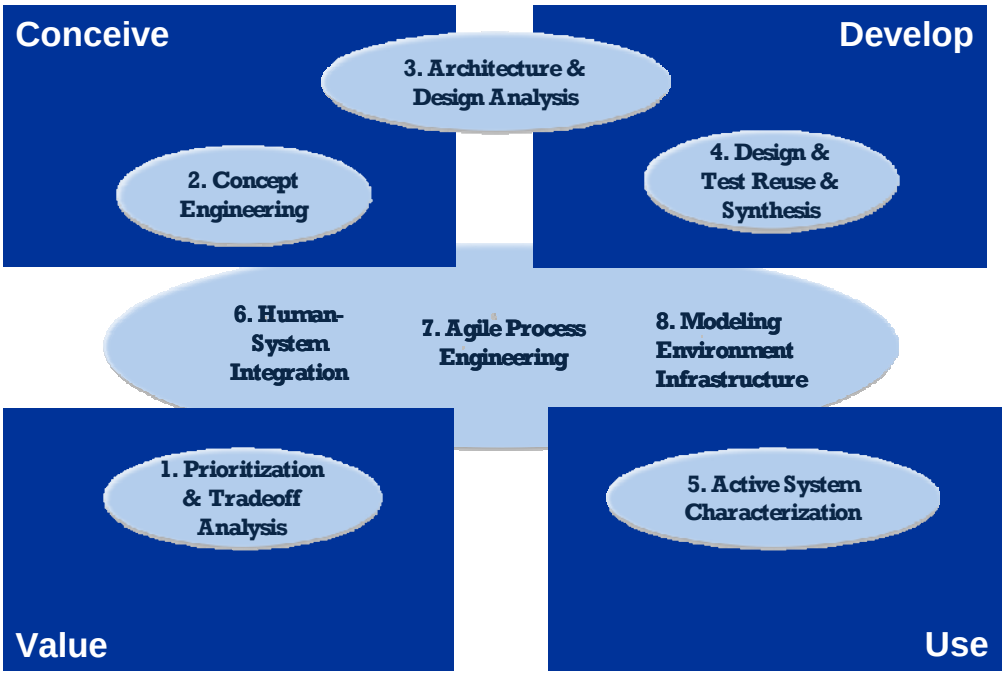


Figure 12: Relationship of SET Research Areas to 4-Stage Lifecyle

The sum total of these research areas provides for an integrated, yet modular architecture. While each of the individual areas can be pursued independently and provide incremental value, each provides additional benefit to the other areas in which the collective research provides more value than the sum of the individual parts. The relationship of these research areas to various levels of modeling, from high-level virtual abstractions to physical entities is shown in Figure 13. Each of these research areas is described in more detail below.

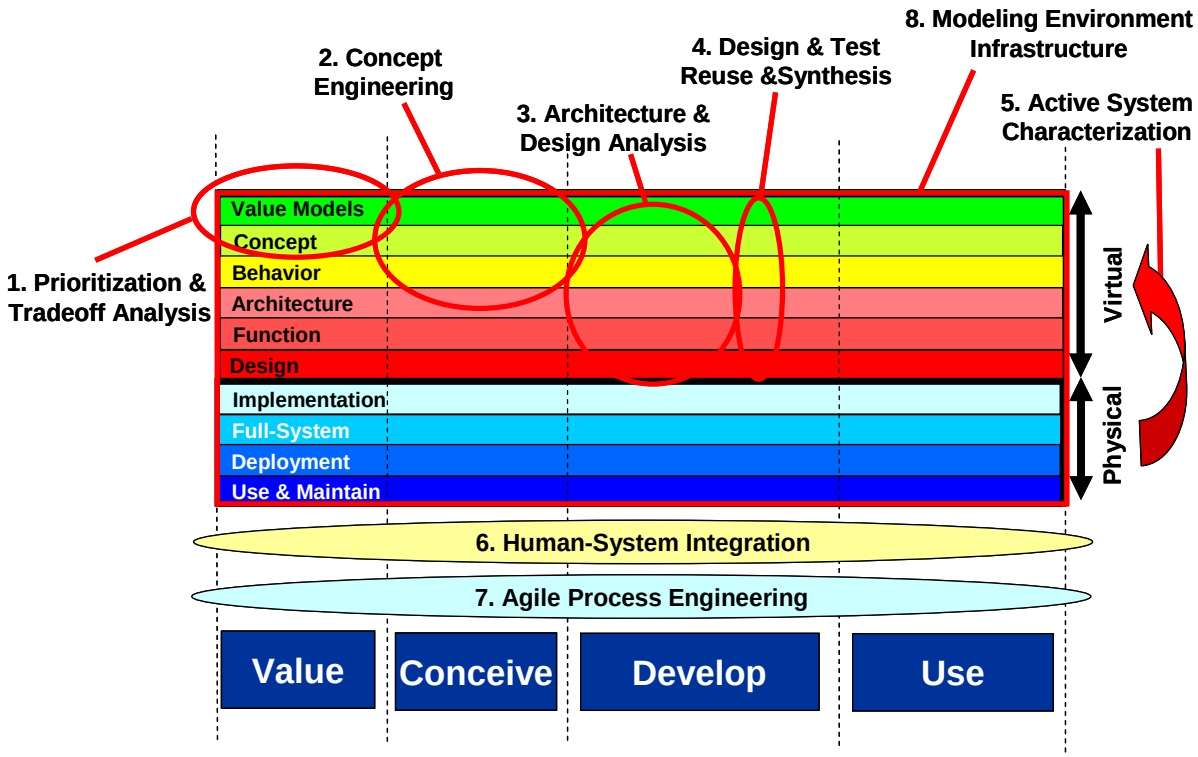


Figure 13: Relationship of SET Research Areas to Modeling Levels

4.4.1 PRIORITIZATION AND TRADEOFF ANALYSIS

While systems engineers quite often conduct prioritization-based trades, this usually is interpreted in a budgetary sense (e.g., how much thrust can we afford, how much weight can we afford) , but not in the sense of how much security can we afford, or more nuanced, how much accessibility or usability will we sacrifice for the degree of security we need. So value means more than simply budgetary limitations, it also should be expected to include the overall value created for the systems stakeholders. This research will explore the available techniques that are available to analyze prioritization trade-offs and determine how they can be used to construct tools to both improve the quality of decision making and reduce the time that is necessary to accomplish it.

4.4.2 CONCEPT ENGINEERING[6]

Currently, the conceptualization phase of a project is done either through a laborious document driven process or in an ad hoc manner with inconsistent results. In either case, system developers and users alike often have inconsistent understandings of what the system is actually supposed to do, and how it creates value. This research will explore the tools and processes that provide an efficient interactive environment where multiple stakeholders can create a shared mental model during brainstorming processes—from concept of operations development throughout the lifecycle.

4.4.3 ARCHITECTURE AND DESIGN ANALYSIS

Architectural descriptions and designs of large systems are generally far too complex for architects and systems engineers to understand their attributes and emergent behaviors, in order to make appropriate design tradeoff decisions. This research focuses on the development of visualization and analysis tools to facilitate the decision making process throughout the architectural specification and design process. Such tools may provide visualization for fragility, change propagators (areas not to touch), extensibility, modifiability and security. Such guidance could include how and where a system should be modified with the least amount of impact to design and test.

4.4.4 DESIGN AND TEST REUSE AND SYNTHESIS

Much time and effort may be spent in the design, development and test of functionality that has already been implemented with similar capabilities. However, it is often difficult for a developer to know what already exists for this leverage, and understand the implications of using existing technology. In addition, the developer may not have the time or inclination to make their work accessible for use by others. This research will explore how technology can be used to mine existing architectures, designs and tests looking for patterns which can provide a means to categorize and catalog such work for reuse. Low-effort means of making designs and tests more accessible, and the requirements for synthesis of higher level architectures into lower level implementations using these repositories will be explored.

4.4.5 ACTIVE SYSTEM CHARACTERIZATION

It is very difficult to understand how systems actually behave based on design documents and models. In addition, systems are generally composed of legacy element and many undocumented “features”. However, for new capabilities and features to be added to an existing system, it is often critical to understand its current behavior. This

[6] The term “Concept Engineering” was coined by [Carlini, 2009].

research will look at how existing systems can be instrumented, measured, and analyzed to provide the necessary model fidelity. This applies to both system models that can aid in subsequent design efforts, and also conceptual models used to validate how the system is being used and value is being created.

4.4.6 HUMAN-SYSTEM INTEGRATION

The need to accurately model the system is limited not just to technology, but includes the human element as well. The human factor needs to be considered not only with respect to the usual human-factors issues such as ergonomics and safety, but also considering humans to be an element of the system and improving the entire system to ensure that the overall system–both technology and human components–is being optimized. Research is necessary in two areas. The first is the development of appropriate models for humans who interact with, and thus are elements of a system. The second area is an exploration of the capabilities and limitations of people who are developing the system.

4.4.7 AGILE PROCESS ENGINEERING

In general, there are no best practices, but rather there are practices that have been shown to be beneficial in a given environment. Agile approaches have been successful in a number of applications, but the practices may be quite brittle when the application attributes change. To realize the benefits of interactive, model-centric engineering, the supporting processes must be easily adaptable, taking full advantage of the evolving capabilities of tools and technology, within the environmental constraints. This research will focus on the development of a process (or suite of process assets) and the related governance that supports the rapid development of systems in environments (such as classified defense/IC work) that may not be the natural home ground for agile development.

4.4.8 MODELING ENVIRONMENT INFRASTRUCTURE

An integrated modeling environment is required to provide effective and coherent communication between the users of the collaborative modeling and design environment. Without this capability, the users would have to fall back into sequential, isolated development which mitigates many of the advantages of modeling. The research will focus on the development of such an environment that provides effective model interoperability and management. In addition, work will be done to understand how the environment can best present information to each user and increase design efficiency.

5 3-YEAR ROADMAP

5.1 OVERVIEW

The following overview contains the description of a high-level framework to provide a context for each of the constituent research areas that comprise the integrated, modular roadmap. This is followed by two operational scenarios that describe how the transformed methods, processes and tools may be used throughout the lifecycle of a greenfield and a brownfield system. Next, there are descriptions of each of the eight research focus areas including the problem and opportunity, proposed research advances and benefits, the impact on critical gaps, and the resulting 3-year roadmap. Finally, there is a short summary of the results. Even though this is a research program and the details described below are almost certain to change, the eight descriptions provide a useful snapshot of the research vision that can be used to synchronize the subsequent efforts in each of the SET research areas.

5.1.1 RESEARCH FRAMEWORK

The framework for the focused research areas is shown in Figure 14.

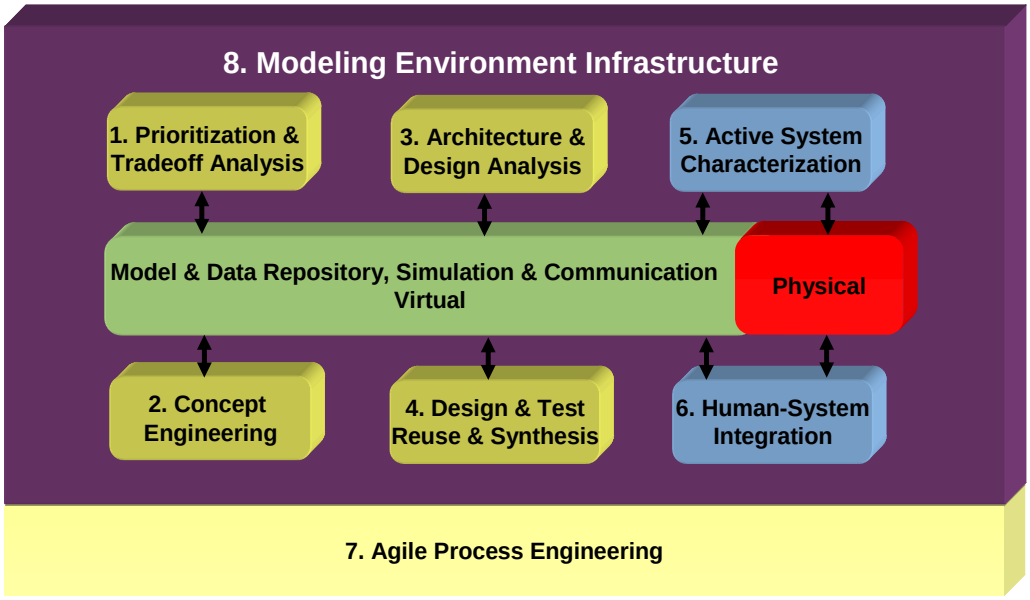


Figure 14: SET Research Area Framework

Central to this framework is the Model and Data Repository, Simulation and Communication (MDRSC) substrate which provides the ability for the various tool sets and capabilities to work closely in synchronization. Although the current SET research project has focused on network centric, software intensive systems, the resulting research framework is extensible to general systems and is not restricted to a particular domain. Each of the focus areas noted in Figure 14 may address systems composed of arbitrary combinations of hardware, software, human agents, and governance systems. The systems being developed may be a complex system of systems, a standalone platform or a rapid response action. The following is a brief description of how these elements work together as an integrated, modular framework.

The **Prioritization and Tradeoff Analysis** module provides the capability to input the particular factors relating to the relative value and priority of high-level capabilities of the system under development. The output of this module is the creation of models which are invoked during subsequent Concept Engineering, Architecture and Design Analysis, and Design and Test Reuse and Synthesis work. The creation of a central set of models enables each subsequent development effort to work under the same set of valuation and prioritization constraints. In addition, this module provides visualization capabilities to enhance the user's ability to understand and assess the validity of these value and prioritization models.

The **Concept Engineering** module provides an interactive, collaborative, multimedia environment for multiple stakeholders to quickly construct concepts of operation and other high-level abstract models of the system under development. Within the environment is a library of concept modules, a variety of scenarios, and Reuse and Synthesis capabilities. Models are simulated against scenarios in the repository while the Prioritization and Tradeoff models are employed to determine the relative value of each concept. Models are also supplied by the Human-System Integration module for the scenarios being investigated and also for the actual system models. The Human-System Integration modules may also be useful in the interactive concept development process by enhancing the ability of the stakeholders to develop a shared mental model by analyzing the behavior of their interactions and noting shortfalls to the appropriate users. Design and Test Reuse and Synthesis may also guide the user concepts towards more time, effort, cost and risk optimal solutions due to availability of reusable or synthesizable design and test. In addition, these capabilities can be used to construct a number of conceptual views of the systems with increasing levels of fidelity.

The **Architecture and Design Analysis** module provides the human operator with the ability to develop and optimize an architecture and design which supports the conceptual view while providing an optimal solution based on the Prioritization and Tradeoff Analysis models described earlier. The Design and Test Reuse and Synthesis capabilities provide the Architectural and Design Analysis module with the capability to quickly perform what-if analysis across a large set of trade spaces. Some of this work can be done through computationally automated means, but much can be done by

providing the human operator with a visualization which enables the development of insights into the appropriate actions. This is an area in which human capabilities can be greatly extended.

Design and Test Reuse and Synthesis provides the means, by leveraging existing assets and utilizing computational capabilities, to rapidly translate high level abstractions into lower level ones. These capabilities can be used across the entire range of design and test abstractions from concept to implementation. While it is improbable that synthesis and reuse can be effectively used in a turnkey manner for the entire system through all the various levels of abstraction, these capabilities certainly can decrease design and test development time, and reduce system development cost and risk if applied in areas of greatest leverage. Reuse and synthesis will be guided, both automatically and with human guidance, based on the information and analysis capabilities from the various SET modules. For example, the models created from the Prioritization and Tradeoff Analysis module provide optimization guidance for Reuse and Synthesis decisions. Architectural and Design Analysis capabilities can also be used to provide direction for Reuse and Synthesis decisions. Human-System Integration models provide the fidelity necessary to ensure that the entire system can be optimized, not just the individual technology and human subsystems and agents. The models and test scenarios developed in the Concept Engineering module can be used as an entry point into this module.

Active System Characterization has the role of providing feedback between the virtual and physical system domains. While many of the simulations that support the above work can take place using virtual or hybrid simulations with physical components, it is very difficult to understand and model the actual deployed environment with a high degree of fidelity. Understanding is increasingly difficult when this environment is a complex system of systems where there may be no centralized control or where human agents play a significant role. This module constantly monitors the actively deployed system and feeds back this information into the model and data repository ensuring that that this information is up to date in near real time. This information not only provides updates to the models of technical systems, but also of human behavior and system usage both for the system that is being developed and the system that is doing the developing. For example, performance attributes of the deployed system can be used to update the performance expectations of the system being developed with a new set of capabilities. The behavior of the developers can be used to determine potential changes in the development processes, or can be used to determine the effects of making changes to the system in both time, effort (exactly who is being impacted), cost and risk. Much of this work can be done through computational data mining. The modules in this area provide the necessary feedback path to effectively synchronize the virtual and physical worlds.

Human-System Integration, true to its name, is integrated throughout the system lifecycle activities. In particular, this module provides inputs to the rest of the modules

to ensure that the human factor is properly taken into consideration and modeled with the end goal of optimizing the entire system, not just individual technical and human subsystems and components. Specifically, this is not limited to ergonomics and task analyses. This module takes inputs from the Active System Characterization module to ensure that its models are kept up to date and accurately reflect the behavior of the human agents in the existing system which may change over time. Not only is this module needed for the optimal development and deployment of the system being created, but also for the system which is creating the system. Thus, Human-system Integration provides critical information to the Agile Process Engineering module to ensure that these processes are properly optimized.

Agile Process Engineering is necessary to provide the processes and governance to enable productive parallel development in each of the aforementioned areas. Without such processes, these parallel developments would either be chaotic and counterproductive, or unnecessarily constrained through forced sequential activities. In addition, it is critical that the processes themselves are agile and tuned for the particular mission, system being developed and environments in which they are both being developed and being used. This module will use the models from the Prioritization and Tradeoff Analysis module to determine the relative importance of various system attributes to tune the processes used to develop them. The models from the Human-System Integration module will also provide input on how best to use this organization. In addition, processes will be reused and/or synthesized for improvement. It should be understood that the development system should be architected, designed and optimized with many of the same techniques used for the systems that it is developing. These Agile Processes will act as the control system which governs how each of the SET modules interact. This module and the Modeling Environment Infrastructure together comprise the environment, processes and governance that supports the SE activities.

Finally, there is the **Modeling Environment Infrastructure**. This infrastructure not only supports the central MDRSC capabilities, but it also supports all of the aforementioned modules and infrastructure while providing the most effective interface to the users of the system who may involve a broad range stakeholders. This environment provides the means by which the Agile Processes are realized in operation. This environment plays a critical role in moving SE from an oversight role into an integrated, embedded capability in the engineering and support of systems throughout their lifecycle.

The eight research areas described above address the SE gaps identified in Section 3.4.4 as shown in Table 5. High impact is noted in green for gaps which are a primary beneficiary of a research area. Medium impact is noted in yellow for gaps which are secondary beneficiaries of the research area.

Table 5: Impact of Research Focus Areas on Gaps

	Research Focus Areas							
Identified Gaps	Prioritization & Tradeoff Analysis	Concept Engineering	Architecture & Design Analysis	Design & Test Reuse & Synthesis	Active System Char-acterization	Human-System Integration	Agile Process Engineering	Modeling Environment Infrastructure
System Requirements								
Low-Overhead Communication								
Architectural Design Support								
Risk/Opportunity Management								
Verification & Validation								
Legacy Integration								
Human Aware/ Self-Adaptive								
Complexity Handling Capabilities								
Cycle Time Reduction								

The gaps noted above are the result of interviews, literature reviews and analysis as described in Section 3.4.4. The specifics of how each research area impacts these critical gaps are described in Section 5.2.

5.1.2 Operational Scenarios

The integration of the SET framework and capabilities, shown as the Collaborative Foundation, into the Enterprise is shown in Figure 15. It is clear from this diagram, that the SET infrastructure will be used throughout the systems lifecycle by a vast array of users and stakeholders across the Enterprise. The following sections present two scenarios illustrating how the collaborative foundation accomplishes the transformation of SE.

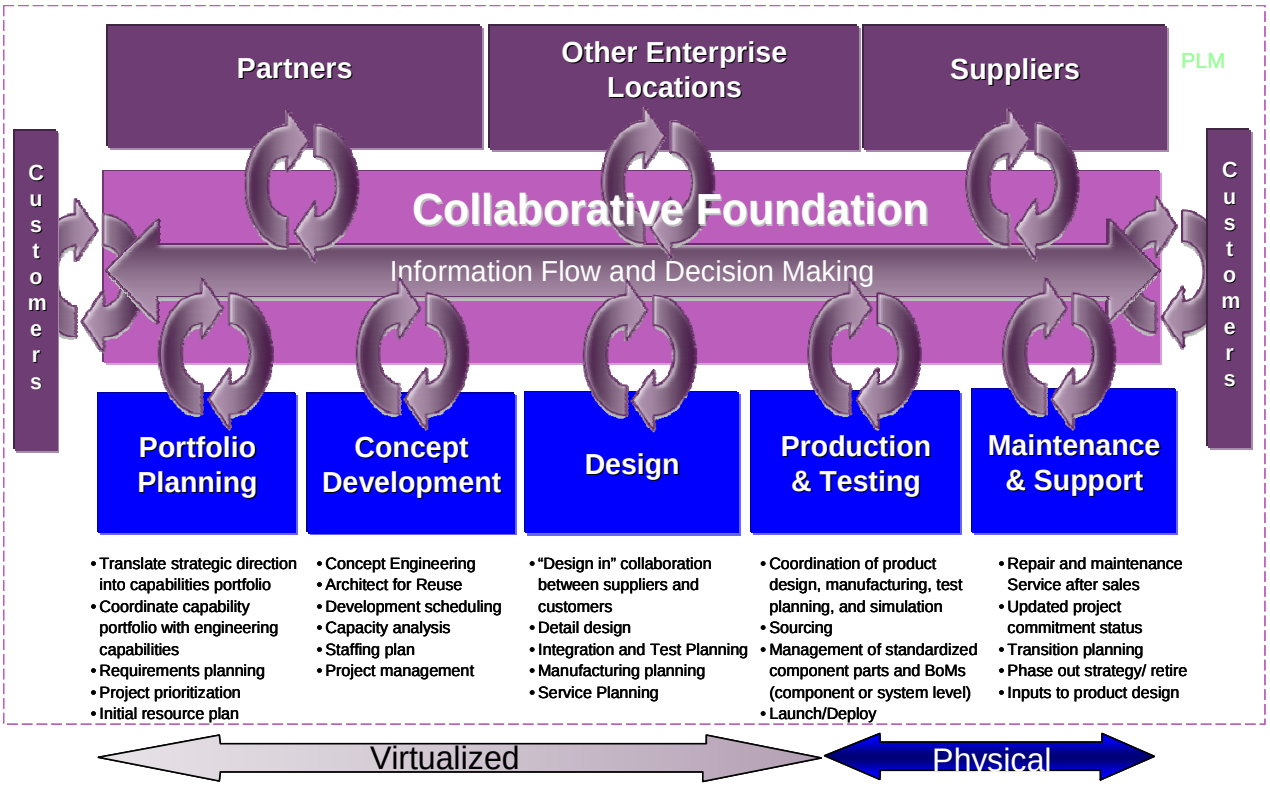


Figure 15: SET Framework and Capabilities Integrate into the Enterprise

5.1.2.1 Operational Scenario 1: New System Development

The first operational scenario is that of the greenfield development of a brand new system. In this case, there may well be a lengthy mission assessment and portfolio planning activity to determine the necessary return on investment for this system development and deployment. These activities will likely involve substantial efforts in constructing Prioritization and Tradeoff Analysis models to determine how investments in various areas provide value. This work will provide the basic infrastructure which will be updated over time to determine how to prioritize efforts in this area once the system is developed, deployed and being optimized over time. From the beginning, it is important to note that the framework enables integrated and continuous V&V capabilities *throughout the lifecycle*.

Once the mission assessment and portfolio planning activities have started, a series of Concept Engineering activities are initiated exploring the various possible system opportunities. Rough cost, effort and development time estimates can be made based on high-level Design and Test Reuse and Synthesis evaluations. The ability to quickly assess the approximate attributes of these systems, as well as the cost and effort, allows for rapid and rational decision making. Both the tradeoff analysis and concepts are tuned in parallel during this process.

Once a 'go' decision has been made, the architectural design analysis and design can begin in earnest. The process that will be used is tailored for this particular project using a library of information of projects completed in the past with tuning in the appropriate areas. The work that had been done during the evaluation process can be leveraged and moved forward to the next higher degree of refinement and fidelity. In all cases, the development of models and information from past projects has a very beneficial impact on the time, cost, effort and risk in this new program. Each refinement of the architecture and design results in increased depth of understanding of the value proposition and assists in the prioritization process moving forward. Throughout this development, the process is continued to be tuned based on the performance of the organization using Active System Characterization. Since this is a new development, it presents a rare opportunity to architect the sensors into the system so that it can be effectively monitored during its operation to provide critical data for Active System Characterization.

The conceptual, behavioral and architectural models, the design, and the implementation are constantly being validated and verified through both regression tests and dynamically configured tests. When defects are encountered, they are quickly analyzed to the appropriate abstraction layer to identify and fix root causes, and a new barrage of tests are created to extensively test the affected area. This methodology is very much the same that is currently being deployed today, but has the added benefit of being integrated through multiple levels of modeling abstraction. Throughout the development process, support personnel and end users have access to the simulator (eventually being migrated to the actual system) and are able to validate how the system will behave in their environment. Not only does this train the support personnel and user community, but it also provides feedback on the utility of the system at the earliest possible time. When the system is finally deployed, it behaves much as it did in the simulation and the time is greatly reduced to achieve full impact in the field.

While not specifically mentioned, it is important to understand that Human-System Integration has played an integral role in all of the modeling and decision making regarding the system being developed and the system doing the developing. These capabilities are integrated in such a way that they are both ubiquitous and unremarkable—the human capabilities are regarded as being equally as important as the technical capabilities within the system.

5.1.2.2 Operational Scenario 2: Incremental System Development

This case addresses the use of the system to add incremental capabilities to an existing system. This brownfield case has many similarities to the aforementioned greenfield case, so this description will focus on the differences. In the brownfield case, much of the important information comes from the existing system. For example, Active System Characterization provides information on the ability of the existing infrastructure to

support the desired new capabilities under load. It may well be that these new capabilities are not possible without a major system upgrade, which impacts the cost and schedule of making the necessary changes. The Active System Characterization might also provide information on how the system is actually being used which may be counter to the original intent and impact the roadmap of future changes. For example, in the Hubble Telescope project, it was noted during operation that the telescope was unable to focus properly. This feedback enabled the development team to design a correction which allowed the system to perform remarkably well over a long period of time despite the optical defect. The architectural and design analysis capabilities provide the means by which to determine the impact of making changes and also determine how these changes might best be made.

While the initial architectural definition of the greenfield system was focused on how to make it extensible over time, efficiency and minimization of change is the major challenge for brownfield systems. For greenfield systems understanding possibilities and the creation of value are critical, while for brownfield systems providing maximum benefit with constrained resources is the primary challenge. Both types of development use all of the eight focused areas of research, but they often use many of them in different purposes.

5.2 PROJECT DESCRIPTIONS

Each of the eight research focus areas are described below with respect to the problem and opportunity, proposed research advances and benefits, impact on gaps, and a high-level roadmap of the research effort and timelines that constitute this work.

5.2.1 PRIORITIZATION AND TRADEOFF ANALYSIS

5.2.1.1 Problem and Opportunity

Projects with limited resources must find ways to prioritize their desired capabilities in order to fit within their resource constraints. In some cases, the limited resource may be funding or qualified personnel. For quick-response or rapid-fielding situations, the limited resource is calendar time.

In order to prioritize capabilities, it would be convenient if all the system's success-critical stakeholders had readily expressible and compatible value propositions. "Readily expressible" is often unachievable because the specifics of stakeholders' value propositions tend to be emergent through experience rather than obtainable through surveys. In such cases, synthetic-experience techniques such as prototypes, scenarios, and stories can accelerate elicitation of priorities.

Readily compatible stakeholder value propositions can be achievable in situations of long-term stakeholder mutual understanding and trust. However, in new situations, just considering the most frequent value propositions or success models of the most critical project stakeholders shows that these are often in conflict and must be reconciled.

For example, Figure 16 shows a "spider web" of the conflicts or "model clashes" among these stakeholders' success models that can cause projects to fail. The red or gray lines in Figure 16 show the conflicts found in analyzing a representative failed project, the Bank of America (BofA) Master Net project. This and analyses of other failed projects have shown that these value-proposition conflicts are built into the roles of the four most common project stakeholders: users, acquirers, developers, and maintainers [Boehm, 2000; Al-Said, 2003].

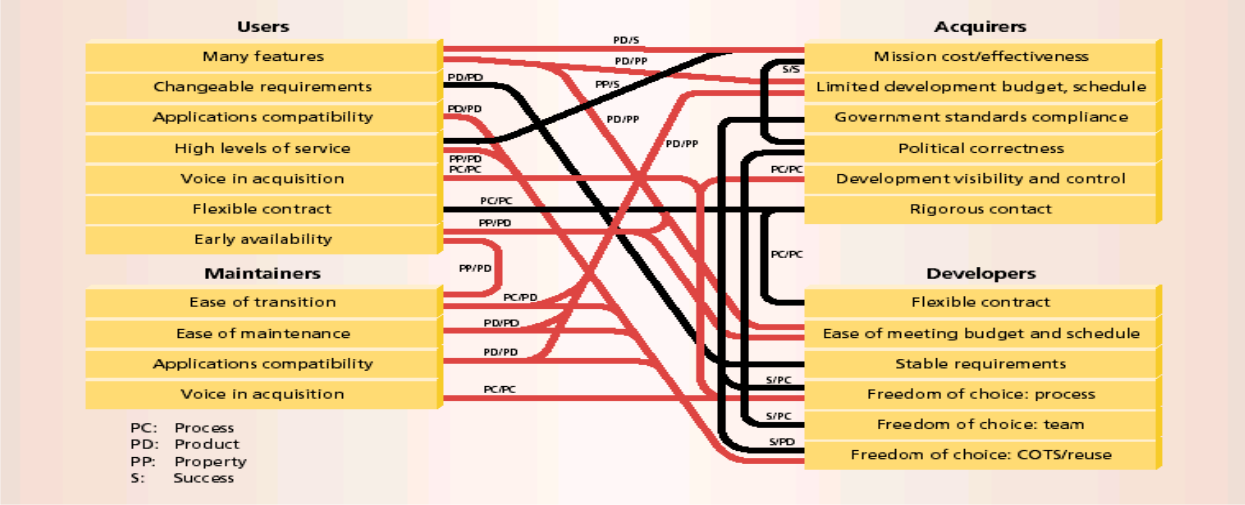


Figure 16: Value Proposition Model-Clash Spiderweb diagram (Source: [Boehm, 2000])

For example, the MasterNet users specified 3.5 million source lines of code (SLOC) worth of wish-list features that were put into the project's requirements specification. Even at an extremely optimistic development cost of $30/SLOC for a project of this size, this would cost $105 million. Thus, the users' product model was in serious conflict with the acquirers' property model of a $22 million budget. Also, many of the wish-list items had no mission effectiveness rationale, conflicting with the acquirers' success model of mission cost-effectiveness.

Faced with this dilemma, the acquirers searched for a supplier who could reuse a related solution to provide useful features at a low cost. They found one in Premier Systems, who had built similar applications for small banks. However, their product model only worked on Prime computers, which conflicted with the BofA users' and maintainers' product-model value propositions of applications compatibility and ease of maintenance, given that BofA was an IBM mainframe operation. Also, the Prime host

could not scale up to BofA's throughput, response time, and reliability needs, causing a property-product evidence shortfall that should have been addressed earlier.

Given the goodly number of model clashes in Figure 16 (and there are potentially many more), the task of prioritizing them may appear formidable. However, there are several effective approaches for stakeholder value proposition reconciliation, such as:

- Expectations management. Often, just becoming aware of the number of potential stakeholder value proposition conflicts that need to be resolved will cause stakeholders to relax their less-critical levels of desire. Other techniques such as lessons-learned retrospectives, well-calibrated cost models, and "simplifier and complicator" lists help stakeholders better understand which of their desired capabilities are infeasible with respect to budget, schedule, and technology constraints.
- Visualization and tradeoff-analysis techniques. Frequently, prototypes, scenarios, and estimation models enable stakeholders to obtain a better mutual understanding of which aspects of an application are most important and achievable.
- Prioritization. Having stakeholders rank-order or categorize the relative priorities of their desired capabilities will help determine which combination of capabilities will best satisfy stakeholders' most critical needs within available resource constraints. Various techniques such as pairwise comparison and scale-of-10 ratings of relative importance and difficulty are helpful aids to prioritization.
- Groupware. Some of those prioritization aids are available in groupware tools, along with collaboration-oriented support for brainstorming, discussion, and win-win negotiation of conflict situations.
- Business case analysis. Determining which capabilities provide the best return-on-investment can help stakeholders prioritize and reconcile their value propositions. Good examples of these techniques are in [Biffl, 2005].

5.2.1.2 Proposed Research Advances and Benefits

Some of these approaches for effective prioritization are covered under other SET research areas. Expectations management and visualization are largely covered under Concept Engineering. Aspects of these, as well as groupware, are covered under Human-Systems Integration. Here we focus on Tradeoff Analysis, which also involves business case analysis.

There are quite a number of quality attributes or key performance parameters that may need to be tradeoff-analyzed and prioritized. These include Security, Safety, Privacy, Reliability, Availability, Survivability, Accuracy, Correctness, Interoperability, Usability, Performance, Adaptability, Cost, Schedule, and Reusability. These come with numerous conflicts that need to be tradeoff-analyzed. For example, high levels of Security add significant levels of Schedule and Cost for certification. Minimizing key-distribution

leakage risk via a single-agent key distribution system will compromise Reliability by creating a single point of failure. Password-protecting a soldier's weapons will compromise Usability, by rendering them useless to fellow soldiers if the soldier is incapacitated. Adding protection layers and other defenses will often seriously degrade Performance.

Well-calibrated parametric models provide help in performing tradeoff analysis. For example, Figure 17 provides a graphical view of the tradeoffs between cost, schedule, and reliability afforded by the Required Reliability (RELY) and Schedule Compression (SCED) parameters in the COCOMO II cost estimation model, as calibrated to 161 representative projects. It clearly shows the familiar "cost, schedule, dependability: pick any two" effect. For example, suppose that the project wants a High RELY level (10K-hours MTBF) and a 20-month delivery schedule for a 100K SLOC set of capabilities within a budget of $5.5M. Unfortunately, a High RELY level and a 20-month schedule require a budget of $7.33M. For a cost of $5.5.M, the project can get a High RELY level and a delivery schedule of 23.5 months, or a 20-month delivery schedule but a Low RELY level. The three circles in

Figure 17 show the three resulting "pick any two" points.

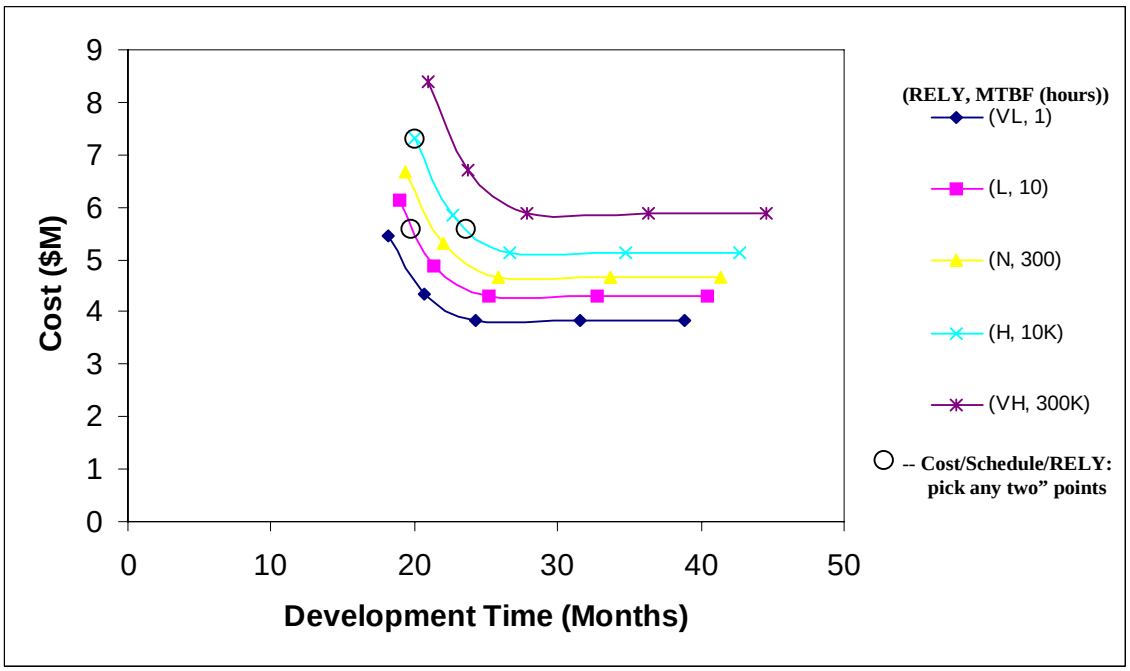


Figure 17: COCOMO II Cost/SCED/RELY Tradeoff Curves, 100K SLOC Project

However, if one is able to get the stakeholders to prioritize the set of capabilities, using the COCOMO II model indicates that the stakeholders can have all three attributes of

cost, schedule, and reliability if they accept an initial operational capability that requires just 77K SLOC of software.

5.2.1.3 Roadmap

For software-intensive systems, good state-of-the-practice and state-of-the-art summaries are available for value-based prioritization in [Biffl, 2005] and for attribute tradeoff analysis in [Clements, 2002]. Another starting point is the just-starting SERC project RT-18, Valuing Flexibility, which plans to include the tradeoffs between Flexibility or Adaptability and other desired attributes as part of its valuation analysis. For rapid fielding, a key strategy to investigate early will be Reusability, which has frequently reduced fielding time by factors of 4 or more, but must be preceded by at least one develop-for-reuse application with a longer fielding time. This could be pursued with the closely related Design and Test Reuse and Synthesis research area. Other rapid fielding strategies that involve aids to prioritization and tradeoff analysis will be rapid prototyping, agile methods, and collaborative innovation laboratories. These would be pursued in conjunction with the Concept Engineering, Architectural and Design Analysis, Human-Systems Integration, and Agile Process Engineering elements.

The expected impact of this research on the critical gaps identified in Section 3.4.4 is shown in Table 6.

Table 6: Prioritization and Tradeoff Analysis Impact on Critical Gaps

Identified Gaps	Impact
System Requirements	Early and rapid prioritization of stakeholder values and resolution of cross-stakeholder conflicts, such as those in the Figure 16 spiderweb diagram, focuses the project on its most cost-effective mission requirements, and avoids wasted time on low-priority requirements. Wiki-based stakeholder collaboration and prioritization tools have reduced requirements convergence times from months to days.
Low-Overhead Communication	Buildup of shared stakeholder values increases tacit knowledge among stakeholders, and reduces the communication bandwidth required to handle development-phase or support-phase change requests.
Architectural Design Support	Well-calibrated tradeoff analysis tools such as the COCOMO II-based cost/schedule/reliability/functionality tradeoff results shown in Figure 17 help stakeholders understand tradeoffs and manage their expectations. The asymptotic cost-schedule tradeoff curves in Figure 17 have helped hundreds of projects avoid unrealistic schedule commitments and resulting overruns.
Risk/Opportunity Management	Tradeoff tools and value-based prioritization aids also help stakeholders understand risks early and to avoid or resolve them before they lead to either late rework or fielded failures. Such late rework can lead to up to 91% cost and schedule penalties for 10- million-SLOC software systems
Verification & Validation	Same as for risk/opportunity management.

Legacy Integration	Tradeoff tools apply to some extent as risk/opportunity management tools, but with more difficulty.
Human Aware/ Self-Adaptive	Tradeoff tools apply to some extent as risk/opportunity management tools, but with more difficulty.
Complexity Handling Capabilities	Tradeoff tools apply to some extent as risk/opportunity management tools, but with more difficulty.
Cycle Time Reduction	Same as for risk/opportunity management.

The roadmap for Prioritization and Tradeoff Analysis research will be performed over a 3-year period with specific deliverables associated with each year as shown in Figure 18.

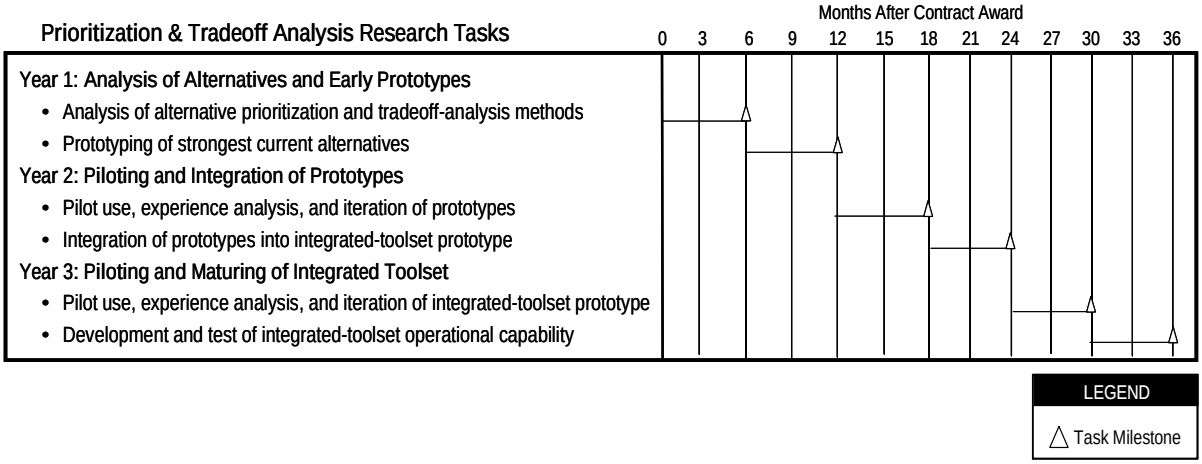


Figure 18: Prioritization and Tradeoff Analysis Research Roadmap

5.2.2 CONCEPT ENGINEERING

5.2.2.1 Problem and Opportunity

The weakest link in Systems Engineering is often the link between what the war-fighters or analysts need, and what the development team "thinks" they need, together with a shared understanding of the operational environment and associated constraints and dependencies. While conceptualization seems more "ad hoc" than engineering, a disciplined approach is necessary to ensure that the stated needs of the customer are transformed into a product or service that meets the actual customer and mission needs. Currently, the conceptualization phase of a project is either an ad hoc event with pens and napkins resulting in presentation slides, or it is a sterile, and laborious document driven process in which a large, unwieldy text-based document is created – destined to become shelf-ware [Cloutier, 2009].

In either case, system developers and users alike often have inconsistent and uneven understandings of what mission and capabilities the system or service must provide. Furthermore, there may not be an agreed upon value of the product or service. Concept Engineering [Carlini, 2009] involves the creation of an infrastructure and processes necessary to provide an efficient interactive environment where multiple stakeholders can brainstorm to develop a concept of operations model that can be used throughout the lifecycle. The approach should include a broad range of tools to enable conceptualization at multiple levels of conceptual refinement ranging from brainstorming to creating and evaluating high-level system behavioral models. Moreover, it should be done in a manner that facilitates input from a set of diverse stakeholders and limits cognitive demands placed on them. Done correctly, the models generated in the Concept Engineering task may also be of value in operator training and validation by deployment personnel.

5.2.2.2 Proposed Research Advances and Benefits

This research provides the opportunity to quickly and graphically articulate a concept of operations, conceptual and behavioral models for new missions, business processes, and feature sets to realize a shared mental model and understanding of the mission, and potential solutions across a set of diverse stakeholders. This graphical environment has the potential to become a mechanism for agile stakeholder expectation elicitation. It is likely that this environment will have to be tailored for selected application domains for operational, mission, and semantic consistency.

The high-level goals of this research are to provide:

- a more effective cognitive concept development environment;
- an environment for rapid evaluation of alternative concepts;
- a vehicle to validate the concept throughout the development lifecycle; and
- a vehicle to perform trade analysis for upgrade options in subsequent versions of products.

This research will apply tools and processes necessary to provide an efficient interactive environment so that multiple stakeholders can create a shared mental model during the brainstorming process through the development of a concept of operations. This work provides a natural transition to the Architecture and Design Analysis research.

The full set of tools would include graphical interface tools to allow the interactive, collaborative development of models representing a concept of operations and system behavior, that can be simulated and tested with a portfolio of canned scenarios, with the resultant behavior being viewable from a variety of perspectives that are tailored to each user's needs as shown in Figure 19.

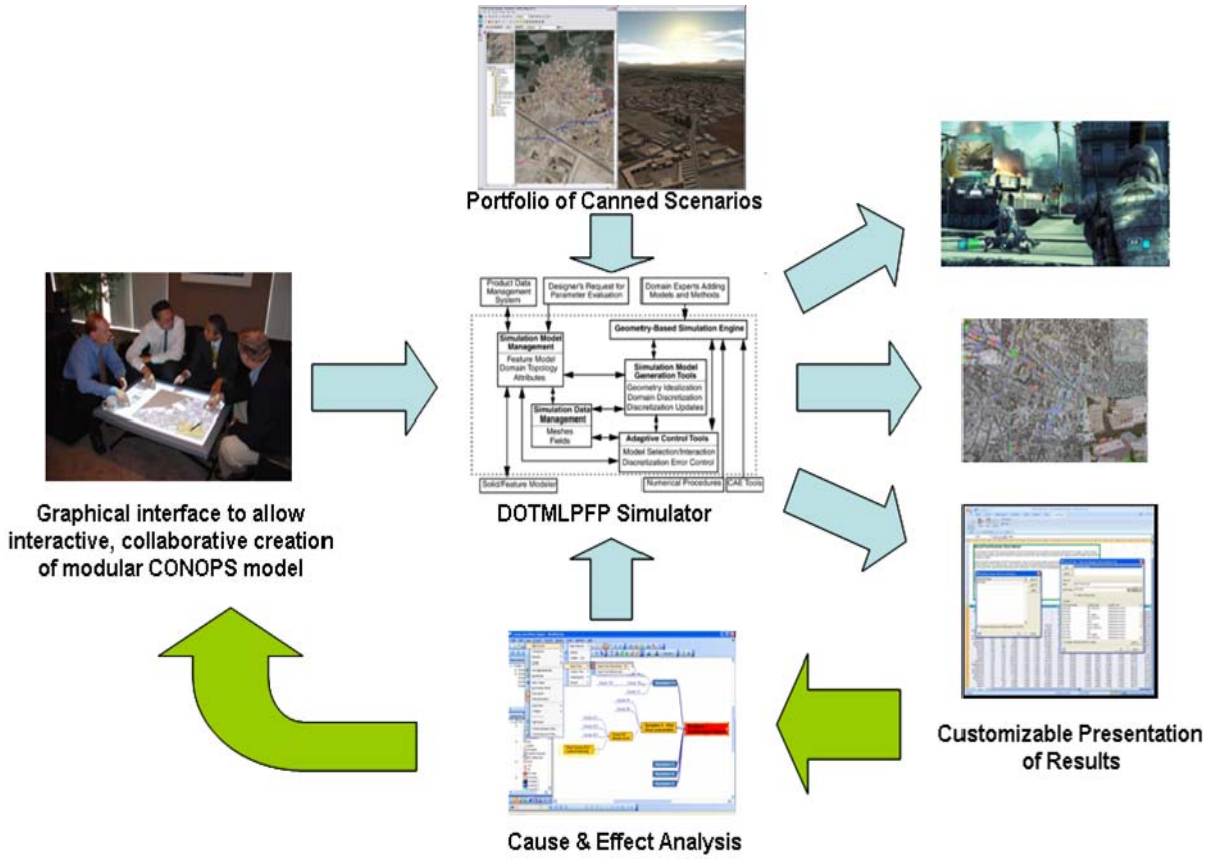


Figure 19: Concept Engineering System (Source: [Cloutier, 2009])

Three phases will be used in the interaction development process: 1) Conceptual, 2) Specification and 3) Design and Implementation, as shown in Figure 20.

The benefits to this research are to dramatically improve the ability to collaboratively and interactively create a model of the desired system behavior which can be used throughout the system life cycle resulting in:

- accelerated time to market through:
  - ability to rapidly evaluate alternative concepts
  - increased capabilities in effective, rapid distributed decision making
  - improved communication with all stakeholders of the value proposition and intended operation of the system
- increased quality through improved ability to:
  - define the right system upfront
  - improved system verification and validation
  - ability to train deployment, service and support personal earlier in lifecyle

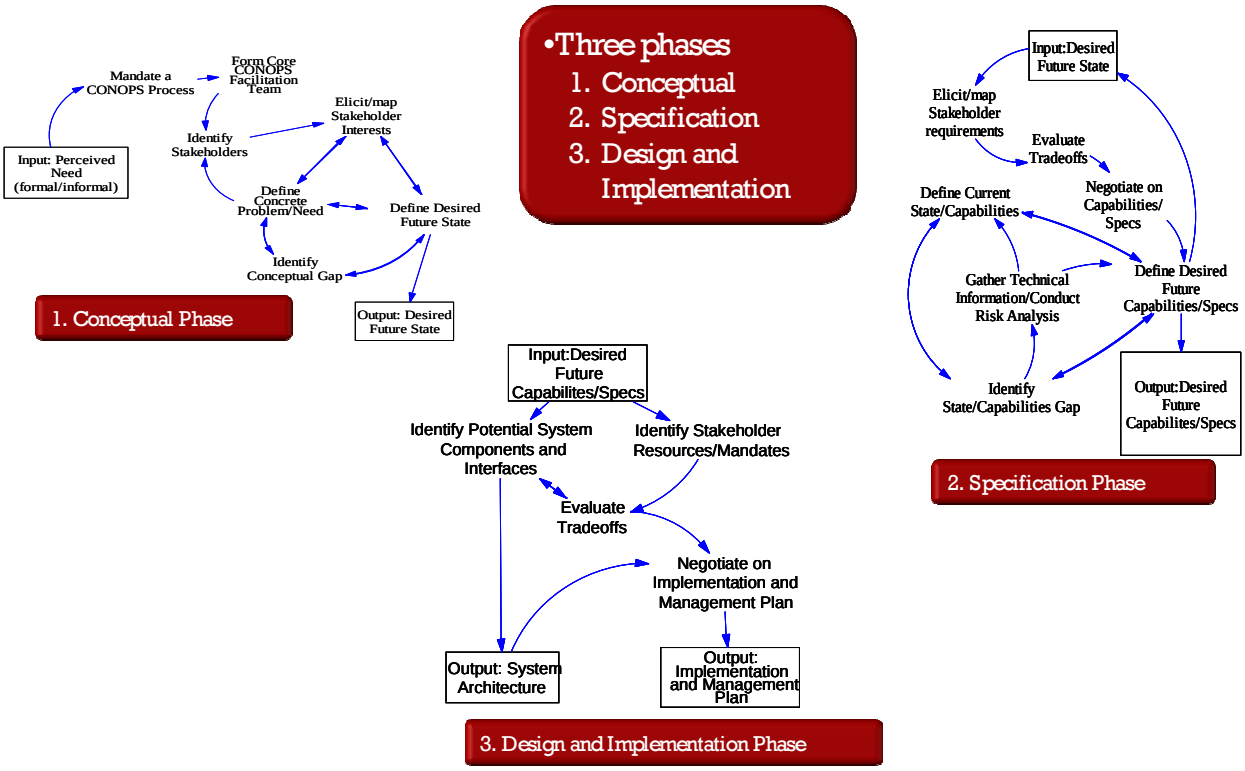


Figure 20: Concept Engineering Phases (Source: [Cloutier, 2009])

5.2.2.3 Roadmap

The expected impact of this research on the critical gaps identified in Section 3.4.4 is shown in Table 7.

Table 7: Concept Engineering (CE) Impact on Critical Gaps

Identified Gaps	Impact
System Requirements	CE provides the capability for all the stakeholders to iteratively and interactively conceive, develop, and validate the operational concept, conceptual behavior and characteristics of the proposed system. Providing this shared and validated model of the system may be the most effective means by which to generate derived system requirements. Rather than creating a complex set of requirements which together define what the system should do, CE provides the means to define and validate the operator expectation for the system and its performance, and then generate the resultant requirements.

Low-Overhead Communication	CE should provide low-overhead communication between all of the stakeholders who interact in the creation of the conceptual model of the system. The interactive capabilities provide instant feedback to the users and between the users. This feedback conveys information in the most effective way by being tailored to the needs of each of the users.
Architectural Design Support	Effective architectural design is dependent on an intimate understanding and vision for the conceptual operation, behavior and desired characteristics of the system. CE provides an executable model which can be interactively used by the architects and stakeholders in providing this shared vision and understanding of the system.
Risk/Opportunity Management	Much of the leverage in a system lifecycle happens during the conceptual phase. Mistakes in this phase are often irrecoverable or amended only at great cost later in the system lifecycle. CE provides the means to validate the value proposition of the system up front thus reducing the risk of discovering problems downstream, while also providing a rapid prototyping conceptual environment for opportunity discovery at the onset.
Verification & Validation	Understanding the conceptual operation of a system up front is critical for both verification and validation. The architectural and/or behavioral model should be incrementally developed and interfaced with CE to validate the solution. CE then provides the capability for continuous validation of the system concept upfront, while providing the fundamental understanding that is essential for effective verification throughout the lifecycle.
Legacy Integration	CE provides the capability to allow users of legacy systems to validate new systems concepts. This provides a bridge for end users between the system that they currently have and the systems that they will have in the future. CE makes use of a library of composable elements, and thus can take advantage of legacy models for future developments, ensuring their consistency. High level models of the current system can be integrated into CE and provide validation at the desired level of granularity. If they do not exist, CE will allow for the creation of "boundary components" to enable simulated interfaces to the legacy systems.
Human Aware/ Self-Adaptive	CE provides an interface that is tailored to the specific user of the system, using the lexicon of the user instead of "engineering speak". Thus, the user's interactions provide a means by which to gauge if the system will conceptually provide humans with the desired behavior. The development of the CE models is a human based activity which is meant to capture the appropriate elements of human behavior.
Complexity Handling Capabilities	CE is, by its nature, a conceptual modeling system, which enables the abstraction of complex systems behavior to the appropriate level under consideration. As the CE models are refined, increasing levels of fidelity may be added while interfacing to existing systems throughout the lifecycle. Thus, these models can be used to provide information at the desired level of detail, while providing the conceptual view of how the system is actually supposed to work. This connection is essential to handling complex systems.

Cycle Time Reduction	By providing an efficient environment for the construction of conceptual models, CE can both reduce the amount of time spent on these activities and more importantly greatly increase its effectiveness. This upfront work will dramatically reduce the efforts down stream by focusing those efforts on the necessary, value creating work. In addition, the ability to do continuous validation reduces the risk of rework which can have a huge impact on schedule and quality of the delivered system.

The roadmap for Concept Engineering research will be performed over a 3-year period with specific deliverables associated with each year as shown in Figure 21.

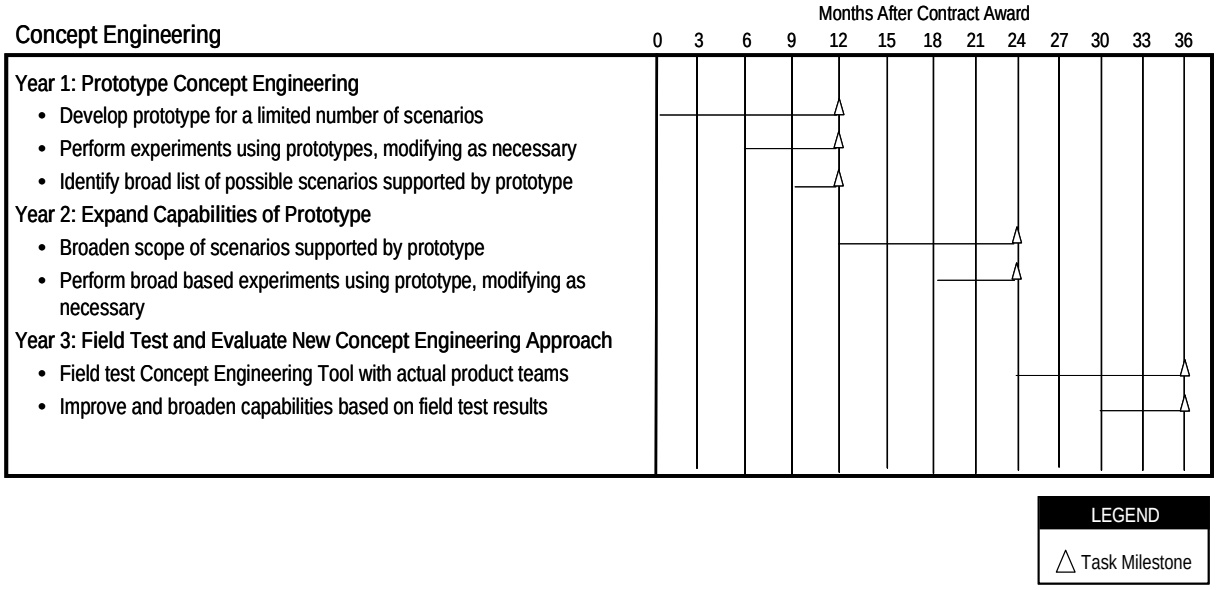


Figure 21: Concept Engineering Roadmap

5.2.3 ARCHITECTURE AND DESIGN ANALYSIS

5.2.3.1 Problem and Opportunity

A system is considered successful if it satisfies stakeholder needs. On the surface, this would imply that designing the system to meet the requirements specification developed at project inception would be sufficient. In reality, however, stakeholder needs change and evolve continuously throughout the development process, and in fact, throughout a system's lifetime [Madni, 2007]. As such, the value that the system provides to its users will diminish during its lifetime unless action is taken to ensure that the system continues to evolve to meet those changing needs [Browning, 2008; Sangwan, 2008a].

For a system to evolve gracefully, its architecture needs to be designed with this recognition in mind. This capability has typically been achieved by adapting the

functional architecture – the architecture developed to meet the functional requirements of the system – through design for "ilities" [Madni, 2007; Blanchard, 2006]. In recent years, significant strides have been made in understanding the implications of flexibility and adaptability, and how they impact architecture and architectural decisions [Fricke, 2005; Engel, 2008]. There is a growing recognition today that ilities, are of critical importance in system design. In fact, one could argue that today they are even more important to the lifetime value of a system than the functional requirements because changing functional requirements can be addressed only if the architecture can endure such change [Ross, 2008; Sangwan, 2008b; Madni, 2007].

In light of changing priorities, one is left with the challenge of "how to create systems with the desired behaviors and to predict and suppress the undesired ones" [Crawley, 2004; Madni, 2008]. In that case, might an alternative approach to architectural design, one that focuses primarily on the non-functional requirements of the system, be more advantageous? Instead of first decomposing a system according to its gross functionality, the architectural design is driven by the important quality attributes that ensure its longtime survival [Sangwan, 2007; Madni, 2007].

The problem does not end, however, once the architecture has been completed. In the evolution of the developing, and subsequently deployed system, the impact of the changes necessary to meet the evolving need are invisible from the development team. The complexity of the system, the degree of interconnectedness, and the hidden, undocumented, dependencies obfuscate the consequences of changes [Madni, 2008, Madni, 2010a]. In essence, the development team needs tools that provide real-time multiperspective visualizations of the system under consideration so that design options can be assessed in terms of their local and global impact at the time the contemplated changes are expected be made.

5.2.3.2 Proposed Research Advances and Benefits

In light of the foregoing, advances are needed along the following key dimensions:

- Develop architecture-centric design approach with system adaptability as a primary design driver
  - unify definitions of adaptability across systems and software
  - develop analytically-based architectural techniques (patterns, tactics, etc.) for building adaptable systems that aide architects and enhance communication through shared vocabulary.
- Develop architecture design tools that help architects visualize the ility trade space and perform tradeoffs
  - leverage and extend existing trade space visualization tools (e.g., Advanced Trade Space Visualization tool developed at PSU Applied Research Lab)

- leverage and extend existing evaluation framework tools (e.g., XTEAM evaluation framework developed by USC's Center for Systems and Software Engineering)
- explore architecture visualization concepts to highlight fragility
- develop methods to guide how and where a system should be modified to minimize impact on design and test.

The benefits of this research are: superior architecture and design synthesis tools based on a unified definition of adaptability across software and systems; and superior coverage of the architecture space leading to effective tradeoffs analysis.

5.2.3.3 Roadmap

The expected impact of this research on the critical gaps identified in Section 3.4.4 is shown in Table 8.

Table 8: Architecture and Design Analysis Impact on Critical Gaps

Identified Gaps	Impact
System Requirements	Non-functional requirements drive architecture-centric design approaches so that the designed system exhibits the desired systemic qualities. Furthermore, it incorporates explicit prioritization of requirements (derived from mission objectives) and conflict resolution.
Low-Overhead Communication	Visualizations of system properties computed from existing artifacts (models, source code, etc.) provide holistic views over systems without manual intervention.
Architectural Design Support	Architecture-centric design incorporates specific design rules and tactics to ensure the design reflects the desired qualities ("ilities"). Architecture visualizations and trade-off tools make explicit the impact of design decisions and provide greater situational awareness to the development team.
Risk/Opportunity Management	Architectural trade-off tools can explicitly examine the consequences of decisions on value, risk, etc. through multi-attribute exploration and design-by-shopping. Architecture visualizations identify and highlight regions of the system that are fragile, rigid, etc. therefore providing valuable insight into the risks associated with upcoming design decisions.
Verification & Validation	Metrics for non-functional requirements, either direct or indirect, provide a means of assessing the degree to which a system under construction meets, or falls short of, the desired capability prior to test and evaluation, and therefore provide opportunities to improve capability earlier in the lifecycle and therefore at lower cost.

Legacy Integration	The ability of a system to evolve is closely correlated with comprehension and simplicity. Architecture visualization tools help developers understand the degree and character of legacy system complexity, visualize existing dependency structures, and highlight "weak points" in the system where change cannot be tolerated.
Human Aware/ Self-Adaptive	The suite of methods, processes, tools described here provide the human developers greater situational awareness of the system under construction so that they can more rapidly construct a complete and consistent understanding of the system, its limitations, frailties, capabilities, and potential. Furthermore, architecture-centric design can be utilized so that adaptability and adaptivity are primary architectural drivers.
Complexity Handling Capabilities	Architectural visualizations help development stakeholders comprehend large-scale, highly-complex systems without compartmentalizing the system and thus losing sight of emergent properties and systemic constraints.
Cycle Time Reduction	In evolutionary development (iterative and incremental) the system design is in constant flux as new capabilities are added and existing capabilities extended. This is the equivalent of maintenance in traditional development, and is concordantly time-consuming. By rapidly and automatically visualizing the architecture, its dependencies, systemic properties, etc. the architects and developers have greater insight into the system and mechanism to examine the impact of their design choices and changes, reducing the time spent in rework and repair.

The roadmap for Architecture and Design Analysis research will be performed over a 3-year period with specific deliverables associated with each year as shown in Figure 22.

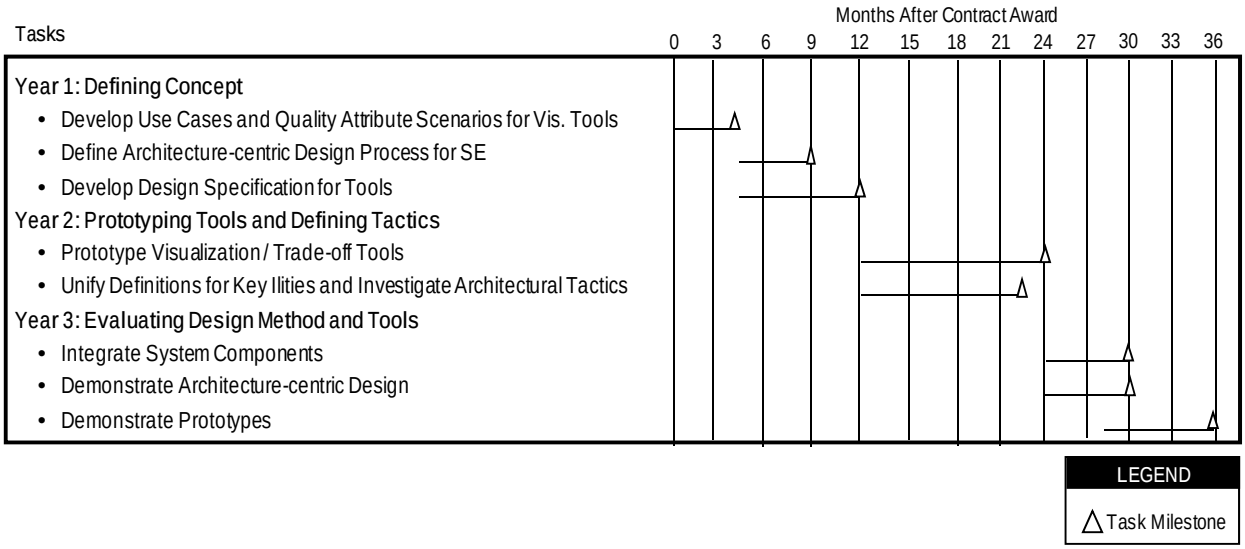


Figure 22: Architecture and Design Analysis Research Roadmap

5.2.4 DESIGN AND TEST REUSE AND SYNTHESIS

5.2.4.1 Problem and Opportunity

Much time and effort may be spent in the design, development and test of functionality, which has already been implemented with similar capabilities on other projects/products. While it is desirable to reuse that similar capability, it is often difficult for a developer to know what has already been developed that can be leveraged, and understand the implications of using existing technology. In addition, the developer may not have the time or inclination to make his/her work accessible for use by others. Finally, it may be possible to synthetically design, develop and test new features and functions using advancing computational capabilities to assemble existing lower level subsystems and components.

This is a major area of opportunity. This work involves the mining of existing architectures, designs and implementations to create a "standard cell" library of reusable components [Cloutier, 2007; Cloutier, 2010]. Current MBSE research and development has shown that one of the benefits of this approach is the creation of and ability to reuse this intellectual property. This area has been the focus of a great deal of research and development [Cloutier, 2007]. In the integrated circuit world, it is possible to directly synthesize lower level models from high-level representations all the way to the physical device geometry through a number of representations including behavioral, RTL (register-transfer-logic), standard cell and gate array cells, transistors, polygons and masks. While there are many papers and books written about reuse in software, it has been less successful, often due to the less restricted nature of the media. Where there has been success, it has generally occurred in limited domains. General Systems Engineering research must be done in a more focused way to determine where the current state of the art can be used in system development, perhaps resulting in the use of domain specific "cell libraries". In addition, non-research work can be done to create an open source community, which shares such libraries; or commercial ventures, which can supply them in the early phases of Systems Engineering.

Traditionally, at the back end of Systems Engineering is verification and validation (V&V). One of the current issues is that V&V is not usually integrated into the overall design and development process. Rather, the system is conceived, architected, designed and implemented and then it goes through a validation process. When problems are found, it is generally far too late to make changes without substantial cost in terms of effort, time and risk. In addition, systems engineers who are removed from the design typically do the verification task. It is not unusual that the results of verification do not directly correlate with the system's mission, but rather address derived requirements. At a high-level, the "V" in the "V" process is fundamentally broken with respect to the rapid fielding of new systems and capabilities.

Rather than happening at the end of the process, validation should take place at the very beginning of the process and should happen continually through the system life cycle. Validation is at the heart of any rapid prototyping process. In addition, design and verification need to go hand and hand throughout the life cycle process. Attention should be made to eliminate the introduction of defects as much as possible through technology and system reuse, and correct by design construction; and to detect and correct early in the life cycle process whenever possible using automated capabilities.

The efficiency and efficacy of Verification and Validation (V&V) should be dramatically improved using model based Systems Engineering. In particular, validation can take place at the front-end of the lifecycle. In fact, rapid prototyping is all about rapid validation, which ensures that the concept satisfies the requirements for value creation. In addition, test "vectors" can be captured throughout the development process at high-levels of abstraction. They can then be used in smaller blocks as their functionality is refined over time. It may be that automatic generation of "requirements" is achievable using this type of process.

5.2.4.2 Proposed Research Advances and Benefits

This research will explore the use of technology to mine existing architectures, designs and tests for reusable patterns, which can provide a means to categorize and catalog this work for reuse. In addition, low human effort means of making designs and tests more accessible will be explored. Finally, the requirement for synthesis of higher-level architectures into lower level implementations using these repositories will be explored.

Objectives for this research include:

- Determine how to use technology to mine existing architectures, designs and tests looking for patterns, which can provide a means to categorize and catalog this work for reuse.
- Create low human effort means of making designs and tests more accessible.
- Define how higher level architectures, designs and tests can be synthesized from lower level implementations using these repositories.

The potential payoffs for this research are [Cloutier, 2006]:

- **Reduce Cycle Time** – reusing and synthesizing architecture, design and test has the potential to dramatically reduce the time required to develop high quality systems
- **Control Complexity** – through the use of architectural patterns to help control the complexity of an architecture by standardizing it on a well known and practiced pattern
- **Mitigate Risks** - using and applying known and characterized architecture, design and test patterns introduces less risk than developing these from scratch

- **Knowledge Management** – enabling reuse of good concepts and implementations, and to preserve them for future projects
- **Common Understanding** - describing parts of the architecture, design and test in the context of known and understood patterns results in a common understanding of the system

5.2.4.3 Roadmap

The expected impact of this research on the critical gaps identified in Section 3.4.4 is shown in Table 9.

Table 9: Design and Test Reuse and Synthesis Impact on Critical Gaps

Identified Gaps	Impact
System Requirements	The Reuse of Design and Test artifacts and methods provide the ability to leverage existing systems requirements in these areas. Pattern mining and documenting efforts can ensure requirements, architecture, design and tests are leveraged singularly and synergistically (as a pattern language) to provide integrated reuse.
Low-Overhead Communication	If automated, the mining of patterns from existing designs will be a means of reviewing and analyzing existing intellectual property for reuse. Tools can also be developed to determine how to best catalog this information and make it readily available to those who conceive, specify, design, test, build and deploy systems. Additional tools can be developed to facilitate tagging intellectual property at the time of creation to improve the efficiency in which it is cataloged and mined for reuse.
Architectural Design Support	The reuse of architecture and designs through architectural patterns may directly reduce the cognitive load and effort required to understand and design new systems. The creation of pattern libraries for this work, along with these capabilities and attributes, provides the potential for tremendous increases in productivity.
Risk/Opportunity Management	The reuse of existing capabilities provides the ability to quickly assess the challenges, costs and capabilities of developing new systems which may be constructed of these building blocks which reduces risk and produces the means to see what is possible. During architectural design and implementation, risk can be managed through the use of existing intellectual property that is characterized and tested. Developing the test plans and artifacts during architecture and design phase ensure the architected system is testable.
Verification & Validation	Existing verification and validation test suites are a tremendous asset that should be leveraged in future system development. The reuse of existing architecture and designs facilitates their direct reuse. In addition, verification and validation test suites can be mined to provide building blocks which can reduce the effort in the development and synthesis of new tests.

Legacy Integration	The reuse of existing requirements, architecture, design and tests has clear advantages when working with legacy systems. This reuse is more likely to be compatible with the existing system than if this work was done from scratch.
Human Aware/ Self-Adaptive	The reuse and synthesis of existing system elements, particularly those that have already been proven to be effective with human operators, is advantageous in the development of human aware and self-adaptive systems. In particular, the reuse of human models that have been developed in the HSI research area provides the capabilities to continually improve the capabilities of the system.
Complexity Handling Capabilities	Reuse and synthesis provides an effective means of reducing the design and test effort in complex systems. Rather than having to rebuild each system from the ground up, it can be composed at higher and higher levels of abstraction. These capabilities are what have enabled the electronics industry to use the potential unleashed by Moore's Law.
Cycle Time Reduction	Reuse and synthesis provide huge benefits in the reduction of cycle time. It is only by leveraging and reusing existing intellectual property, and working at higher and higher levels of abstraction that exponential rates in productivity can be achieved. It is anticipated that cycle time for architecture and design may be reduced in excess of 20% based on other historical examples.

The roadmap for Design and Test Reuse and Synthesis research will be performed over a 3-year period with specific deliverables associated with each year as shown in Figure 23.

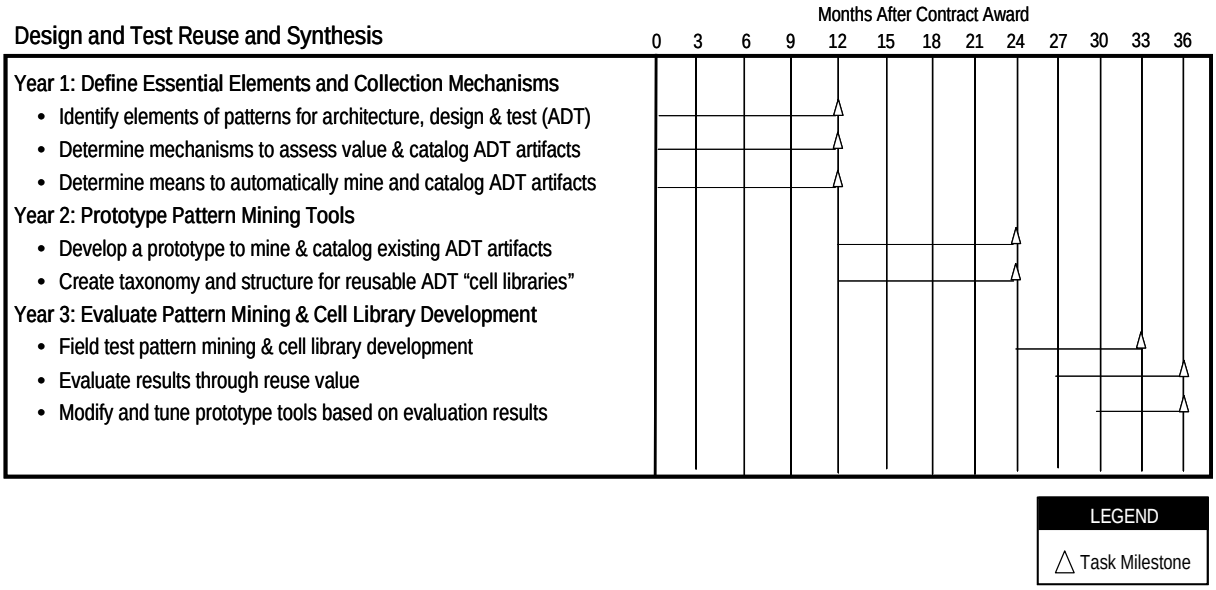


Figure 23: Architecture and Design Reuse and Synthesis Research Roadmap

5.2.5 ACTIVE SYSTEM CHARACTERIZATION

5.2.5.1 Problem and Opportunity

It is very difficult to understand how systems actually behave based on design documents and models. In addition, systems are generally composed of legacy systems, subsystems and components which interact in complex and often unforeseen ways, sometimes using many "undocumented features". The complexity of these systems makes it exceedingly difficult to predict which and when these interactions are critical and significantly impact realized behavior. Transient behaviors, which are generally difficult to understand, tend to dominate the interesting behavior of the system. Finally, with the introduction of the human element, systems may be used in ways which were not envisioned by the developers. However, for new capabilities and features to be added to an existing system, it is often critical to understand its current behavior.

While model-based engineering can be made to be quite effective, it is only as good as the models that it uses. While this may not be a major issue in the development of self-contained greenfield systems with few human interactions and accurate existing models, model fidelity is much more problematic in the development of new features and/or capabilities for systems which depend on existing infrastructure or systems that have a high degree of human interactivity or legacy. In fact, it might well be that most system developments have to contend with legacy system modeling issues. As a result, to get the full benefits of model-centric engineering, the existing, active system needs to be understood and characterized.

Significant research has been done in the instrumentation and analysis of networks for end-to-end performance [Karcali, 2007]. In addition, due to the complexity of contemporary servers and the software stack used in network applications, empirical methods are increasingly being used to analyze and understand systems [Hoffman, 2007; Wang, 2009]. Some providers of internet services have instrumented their server rooms and clients, and have the ability to logon and use these services on a global basis 24/7/365. The resultant data can be analyzed to determine performance and availability effects due to time and date, global position and the types of services being using. This information has been used to predict future server and networking needs, areas of applications to improve, and how to optimize the use of marketing and promotions [Wade, 2010]. Techniques have been devised to automatically generate simulation models based on the measured responses of the system [Pinzger, 2008] and have even used these measurements as an active control system to ensure system level performance guarantees [Lu, 2006]. Google has extended this practice to test all new services, algorithm modifications and artwork with online monitoring. User performance is carefully measured to determine clicks per second and the subsequent value which is provided to the user. These measurements effectively incorporate the human element into the system being monitored and are used to determine future deployment decisions and future feature specification [Helft, 2009].

While much has been accomplished in a number of key areas, previous research has tended to not take a broad systems view. Efforts to understand these effects are often ad hoc and limited in scope. Outside of academic work and certain aerospace and civil engineering applications, measurement efforts are usually deployed once a problem has been encountered in an effort to find an immediate solution. Often the approach is to simply deploy the system and then fix problems as they arise. Unfortunately, this approach usually does not result in deep understanding of the systemic issues and often results in superficial solutions. In addition, it does not provide the insights that can improve future systems with respect to their conception, design, implementation and deployment. This is a major area of opportunity as system developers who are able to instrument, monitor, analyze and make decisions instantaneously will quickly develop a level of expertise and agility that will be a major long-term competitive advantage.

5.2.5.2 Proposed Research Advances and Benefits

This work entails a systematic approach to the instrumentation of the active system such that it can provide data that can be used to characterize it and build the necessary models to support time-based, frequency-based and structural analysis. These models can then be used both to provide an enhanced environment for development and also provide information to determine the system effectiveness. For example, in the internet services domain, systems have been instrumented to provide real-time information on system response times under various amounts of load, at different times of day, location and in the presence of system failure and faults. This information can then be used to determine the current system's capability which impacts the development of future applications, how the system will need to be configured to support future loads and provide the desired level of availability. The system can also be instrumented, as in the case of Google, to instantly measure customer activities and response times to determine the impact of changes that they make in the system. Thus, the impact of system changes can be measured globally and a determination can be made whether or not the changes create sufficient value to justify wider deployment. System behavior can be monitored to determine statistically the difference between normal and potentially malicious activities and take the necessary counter measures.

The development system can also be instrumented and this information can be used to tune the methods, processes and governance used throughout the system lifecycle. Work in this area is synergistic with the research in the Agile Process Engineering area. In both cases, the active system is monitored and this feedback is used enable system adaptability.

The potential benefits of this work include:

- Improving understanding of how actual systems behave and are being used
- Facilitating decision making in new application/feature conception, architecture, design, implementation and deployment

- Facilitating decision making in architectural changes and upgrades to existing systems
- Increasing predictability in performance and reliability/availability of new applications/features being deployed in legacy environments
- Improving efficiency in forecasting the need for and provisioning of system, service and support resources

5.2.5.3 Roadmap

The Active System Characterization (ASC) research objectives are to determine how existing systems can be instrumented, measured in operation, and analyzed to provide the necessary fidelity to system models to aid in subsequent design efforts, and also to validate how the system is being used which impacts future application and feature concepts, and operational training. While there are many possible applications of active system characterization, the first initial major focus will be to provide information necessary for the development of performance/scalability and dependability modeling and analysis tools. The second focus area will be in the instrumentation of the active system to collect data which can be used to assist in the determination of the effectiveness of the system.

The research plan is to initially create a framework for instrumenting, measuring and analyzing active system behavior. This will then be applied to a domain such as net-centric services and a subset of system applications. Tools will be developed to assist in the instrumentation, data collection and analysis of an actual system. Based on the results obtained, these capabilities will be expanded more broadly to additional applications and domains.

The expected impact of this research on the critical gaps identified in Section 3.4.4 is shown in Table 10.

Table 10: Active System Characterization Impact on Critical gaps

Identified Gaps	Impact
System Requirements	ASC provides the capability to understand both how the current system is actually being used, and also how well it is performing under the current load. This information is critical to determining the system requirements for future feature enhancements and upgrades both with respect to the features that will create value, and also with respect to the system capabilities that are necessary to support them.
Low-Overhead Communication	ASC provides a very low-overhead means of communicating information pertaining to the current system. This communication can be automated such that the models used to represent the current system can be updated almost instantaneously without any human intervention.

Architectural Design Support	Understanding the capabilities and actual usage of the existing system is critical to the development and analysis of architectural models. ASC can provide the ability to ensure fidelity in these models which greatly aids the decision making process.
Risk/Opportunity Management	Risk and opportunities can best be managed by understanding the current state of the system and how it is being used. ASC provides the ability to understand how close the system is being run to its actual limits and which parts of the system are being stressed with the current usage patterns. This information can be invaluable in the ability to manage opportunity and risk.
Verification & Validation	Knowing the state of the current system and how it is being used provides guidance on what needs to be validated and verified, and given the risks noted above, where the V&V efforts will have the greatest ROI.
Legacy Integration	Without knowing how your current system is working, it is extremely difficult to perform legacy integration with anything but a "break and fix" mentality. ASC provides the ability to do this systematically throughout the development process through modeling and simulation.
Human Aware/ Self-Adaptive	ASC provides the ability to monitor the behavior of the human agents in the system. Without this monitoring, it is often difficult to know how the system is actually being used. This monitoring can be used to facilitate HSI work to accurately model the human agents in the system.
Complexity Handling Capabilities	With complex systems it is almost impossible to understand how they work, rather the best that you can hope for is understanding how they behave. ASC provides the ability to understand behavior and from this some heuristic based models can be developed to predict future behavior. Extremely complex system may need this level of monitoring to provide the necessary feedback to stabilize the system and ensure long term dependable operation.
Cycle Time Reduction	Understanding the state of the current system provides information that can be used to more effectively allocate resources throughout the life cycle in concept exploration, architecture, design, verification, training and deployment. Not only is unnecessary work avoided, but the probability of failure late in the development cycle is reduced which can provide a major reduction in time.

The roadmap for Active System Characterization research will be performed over a 3-year period with specific deliverables associated with each year as shown in Figure 24.

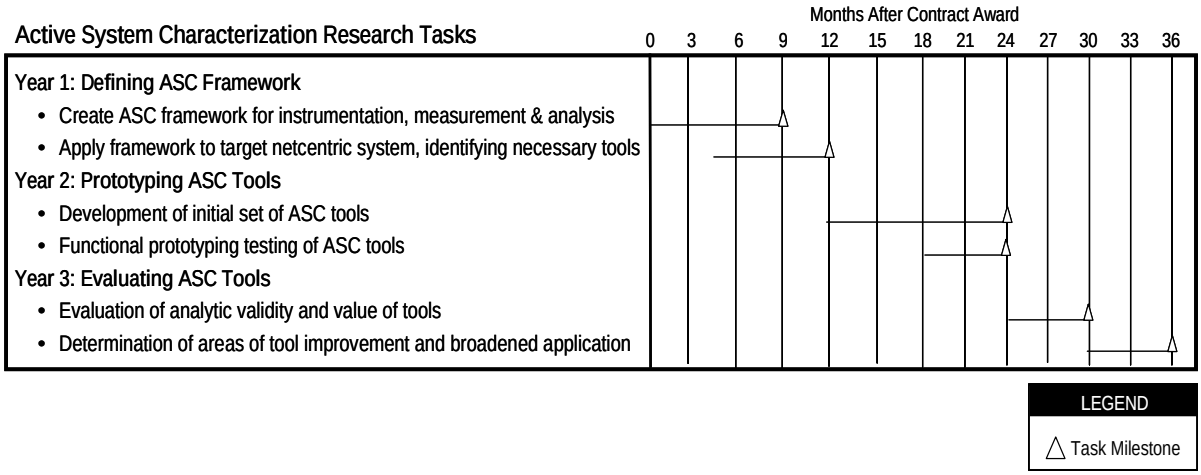


Figure 24: Active System Characterization Research Roadmap

5.2.6 HUMAN-SYSTEM INTEGRATION

5.2.6.1 Problem and Opportunity

Historically, engineers and designers have built human-machine systems with the mindset that the system has to shore up or compensate for human shortcomings [Madni, 2008, Madni, 2010]. In other words, humans are viewed as suboptimal job performers. This mindset fails to capitalize on human ingenuity and creativity. Furthermore, current methods rely on human factors engineering tools that tend to focus mostly on the front-end of the Systems Engineering life cycle. Specifically, current methods do not address human adaptivity, human interaction with adaptive systems, and where and how human-system integration (HSI) considerations need to be introduced in the system life cycle [Madni, 2005; Madni, 2010b]. Finally, existing methods are ill-suited to dealing with human behavior because they do not take human variability into account.

In light of the foregoing, what is needed are human behavior representation models that explicitly reflect human cognitive limitations, human adaptivity limitations, and human cognitive strategies for coping with task overload conditions [Madni, 2010c]. As importantly, the integration of humans with adaptable systems needs to be investigated with a view to identifying "cognitive coupling" requirements during system adaptation [Madni, 2010d]. HSI research needs to address both development time human considerations and operation time human considerations where humans are part of the overall system and interact with other elements of the system.

5.2.6.2 Proposed Research Advances and Benefits

The research that needs to be undertaken to exploit the aforementioned opportunities is reflected in the following objectives:

- Develop innovative HSI methods for integrating humans with adaptable systems
  - identify key human characteristics that need to be reflected in human behavior models
  - identify key adaptability requirements that have an impact on HSI
  - specify new HSI methods that overcome limitations of existing methods
- Identify where and how HSI considerations need to be introduced within the system engineering life cycle
  - choose an appropriate Systems Engineering life cycle model
  - identify key HSI considerations that need to be addressed in different stages of the SE life cycle
  - develop methods and tools to incorporate key HSI considerations in the Systems Engineering life cycle
- Maximize compatibility among tools and tool users (i.e. developers)
  - identify developers, skill set and tool usage competency
  - prototype developer-tool interactions and conduct usability testing
  - identify key features of the tool that specifically address developer characteristics
  - prototype tool that reflects new HSI considerations during system concept engineering, design, test and evaluation (T&E)

The benefits of this research are: superior human-system integration in dynamic operational environments—the key to agility in the operation theater; superior HSI between developers and tools leading to increased productivity during system development.

5.2.6.3 Roadmap

The expected impact of this research on the critical gaps identified in Section 3.4.4 is shown in Table 11.

Table 11: Human-Systems Integration Impact on Critical Gaps

Identified Gaps	Impact
System Requirements	Explicit identification and management of "cognitive coupling" requirements – implications of adaptability on HSI.
Low-Overhead Communication	More efficient communications owing to greater shared understanding that comes from common vocabularies and ontologies and MPTs throughout the SE lifecycle that accommodate human limitations and leverage human cognitive strengths.

Architectural Design Support	Maximizing compatibility between tools and tool users with respect to cognitive load and cognitive problem-solving strategies.
Risk/Opportunity Management	Proper accounting for HSI requirements, risks, and opportunities in the system trade space.
Verification & Validation	Methods for the verification and validation of HSI requirements in general, and human/system adaptability specifically.
Legacy Integration	Insight into how well legacy systems accommodate HSI requirements, particularly with respect to human adaptivity limitations and human interaction with adaptive systems.
Human Aware/ Self-Adaptive	This is the main focus of this research area. It addresses what it takes to maximally exploit human adaptivity characteristics while maintaining human error rates at acceptance thresholds.
Complexity Handling Capabilities	Reflecting advances in human-centered computing and naturalistic decision-making, HSI considerations in full lifecycle MPTs ensure that accommodation is made for the strengths, limitations, and preferences of the human users.
Cycle Time Reduction	Increased productivity due to rapid convergence of the development teams' collective understanding of both the problem and solution options.

The roadmap for Human-System Integration research will be performed over a 3-year period with specific deliverables associated with each year as shown in Figure 25.

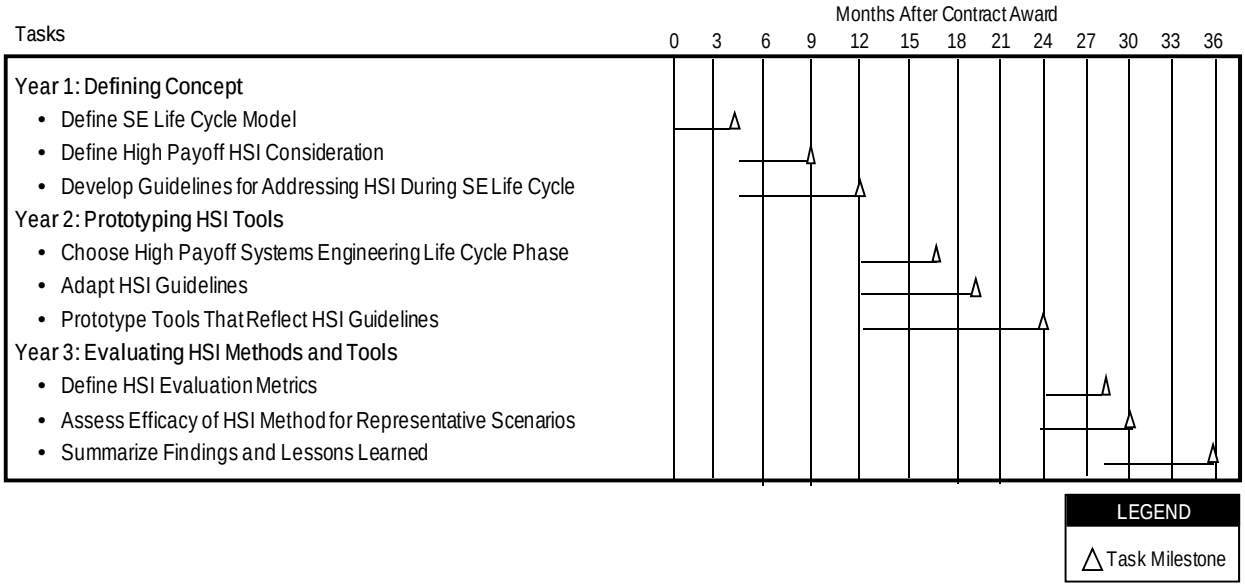


Figure 25: Human-System Integration Research Roadmap

5.2.7 AGILE PROCESS ENGINEERING

5.2.7.1 Problem and Opportunity

Ultimately, the usefulness and effectiveness of new Systems Engineering approaches and Methods, Processes and Tools will be heavily dependent on how they work together in the overall SE process. No one process, or even set of processes, can meet the needs of all the various environments where SE is applied. Furthermore, many SE environments have become less predictable and more fluid, thus limiting the effectiveness of long term, heavyweight SE process improvement approaches. Increasingly, an environmental challenge changes before an improvement can be identified, vetted, piloted and deployed.

The processes that support, control and validate new SE practices need to be both flexible and resilient to adapt to environmental changes and still provide useful and up-to-date information. The creation of such processes is called Agile Process Engineering. Agile Process Engineering will address several shortfalls of the current state of the practice:

- Process models (such as CMMI / ISO / ITIL) are managed at an organizational level, not by process users, resulting in a risk-averse rather than opportunity-aware environment. Thus, scenarios where a change in process may yield improvement are avoided because of misperceived risk.
- Process compliance is too often valued over efficiency and effectiveness.
- Current MPTs are insufficiently engineered to adjust rapidly to changes in process because they lack modular functionality and integrated data.
- There is a general lack of a knowledge base on how to adapt effective “agile” practices into SE and acquisition environments.

5.2.7.2 Proposed Research Advances and Benefits

Agile process engineering is a multi-step process. The first step is to identify concrete challenges within the customer environment, such as rapidly changing requirements or managing concurrent development and sustainment activities. The second step is comprised of two concurrent activities: formally modeling the customer’s process using a modeling framework and identifying MPTs capable of meeting the customer’s challenges. After candidate MPTs have been identified for addressing the sponsor’s challenge, the third step involves modeling the proposed MPTs and identifying how the MPTs can be engineered to fit into the customer’s environment. The final step is to put the appropriate monitors in place which provide the means to determine when and how the MPTs need to be modified to address changes in the environment. The result of these steps is a hybrid Systems Engineering process designed expressly to meet the customer’s current and future challenges. This process is deemed agile process engineering because it first formally examines the customer’s situation, formally examines the options for altering the customer’s process while adhering to

organizational constraints, and generates a solution unique to the customer and their ongoing challenges.

Agile process engineering is critical to realizing the benefits of interactive, model-centric engineering. Agile processes will help organizations implement and adopt MPTs to enable rapid deployment, to provide support for changing and emergent requirements, and to adjust to the scale and complexity of the project. Agile process engineering will also incorporate and optimize necessary governance and oversight characteristics, and adapt the process according to the size and distribution of the team.

The objective of this research is to provide increased efficiency and effectiveness throughout the system life cycle, by delivering:

- MPTs that are adaptable to support changing environmental and project conditions
- Management approaches and tools to enable the effective use of adaptable processes
- An infrastructure to support the above activities

5.2.7.3 Roadmap

Research in Agile Process Engineering will take place in the two areas described below.

Research Area 1 - Defining Effective, Agile Processes
Work in this area will identify the basic principles that processes must address in order to be effectively agile, identify the concrete MPTs that can be applied by teams to enact those principles, and identify the methods of composing MPTs to ensure that the overall program needs are being met. Processes will be considered as systems, following a development approach with the following phases:

- Phase 1 - Define agility requirements: Extend and deepen previous SERC research, analyzing the critical functional elements of MPTs that enable agility. Understanding these critical elements enables the creation of new MPTs ready for the Systems Engineering domain. Extend and create bridge diagrams [Turner, 2009b] for visualization of results and create a knowledge base documenting effective MPTs for dealing with common challenge areas.
- Phase 2 - Define governance requirements: Identify research management approaches for adopting agile process engineering and implementing process change. Also, identify feasible measures of process agility, process compliance and project status. Investigate mechanisms for automatically collecting these data.
- Phase 3 - Develop an architecture and design for an end-to-end agile process: Use bridge diagrams to match candidate MPTs to agility needs. Apply modeling techniques such as “Little JIL” [Cass, 2000; Wise 2006] to understand relationships and failure modes among MPTs. Characterize these MPTs with a

set of attributes that allows one to understand the contexts in which they are appropriate and their impact on overall process flexibility.

- Phase 4 - Ongoing verification and validation of the defined agile process: Provide the feedback loops necessary to ensure that proposed solutions are applicable and effective. These may include feasibility checks, modeling and simulation efforts, and empirical studies under representative conditions. An implementation of an agile process will be targeted for a specific domain and its effectiveness monitored in use.

Research Area 2 - Management of agile processes: Tools for Selecting, Tailoring, and Governing Agile Processes

Research Area 1 will produce results that will help a team choose the right agile MPTs for their needs and context. However, the development context needs to be managed so that when conditions change, the MPTs being applied can change too, without undue impediment from unnecessary process compliance. This research area will develop and experiment with tool support aimed at making this vision feasible and practical. The research will involve the following activities:

- Investigate the automation of metrics collection and visualization: Research in Active System Characterization will inform the instrumentation of various MPTs. Aspects such as complexity or change density of portions of software-intensive systems can be monitored and this information used to suggest different processes for those components (e.g. investing time in refactoring or additional V&V activities) [Olbrich, 2009; Zazworka, 2009]
- Facilitate the composition of MPTs into an overall agile process: The research will determine how best to represent, manipulate and tailor MPTs to the specific environment and application, with the requisite adaptability in the areas with the greatest likelihood of change. Based upon satisfying the project needs and constraints, an easy-to-use decision support tool will be developed. The tool will exploit the wealth of information represented in the bridge diagrams of Research Area 1.

The expected impact of this research on the critical gaps identified in Section 3.4.4 is shown in Table 12.

Table 12: Agile Process Engineering Impact on Critical Gaps

Identified Gaps	Impact
System Requirements	Processes that support changing requirements are key to agility.
Low-Overhead Communication	Operating on tacit knowledge is the essence of low-overhead communication. Processes must be engineered to take advantage of models, information radiators and other low-overhead techniques.

Identified Gaps	Impact
Architectural Design Support	Agile processes may be architecture-informed (i.e. are tailored to support and evolve with the architecture).
Risk/Opportunity Management	Agile processes allow changes to address risk or exploit opportunities.
Verification & Validation	Parallel formal process modeling can find critical defects in rapidly engineered processes that would cause significant delays.
Legacy Integration	Agile processes can take advantage of knowledge from experience and legacy-stakeholder participation to make legacy integration more effective.
Human Aware/ Self-Adaptive	Agile processes will support humans more fully by allowing change to be a part of the work rather than a hazard.
Complexity Handling Capabilities	Complexity generally increases change and agile processes lessen the impact.
Cycle Time Reduction	By adapting more rapidly to change, agile processes reduce cycle time.

The roadmap for Agile Process Engineering research will be performed over a 3-year period with specific deliverables associated with each year as shown below in Figure 26.

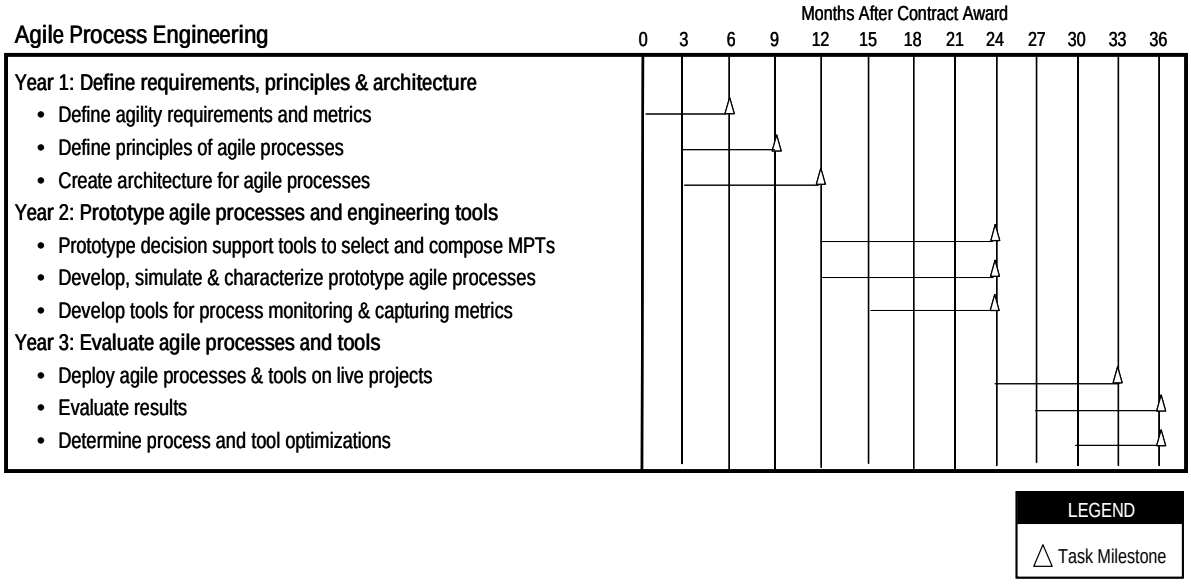


Figure 26: Agile Process Engineering Research Roadmap

5.2.8 MODELING ENVIRONMENT INFRASTRUCTURE

5.2.8.1 Problem and Opportunity

An integrated modeling environment is required to provide effective and coherent communication and management of information and knowledge among the users of the collaborative modeling and design MPTs. This environment needs to provide the means by which all of the various forms of models, which represent different aspects of systems, can communicate both semantically and syntactically. In addition, there is the need for efficient model synchronization and dependency notification across the entire life cycle of system development. Model consistency is required to facilitate parallel development without the need for unnecessary sequential restrictions. The development of standards will be necessary to enable the use of heterogeneous tools in this environment, and to enable exchange of information between technical disciplines. In addition, the environment should provide the ability to effectively manage the models and data over time (the life cycle) while lowering the overhead necessary in locating, accessing and using them. Finally, this environment should provide, or at least allow for, the capability to mine the model and data repository for patterns and provide the means to effectively create and reuse them.

The following are a number of the limitations of current modeling environments:

- Tools and processes in today's system development environment are highly stove-piped and sub-optimized to a particular discipline
- Interchange of information is often manual and error prone, thus introducing schedule delays and added costs
- Participants upstream in the life cycle become disconnected from lower-level decisions downstream and vice-versa, especially in a changing environment
- Complete, prioritized needs and requirements are not used consistently across the life cycle to make value based tradeoff decisions
- Tools do not enable cross-disciplinary tradeoff decision making and recording of decisions
- Legacy capabilities are not well understood or documented making integration difficult and unpredictable

The Modeling Environment Infrastructure is the substrate for providing fluid communications across the life cycle to enable priority based tradeoff decision making and recording. It provides views of information that are tailored and relevant to inform the receiver of information and facilitate decision making. The transformation of Systems Engineering is clearly dependent on the capabilities of this environment which will be leveraged by all of the research areas described earlier.

5.2.8.2 Proposed Research Advances and Benefits

This research will focus on the development of an environment which provides effective modeling capability, model interoperability and management across the life cycle. In addition, work will be done to understand how the environment can best present relevant information to each user and increase design and test efficiency. This research will provide the technical infrastructure to increase the capabilities in the other research areas.

The research objectives are to:

- Research, define and create a design and development environment which facilitates the rapid modeling and iterative refinement of user needs and priorities into concepts, designs and fielded capabilities that deliver value and expected utility to the interested stakeholders
- Develop integrated/federated methods, processes and tools which encapsulate knowledge to guide how and when to perform specific activities and tasks with associated execution aids that enable the human to manage, and act on multiple complex dimensions of information across the enterprise/project life cycle including:
    - manage prioritized stakeholder needs – functional, physical, behavioral, performance and other 'ilities'
    - address multiple levels of system detail (Level 0 – n)
    - enable interdisciplinary tradeoff and decision making
    - address forward and backward communication across the life cycle (synchronous and asynchronous)
    - automate repetitive and algorithmic tasks where possible
    - enable the integration of legacy capabilities

The following are the novel aspects of this research:

- Use a comprehensive life cycle approach to facilitate the flow of information from need to fielded capability
- Follow a comprehensive approach to prioritizing and managing multiple stakeholder needs and requirements across the life cycle – portfolio based approach
- Emphasize the use of standards as a means for efficient interchange of information and data in both a forward and backward direction across the life cycle
- Continuous verification and validation across all life cycle phases

There are many potential payoffs including the following:

- Enables richer cross-disciplinary communication across the complete life cycle for priority based tradeoff decisions at multiple levels of design and development detail

- Enables the managed evolution of user needs and changing requirements through multi-level, cross-disciplinary communication
- Results in systems which realize prioritized stakeholder needs and requirements, and facilitates the management of stakeholder expectations through improved communication of priority based tradeoff decisions

5.2.8.3 Roadmap

One of the major challenges with research in this area is the enormous scope of the challenges at hand. Rather than attempting to solve each of the major challenges across the entire space, focused efforts will be made in the highest payoff areas to successfully demonstrate proof of concept in a particular domain which can then be extended more broadly. For example, one of the highest leverage/payoff areas is the accurate and rapid translation of user needs into alternative system concepts with balanced tradeoff criteria that represent the collective prioritized needs of a diverse set of stakeholders. In particular, this research will focus on software intensive system development environments, rather than ones which have significant amounts of non-computational hardware. The former types of systems are more easily virtualized thus lowering the barriers somewhat to breakthroughs in environmental capabilities. Systems dominated by non-computational hardware tend to have environmental challenges in linking multi-physics types of simulations. Linking these simulations presents significant physical science challenges outside the realm of Systems Engineering, particularly when they need to be closely linked in small time and physical scales across multiple physical domains (e.g., chemistry, material science, electromagnetics, thermodynamics, etc.).

The major research focus areas are to improve communication in the following three modes through the use of tools, technology and governance:

- **MS2MS** - models and simulations (M&S) to M&S: appropriate semantic and syntactic operation, with consistent assumptions pertaining to time, space, cost, etc. One initial target area may be in the interfacing of Prioritization, Conceptual, Behavioral and Architectural modeling descriptions with actual design.
- **H2H** - human to human: ability to present information in the proper representation both for the sender and the receiver, and provide automated notifications on both sides determining when communication is most advantageous. One initial target area may be communication and decision making across a set of diverse stakeholders who are directly involved with the conception, specification, architecture, design and test, and validation of a system.
- **H2MS** - humans to/from M&S: ability for the human to understand and efficiently navigate the modeling, simulation and data repository; and for the M&S to determine who to contact when, with what information; and to support governance models of who has the ability to access and commit which information to support parallel development. One initial target area would be in

automated notification of critical information to the parties directly involved with system development of changes that impact them and provide expert assistance in locating information that is useful to them.

Note that the most likely approach for this research would be to analyze the issues that are faced in a sponsor's environment and determine how advances in the above areas could provide the greatest ROI.

The expected impact of this research on the critical gaps identified in Section 3.4.4 is shown in Table 13.

Table 13: Modeling Environment Infrastructure Impact on Critical Gaps

Identified Gaps	Impact
System Requirements	Provides an environment to develop and capture the functional and non-functional requirements of a diverse set of stakeholders, and manage their collective priorities across the life cycle as requirements change.
Low-Overhead Communication	Facilitates the efficient interchange of information and knowledge among a set of diverse stakeholders who represent different disciplines, specialties, technologies, organizations, geographic locations and work cultures.
Architectural Design Support	Enables the creation of alternative system concepts, along with tradeoff criteria that enable a diverse set of stakeholders to make a concept selection based on the collective priorities of the stakeholders. Enables the application of design margin into the architecture and design to accommodate flexibility for future requirements.
Risk/Opportunity Management	Treats risk/opportunity as an integral part of the system development process and enables the entire development organization to continually identify, assess and manage risk across the system development life cycle.
Verification & Validation	Establishes an environment that supports continuous verification and validation with traceability to the system requirements. Enables the linkage of verification and validation scenarios and scripts back to the requirements and provides automated verification and validation thus reducing cycle-time.
Legacy Integration	Provides a set of technologies that characterize legacy system functional, non-functional, physical, behavioral and interface capabilities, thus facilitating the accurate and rapid inclusion of legacy systems into new system, or system-of-systems development projects.
Human Aware/ Self-Adaptive	Enables the creation of human friendly systems by providing technologies that involve the human end-user in the specification, design, development and testing of systems, ranging from single-use to complex human aware and adaptive systems. The benefit is the creation of systems that provide greater utility and value to their human end users.

Complexity Handling Capabilities	Manages multiple, simultaneous and often conflicting dimensions of a system and assists the set of diverse stakeholders to organize and manage the increasingly complex information and decision making across the life cycle. Reduces cycle time and cost, and increases quality by managing complexity.
Cycle Time Reduction	Shortens cycle time by facilitating the efficient exchange of information and knowledge across a diverse set of stakeholders from a variety of disciplines, specialties, technologies, organizations, geographic locations and work cultures, to enable quick and fact based decision making.

The roadmap for Modeling Environment Infrastructure research will be performed over a 3-year period with specific deliverables associated with each year as shown below in Figure 27.

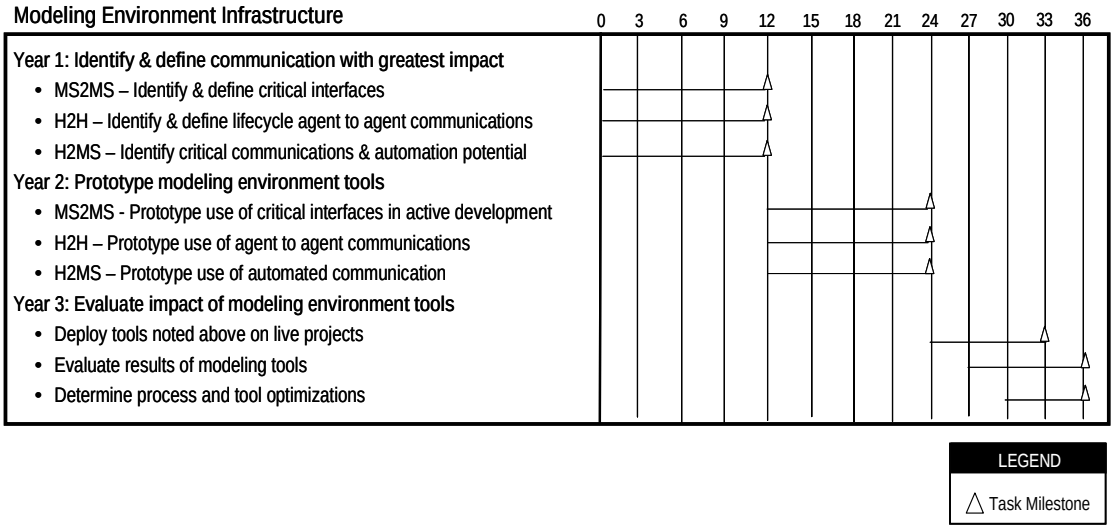


Figure 27: Modeling Environment Infrastructure Research Roadmap

5.3 ROADMAP SUMMARY

The objective of this research topic is to investigate and create a 3-year roadmap for the transformation of Systems Engineering necessary to address the emerging critical challenges. This involves understanding the emerging critical challenges (Section 1), determining the current state of the art and gap areas in Systems Engineering (Section 3), formulating a vision for the future of Systems Engineering (Section 4), and finally constructing an integrated and modular roadmap of innovation necessary to support this transformation (Section 5).

Each of these eight research areas satisfies the following criteria:

- Critical to the transformation of SE
- Furthers the sponsor's mission

- Requires multidisciplinary research which is not currently being done
- Appropriate scope and scale for an academic research program
- Supports a 3-year or longer roadmap of research
- Anticipated to have measureable impact

Each of the research areas has capabilities which leverage the current state of the art in computation, visualization, communication and information technologies. Future advances in these areas will only increase the capabilities of the technologies developed in these research areas, thus keeping Systems Engineering "on the curve". However, success in these areas is not dependent on future technology advances.

In this section, an integrated, modular framework supporting the transformation of Systems Engineering has been presented along with two operational scenarios which describe how the system might be used. Then, each of the eight research areas were described with respect to the challenges that they are addressing, the opportunities, the research advances and expected benefits. This is followed by a description of their impact on the identified critical challenges and a high-level three year roadmap for the research. These descriptions note how these research efforts are integrated, such that they interact closely with one another providing value that is greater than the sum of their parts, yet remain modular such that each area can proceed and provide value independently.

Due to the integrated, yet modular nature of this work, the funding of the SET roadmap is extremely flexible with respect to amount, time and source for each research area. The modular nature of the roadmap allows for independent funding sources with varying timeframes. While no research area is directly dependent on another research area for success, the integrated nature provides the opportunity for collaborative, synergistic investments. Advances in one research area increase the benefits of research in the other areas.

Taken separately, these research areas have the potential to significantly advance the state of the art of Systems Engineering. Taken together, they have the potential to transform Systems Engineering.

AFTERWORD: AN SE RETROSPECTIVE TALE

The End State

There were two more hours of testing to be completed before System-X would be released and deployed for operational use. Joe Johnson, responsible for its overall success, was confident that this would be a high-quality, on-time delivery. His confidence was based on the fact that this was the 7th in a series of releases over the past couple of years that had continually gotten better. They were currently releasing this product on a quarterly basis, but there was talk about even more frequent releases. Joe enjoyed this talk, since the challenge to quicker releases wasn't in development and test now, but in fielding. It simply took too long to train users in the field how to apply the new features appropriately. Clearly more work was needed to support rapid deployment, but since the company was now focused on constant improvement, he imagined there would be effective changes soon. Of course, it hadn't always been like this. Critical challenges had required dramatic changes a few years ago, including the emphasis on improvement.

The Old Model

Keeping an eye on the test status dashboard, Joe let his mind wander back to those pre-crisis days, when it was rarely a question of shortening cycles. In fact, there always seemed to be enough time to "do things right." His organization was given a clear mission, which the entire team understood, at least at some level. They spent lots of time with the stakeholders building a detailed set of needs and understanding the system context. There was a whole team of SEs to translate those needs into a complete set of requirements, which became the basis for building a concept of operations, and an architecture. Requirements were decomposed into functional blocks, assigned to appropriate development organizations to build components (which were always tested extensively) and then everything was integrated into the system. Joe chuckled, remembering that integration had always been a bit of an eye-opener and the rework could cause real panic. But, because of the built in slack in the schedule, they were usually able to work things out eventually. This was the phase where the corporate heroes were often trotted out and managers and teams made or lost. His team had been through a number of these integrations, but they were good and tended to communicate better than some of the other teams and so usually survived. They generally put their system through a series of increasingly more difficult tests and fixed things as they went. This modified "V" process was certainly better than the old Waterfall and had seemed to serve them well.

The Challenge

Joe couldn't put his finger on when exactly it happened, but things started to breakdown over time. It really happened across the entire process. There were all sorts of

problems – things just seemed to have to happen quicker. The pace of change, losing control of the environment and technology, and the complexity of the products that made it hard to separate out functional blocks to work concurrently made everything more difficult. There wasn't time to solicit and create a set of requirements, and when they did, the needs changed so rapidly the requirements were useless. It was a endless task, some called it a "rock fetch" or worse, and the team often found themselves just going through the motions to satisfy the auditors. The real work was happening outside the process, because the process couldn't keep up. The same was true about the operational environment. They were forced by budget cuts to outsource more and use COTs whenever possible. It sure sounded great on paper, but in reality they lost control of the technology—ensuring a stable supply chain and product roadmaps for the essential system capabilities was a nightmare. In the good old days you could usually see what you were building. Sure, there was some software thrown in, but the system decomposition was generally fairly obvious. Now software was the critical element and drove most of the complexity.

Decomposition had become a major challenge and it seemed as though everybody had to talk to everyone all the time. Why couldn't everyone just know what they had to develop like in the past? The tradeoff seemed to be either everyone could spend their time in meetings and know what to do, or they could do development work with the likely odds that it wouldn't integrate. Well, at least if you got your work done you could blame someone else if it didn't integrate, so most folks just plowed in to the development. Validation testing was even more challenging since the validation crews didn't have a clear specification of what the system was supposed to do when it reached their shop. They did the best that they could with what they had, but ended up spending most of their time in integration work and ensuring that the system was at least not DOA. The customers finally did most of the actual validation work and were none too happy about it. They were even less happy when the features and functions delivered weren't the ones they needed. In fact, most new features had very little value. Nothing was going right. Joe had felt like the proverbial frog in that increasingly hot pot of water. He couldn't really tell when things started to get uncomfortable, but he knew that if they didn't jump out of the pot the consequences would be dire. Luckily, he had made the decision to jump—bringing the entire team with him.

Critical Insights

Taking a quick hit of coffee and scanning the floor for any unusual activity, Joe sat back in his chair and actually shuddered a bit when he thought about that "call to action" offsite. Everyone knew they had to make changes, but no one seemed to know where to start. Then there was that eureka moment when suddenly they collectively seemed to realize that the "V" had it all wrong. You don't do validation at the end of the process, you do it at the beginning and you do it continually throughout the program. Validation is necessary to ensure the system provides a sound value proposition, and needs to be continuous or else time and effort are wasted on low value work. But how to accomplish that when there isn't a complete system in place until after it's designed? And how to

get rid of the constant interruptions of sync up meetings currently plaguing their development efforts?

The only approach that seemed sound was to create an environment where simulation was used to model the system throughout the lifecycle. Hierarchical models could be filled in with the level of details that were needed and available at each point in development. Having a shared model in place as a central reference point would allow each member of the development, test and deployment teams to work coherently without having unnecessary forced synchronizations. They could be informed of changes that impact them, and disregard the rest. The intent was to create a common shared "Borg-like" state. This new environment would need to be: interactive supporting iterative design, execution and re-design; based on executable hierarchical models that provide on-the-fly simulation and can be used throughout the lifecycle; and supportive of multisensory and multimodal inputs and outputs to provide information personalized to the needs of the stakeholder or user.

Their second major insight was that the standard approaches to system analysis and design, based on decomposition and allocation of functional requirements of the system, usually generate systems that are resistant to adaptation and difficult to secure in the faces of changing threats. They needed a design approach that reflected the true objectives of the system—security, flexibility, performance, etc.—and mechanisms that balance the mission objectives with the functional capability and cost/value.

Both new system knowledge and tools would be necessary to bridge the gap and truly bring value-based system engineering to life. Analysis tools would determine both the value of flexibility and the means by which to achieve it. Other tools that could automatically perform options analysis across trade-spaces to determine value were essential in the decision making process. Everyone agreed that their current tools simply couldn't provide architects and designers with the information to understand the emergent behaviors of their complex systems. They also agreed that the team members were proficient at visual pattern recognition and analysis, but their capabilities of understanding these relationships from text-based information was really limited. Clearly work had to be done to support high-level value decisions involving architectural and implementation tradeoffs.

The final insight was that the systems they developed needed to be adaptive if they were to keep up with the pace of change in the environment, technology and more important, the mission. One of the most adaptive elements in any system had been largely ignored to date, and that was the human factor. Understanding the relative strengths and weaknesses of humans and incorporating them into a system appropriately was surprisingly new ground that desperately needed to be developed.

Joe could see now how the confluence of these three major insights resulted in the set of actions and activities that followed. The meeting resulted in a whirlwind of discussions, planning, research, and finally actions to fix the engineering approach.

The Change

Prioritization and Tradeoff Analysis

Joe and his team started at the front end of the process where many of the problems seemed to originate. In fact, the trouble seemed to start even before they had made any decisions. Joe realized that the critical first step in value-based decision making was to determine the relative worth of various system attributes including flexibility, agility, resilience, sustainability and dependability. It seemed no one had a clear understanding of how the "goodness" of solutions could be judged. Everyone knew that lower cost, higher performance, higher reliability and shorter time to deployment were good things to achieve; but there was little agreement on the relative value of each. They decided to try models as a quantifiable means of making these determinations. At first, the models were crude, but they were refined over time and their predictions were tested and tuned based on feedback from the field. Joe had really come to appreciate the ability to show his executive management tangible and quantifiable benefits supporting the various decisions the team made.

Concept Engineering

It was clear to everyone that long drawn out requirements elicitation processes were not working and that most documentation ended up being "shelfware". In fact, the exchange of text-based documents simply could not effectively convey information to their diverse set of stakeholders. The people creating the system concepts needed a tool that they could touch and feel; something where they could quickly iterate and test their concepts. Systemigrams and other graphical tools helped at the whiteboard, but they needed more. Looking for a better solution, one of the engineers modified a graphical modeling language that her 11-year old son was using to program Lego robots and brought it in to try out with the team. When Joe heard about it, he'd figured his folks were just playing around. However, when he decided to pay the engineer a little visit, he found an intense group of engineers, field service and marketing people gathered around a conference table with their laptops manipulating symbols, graphs and watching data change on a big screen. When he had finally gotten their attention, they'd told him, "We're creating the concept for the new System-X project." He'd been blown away. He hadn't thought these people knew each other, let alone speak the same language. Yet, as it was explained to him, there they were—building and simulating concepts and comparing the results based on the prioritization models. Joe had stopped on his way home that night and bought his son a set of Lego Mindstorms. At least he had told his wife that it was for his son.

Architecture and Design Analysis

Although the efforts in Concept Engineering and simulation and model-aided development were paying off, there still were major gaps in the architectural specification and design of the system. In particular, it was not clear to Joe that the decisions being made maximized the value of the system. It was extremely difficult to understand the emergent attributes of the system throughout the development process. There was no longer a single "go to" expert who understood how it all worked, and how each component affected the overall operation of the system. So, they had looked at developing tools that would assist them in determining where changes could be best made in the system to avoid gratuitously breaking things. This work was quite useful in determining which parts of the system suffered from unacceptable levels of "architecture and design rot" and needed to be re-architected, redesigned and refactored. Unexpectedly, though, it also showed which areas of the system were likely to be the least stable and thus the greatest reliability and security risks. The tools evolved to take on more capabilities and support other nonfunctional attributes including flexibility, reliability and performance. The development of these tools was very much an agile process, dynamically driven by what they learned along the way.

Design and Test Reuse and Synthesis

While making great progress with the efforts in the above areas, Joe recognized they still were not achieving the level of productivity that would consistently develop systems at the demanded rate. One of his engineers observed that each system design was essentially a new experience; they weren't really leveraging the work that had been done on previous releases. He suggested that they should focus more on finding and using patterns and on technology reuse. Additionally, they were not taking full advantage of the models created in conceptual design. There were simply too many "one offs" in each of these areas for this to be a productive use of time and effort. A task force was assigned to mine patterns from concepts, architecture, design, implementation and tests. The findings were quite surprising as they confirmed how many different ways they architected, designed and tested the same functionality. Much as has been done in the area of integrated circuit technology, they were able to provide consolidation with the creation of primitives, components, subsystems and systems in a reusable library. In the case where new design was necessary, they were often able to use existing building blocks. The net result was not only in a reduction in time and effort, but also an increase in quality as they were able to use well tested and characterized components.

Integration of V&V into Development

The use of simulation and modeling technology for conceptual and high-level design certainly greatly improved design productivity, but there was still the challenge of proving that the system was correct in function and operation. Already V&V was a huge effort, and it certainly wasn't getting any easier. It required total focus, and most importantly in Joe's company, a major culture shift. At that time in Joe's firm, the V&V staff had not been seen as the most talented and empowered people. In fact, this was where staff traditionally went when they weren't selected for the architecture and design

teams. The V&V team only operated late in the development pipeline, got the product late to test, always had their test time reduced, were forced to compromise their standards, and were immediately blamed for any problems that escaped into the field. In short, they felt like and often were the victims. They often worked manually and reacted to problems by throwing bodies at whatever was the crisis of the day. This was clearly not a road to success.

Joe and his team looked for a silver bullet to solve these problems, but none was found. However, the team had noticed that companies producing high quality systems and had short release times all seemed to have similar characteristics. For example, they all had high quality teams that focused on automating every part of the test process. Humans were removed from the process of setting up, running and collecting test results. Instead, humans were focused on the analysis of test coverage and failure data, determining where additional tests were needed, and if so, how to automatically create them. The other major focus area was to track failures throughout the lifecycle process, provide root cause analyses, and ensure that tests were in place to find them as early as possible in the development/deployment process. To succeed with this strategy required a major commitment across the organization and without executive support it would never have been successful. Test Driven Development was included at the front-end of the process, and design and verification engineers were teamed together. No longer could someone point their finger at an individual for blame for a fault, but rather teams were held jointly accountable for schedule, functionality and quality. They had made much progress, but cultures don't change quickly.

The conceptual and high-level design efforts provided major benefits in V&V as these hierarchical models supported the continuous development and execution of tests throughout the development process. In addition, this model based approach was accompanied with significant levels of reuse which provided the means by which to compose correct-by-design constructions. Finally, it was possible to quickly verify and validate incremental changes to a system to allow for rapid deployment of incremental capabilities. These approaches resulted in substantial and continuing improvement in deployed system quality. Joe noticed that in hindsight the barriers between design and test were artificial, and over time they blended into the single integrated activity of development.

Active System Characterization

Amidst all the excitement, Joe unhappily noted that while greenfield system development was the "cleanest" and resulted in an up to date set of models, he could not remember the last time they had gotten to start with a clean slate. Living with legacy was a way of life. To make matters worse, there was much that they did not know about their legacy system. It seemed to Joe that they discovered all of their system limitations by deploying new system capabilities and then waiting for user complaints. There had to be a better way. Their solution had been to develop a set of automated tools to instrument, monitor and characterize their existing system. This was done both in the

field in the form of a number of system monitors that reported real-time parametric data, as well as within the in-house system testing. Joe was pleased to know that the team made a number of new discoveries about their system and how it operated, and that these had made a significant impact on subsequent decisions.

Human-System Integration

Modeling was a major success for the organization, but it still had a major flaw: its predictions were sometimes very wrong. He saw the same issues with problems in deployed systems. When Joe asked what happened, he invariably got some excuse about operator error or the unpredictability of people. When he tried to pin down the systems engineers, they told him that they were only responsible for the technology, not for the people. After the last system failure due to "operator error", Joe had had enough. He'd called in the SEs, program and product managers and read them the riot act. They were all responsible for the success of the entire system, and that included people, so they better figure out how to include people in the equation. There had been a hushed silence in the room when they realized that the blame game was over–and no one had answers. The organization had some experience with how people actually used the system in their Concept Engineering work, but they needed to take it a step further. They started by having end users interact with the product in the simulators throughout the development cycle. Later they were able to incorporate some models of how people might interact with the system. But, where they made the most impact was in determining how to allocate system capabilities between human and machine to optimize overall system performance.

Agile Process Engineering

One of the major problems that Joe and his team encountered throughout the transition was the simple fact that it was a major transition; one that could not occur immediately and was likely to continue for many years. In fact, if successful, continual change would be a permanent aspect of their future operations. While many of the leaders in his organization talked about processes that enabled agile development, it eventually dawned on them that they needed processes that were themselves agile. There simply was not a golden process that would support the entire organization and the rest of the enterprise continuously over time. This was a breakthrough notion and was perhaps the most difficult bit of cultural change necessary support the new development paradigm. One major advantage, though, was that these processes were embedded in an integrated, highly-automated environment, which kept the complexity hidden from the users. As far as Joe could tell, the intelligence of the processes guided and supported the developers to make them more effective rather than act as the traffic-cop SEs of the past.

Integrated Environment Infrastructure

While all of the tools and processes that had been created were a great help to the team, they still had some major communication problems which limited their success. Joe heard the complaints over and over again that this tool could not talk to that tool, or that

data was not available in the right format, or it was impossible to find the most up-to-date information. At times the project seemed to come to a halt, while everybody synched up to the latest release. Joe called a meeting of his best and brightest to solve the problem. As it turned out, they were able to create some reasonable solutions to permit communication and data exchange between their homegrown tools, but it was a much bigger challenge for the commercial tools that they were using. Joe knew that he could not single-handedly move the market place, so he focused his efforts where they had the greatest impact. In some cases this meant consolidating their tool use and in others it meant creating some internal interfaces and translation technology. Joe had also formed an alliance with some other systems developers to influence the CAD tool vendors, but this was going to be a long, long road. Where they had made some amazing success was in the ability for tool and environment users to find information and communicate with one another. The breakthrough idea here was optimize the system for the user rather than the other way around. Developers were not disturbed with notifications unless there were changes to the system that impacted them. Artificial intelligence was used to make the right information available to the right people. Joe noted that while Amazon had been using the technology for years, it still surprised him when the Integrated Environment seemed to anticipate what he wanted at work. The Environment truly had become a central communication point for all of the involved stakeholders and developers.

Conclusion

Joe checked his watch and noted that two hours had gone by. He glanced at his monitor and saw that the final tests had passed and the new system was being switched over to operational status. He wondered what they might learn with this latest deployment. Whatever it was, the information would be available and used in the next release which was already well into development. Joe realized before calling it a day that while they had accomplished much in the past few years, there was still much more that needed to be done. Continuous improvement for transformation had turned out to be just that – continuous.

Appendix A WORKSHOP RESULTS

PURPOSE

The goal for the current phase of the Systems Engineering Transformation (SET) research task is to develop a 3-Year roadmap that will facilitate the transformation of the discipline of Systems Engineering. In order to achieve this goal, an investigation into the current state of the art of the Systems Engineering discipline in the literature, research community, and in the field of practice, is required. This workshop was planned with the vision of attaining a better understanding of the sponsor's needs and their environment; reviewing, refining, and prioritizing the research findings and focus areas; and finally, enhancing the communication between the sponsor and researchers to provide a solid foundation for future research activities.

TIME AND LOCATION

The workshop took place at the Stevens Institute of Technology facility in the Ronald Reagan Building, 1300 Pennsylvania Avenue NW, Washington DC, on Wednesday 27th and Thursday 28th January 2010. The workshop participants included researchers from the Systems Engineering Research Center, and personnel from the offices of the sponsor.

PRESENTATIONS AND DISCUSSIONS

Day 1 of the workshop began with an introduction from the sponsor, which provided a context for the problem, and their vision for the workshop. The SET principal investigator then provided a program overview for Systems Engineering transformation. These presentations provided a segue for an open discussion on how the sponsor views their challenges and where they see opportunity for research in this area. This discussion included a description of the sponsor's agile development process. This description detailed both the project and program levels, as well as the relationship between the two. By the end of the discussion an understanding was reached as to the sponsor's key challenges, the gaps in their current practices and the desired state they wish to achieve. Finally, an overview was conducted of the high-level research areas. The SET research group identified eight research areas, and Day 1 saw the presentation and discussion of four of these areas: Prioritization and Trade-Off Analysis, Concept Engineering, Architectural and Design Analysis, and Design Synthesis. On Day 2, the final four research areas were presented and discussed: Active System Characterization, Human-System Integration, Agile Process Engineering, and Modeling Environment Infrastructure.

The **Prioritization and Trade-Off Analysis** research area, coupled closely with Research Task 18, primarily focuses on the shortcomings of the current practices of prioritization and trade-off analysis, which are typically based on budgetary constraints.

This area suggests an alternative approach to prioritization and trade-off analysis that would not only be of use to the sponsor but also the Systems Engineering community at large. Instead of using budget as the primary constraint, this area suggests a value-based approach, which essentially defines success as achieved only if value is added to the success critical stakeholders. There are currently no tools to accommodate this new value-based prioritization and trade-off analysis, neither are there tools for related requirements conflict resolution and requirements change impact analysis. Therefore, the SET team hopes to leverage existing tools to develop a new toolset that will improve the quality of decision making and reduce the time that is needed to accomplish it.
The Concept Engineering research area, coupled closely with Research Task 3, focuses on the conceptualization phase of a project. The current practice requires either a document driven concept of operations or ad hoc techniques, which limit the developer and stakeholder's understanding of a proposed system. The SET team hopes to develop an interactive tool for use during the conceptual phase, which will allow stakeholders to produce a shared mental model of the proposed system, creating a higher degree of fidelity between what the customer says they want, what they understand they want and what the developers deliver. This tool will also produce a more cognitive concept development, an environment for rapid evaluation of alternative concepts, and a vehicle to perform system analysis.

This **Architectural and Design Analysis** research area is aimed at developing an architecture centric design approach with system adaptability as the primary design driver. This approach will utilize visualization and analysis tools to facilitate the decision making process throughout the architectural specification and design phases. These visualization tools also allow for analysis of the ripple effect commonly felt throughout the system when changes are made during these phases. This new approach, and accompanying tools, will allow for improved rapid development and fielding, as well as superior system evolvability and rapid exploration of design trade-offs.
The Design Reuse and Synthesis research area involves the use of technology to mine existing architectures, designs and test for patterns. Once mined, the SET team hopes to establish a common lexicon for the patterns, and build a repository to allow for their easy reuse. This area also examines the possible use of modeling tools such as SysML as an integrating tool for architecture and pattern repositories, to increase efficiency and reduce time and cost of the architecting and design phases. This area hopes to mitigate architecture and design risks, create a common language to enable widespread understanding of architecture and design, control system complexity through standardizing architecture, and allow for the capture and reuse of quality architecture and design concepts.

Active System Characterization provides understanding of how legacy/existing systems, which were typically developed with little or no documentation, actually behave in their deployed environments. This research will look at how existing systems can be instrumented, measured, and analyzed to provide the necessary fidelity to system models to aid in subsequent design efforts, and also in the validation of the conceptual

model on how the system is being used and how value is being created. With respect to monitoring, the sponsor highlighted that you can monitor for maintenance, performance improvement, and application enhancement. In each instance, there are different reasons for striving toward adaptive and real life feedback, and the sponsor wants to focus on an SE perspective/rationale, which includes thinking about lifecycle, security, etc. The SET team indicated that before and during the progression to an SOA is the prime time to address performance and other monitored issues... The monitoring issues addressed shouldn't be limited to the system under development, but also should consider the system used to develop the desired system.

Human-System Integration introduces the human element, both from an operations and a system design perspective, to system modeling. The human factor needs to be considered to ensure that the overall system (technology and human) is being optimized. Research is necessary in two areas. The first is the development of appropriate models for humans who interact with (operators/agents), and thus are elements of a system. The second area is an exploration of the capabilities and limitations of people who are developing the system with the key considerations including: developer-tool compatibility, end user compatibility relative to operational tasks, and end user coupling during system adaptation. A key concern for the sponsor is that tools with steep learning curves are not perceived to add value. Tools are needed that have user friendly and intuitive human interfaces, template driven interfaces for example. From the sponsor's perspective, we have an opportunity to re-engineer SE. In doing this, the automation you put in place changes the human interface, so how can we get the right level of interface to ensure the best payback? The SET team acknowledged this noting that the context is not a software development one, but instead is dynamic systems adaptation. The SET team concluded that automation could introduce bad habits; therefore we need to be careful about how we automate.

The **Agile Process Engineering** area is one in which there are no best practices. Instead, there are practices that have been shown to be beneficial in a given environment; however, these practices may be quite brittle when the application attributes change. To realize the benefits of interactive, model-centric engineering, the supporting practices must do more than just help one complete something in an agile way. The practices should not be static; they must be easily adaptable, taking full advantage of the evolving capabilities of tools and technology, within the environmental constraints. The SET team's vision is that we move away from top-down organizational level practices, and move towards a bottom-up approach, while being mindful that scalability is a challenge. Approaches to the problem should be "tailorable" and start at the user level. The sponsor's main concern with agility relates to the introduction of too much risk – how do you gauge this? The sponsor needs a process developed where you don't have a concern about implementing it, things just happen consistently and in an integrated fashion as the front-end works on a daily basis. The concept of patterns and reuse of ideas was brought up with a view to introducing feedback. However, both the subjective nature of how different people interpret information, and the fact that

decisions are often flawed due to the fact that change is occurring more rapidly than the response time of the communication cycle, needs to be considered.

An integrated **Modeling Environment Infrastructure** is required to provide effective and coherent communication between the users of the collaborative modeling and design environment. Without this capability, the users would have to fall back into sequential, isolated development, which mitigates many of the advantages of modeling. Effective model interoperability and management is needed, in addition to an understanding of how the environment can best present information to each user and increase design efficiency. With an integrated modeling environment a FULL lifecycle approach is needed that includes feedback throughout. In addition, once a system is retired, we need to consider the transition to new systems. The constraints that come with modeling tools are a concern for the sponsor. For example, if we consider early concept engineering, where you don't want to be too specific, use generally gravitates towards tools such as MS PowerPoint and Visio. This selection of tools is based on a perception that the other tools are too constraining – teams don't want to get into too much detail until later on in the lifecycle. The SET team highlighted that smarter tools don't have to be used to high levels of detail early on in the lifecycle – the appropriate elements can be selected for different stages of the lifecycle.

CONCLUSIONS

Following these presentation and discussion activities, the sponsor provided a presentation on how they viewed the discussed research areas as fitting into their context, and the SET team consolidated the workshop findings. Of the eight research areas presented, the sponsor found them all important and highlighted the following as their preferred focus areas: Agile Process Engineering, Concept Engineering, Active System Characterization, Architectural and Design Analysis, and Prioritization and Tradeoff Analysis. The sponsor did not highlight any new research areas. However, they would like a replacement for the existing Vee Model.

The SET team presented a vision, based on the workshop discussions, for the future state of SE as:

> "a transformed SE ability which consistently enables rapid, efficient delivery of continuously evolving capability without the pain that you currently feel while staying ahead of increasing system demands supported by technology and mission demands."

In order to achieve this vision, the transformed SE must: be seamlessly integrated into life cycle development, have the ability to be tailored to the environment's/projects specific needs, make the best use of human agents in development and operation, support asynchronous development, be automated where possible, be knowledge based and continually evolving, support analysis and decision making through automated data mining of artifacts, and focus on the interfaces.

REFERENCES

Section 1: Purpose

[Barnabe, 2009] Barnabe, Dennis, private communication, 2009.

[Cloutier, 2009] Cloutier, Robert et al, "Investigation of a Graphical CONOPS development Environment for Agile Systems Engineering." Final Technical Report (A013), 2009.

[Turner, 2009a] Turner, Richard et al, "Evaluation of Systems Engineering Methods, Processes and Tools on Department of Defense and Intelligence Community Programs." SERC Phase 1 Technical Report (A013), 2009.

[Turner, 2009b] Turner, Richard et al, "Evaluation of Systems Engineering Methods, Processes and Tools on Department of Defense and Intelligence Community Programs." Final Technical Report SERC-2009-TR004.

[Wilcox, 2010] Wilcox, J., "Embracing Change." CSER Presentation, 2010.

Section 2: Research Framework

[Boehm, 2003] Boehm, Barry., Turner, Richard., "Balancing Agility and Discipline", Addison Wesley, 2003.

[Fine, 1998] Fine, Charles H., Clockspeed: Winning Industry Control in the Age of Temporary Advantage, Basic Books, ISBN 0-7382-0153-7, 1998.

[INCOSE, 2004] Consensus of INCOSE Fellows, "What is Systems Engineering?", 2004, http://www.incose.org/practice/fellowsconsensus.aspx

Section 3: Current Practice

[Boehm, 1986] Boehm, B. W. (1986). "A Spiral Model of Software Development and Enhancement," ACM SIGSOFT, Software Engineering Notes, Vol. 11, No. 4, Aug. 1986, pp. 14-24.

[Boehm, 2003] Boehm, Barry, Turner, Richard, "Balancing Agility and Discipline", Addison Wesley, 2003.

[Bonvillian, 2004] Bonvillian, William B., "Meeting the new challenge to U.S. economic competitiveness: collaboration among government, industry, academia, and labor succeeded in the 1980s. That coalition must act again," Issues in Science and Technology, Vol 21, Issue 1, pg. 75, September 22, 2004.

[Burger, 2009] Burger, David, "Magnetic Core Memory", IEEE Global History Network, http://www.ieeeghn.org/wiki/index.php/Magnetic-Core_Memory, Retrieved on 2009-06-08.

[Carlini, 2009] Carlini, James. "Technology Tools for Rapid Capability Fielding," DDR&E Tools Briefing, December 15, 2009.

[Crisp, 2007] Crisp, Harry E. "INCOSE Systems Engineering Vision 2020." INCOSE-TP-2004-004-02, September 2007.

[Dee, 2009] Dee, Michael., INCOSE Model Based Systems Engineering Initiative (ppt), INCOSE MBSE Working Group, 2009.

[Eremenko, 2009] Eremenko, Paul. "META Novel Methods for Design & Verification of Complex Systems." DARPA Presentation, December 22, 2009.

[Estefan, 2008] Estefan, Jeff A., Survey of Model-Based Systems Engineering (MBSE) Methodologies, INCOSE-TD-2007-003-02, rev B, June 10, 2008.

[Fagen, 1978] Fagen, M., A History of Engineering and Science in the Bell System: National Service in War and Peace, 1925-1975. Bell Telephones Laboratories, New York. ISBN: 0932764002, 1978.

[Forsberg, 1991] Forsberg, Kevin; Mooz, Harold, The Relationship of System Engineering to the Project Cycle, Chattanooga, Tennessee: Proceedings of the National Council for Systems Engineering (NCOSE) Conference, pp. 57–65, October 1991.

[Forsberg, 2005] Forsberg, K., Mooz, H., Cotterman, H., Visualizing Project Management, 3rd edition, John Wiley and Sons, New York, NY, 2005. Pages 108-116, 242-248, 341-360, 2005.

[Friedenthal, 2007] Friedenthal, Sanford., Sampson, Mark., Griego, Regina., INCOSE Model Based Systems Engineering (MBSE) Initiative (pdf), INCOSE IW 2007, http://www.incose.org/enchantment/docs/07docs/07jul_4mbseroadmap.pdf

[Friedenthal, 2010a] Friedenthal, Sanford., Sampson, Mark., Griego, Regina., INCOSE Model Based Systems Engineering (MBSE) Workshop Outbrief Summary (ppt), INCOSE IW 2010.

[Friedenthal, 2010b] Friedenthal, Sanford., Sampson, Mark., Griego, Regina., INCOSE Model Based Systems Engineering (MBSE) Workshop Introduction (ppt), INCOSE IW 2010.

[Gelsinger, 2008] Gelsinger, P., "Moore's Law: We See No End in Sight," SYS-CON. 2008-05-01. http://java.sys-con.com/read/557154.htm. Retrieved on 2008-05-01, 2008.

[Hall, 1962] Hall, A. D., A Methodology for Systems Engineering. Van Nostrand Reinhold. ISBN 0442030460, 1962.

[INCOSE, 2007] INCOSE, Systems Engineering Vision, INCOSE-TP-2004-004-02, Version 2.03, 2007.

[Lemnios, 2009] Lemnios, Zachary., DDR&E Rapid Capability Fielding – Technology "Toolbox" Study – Terms of Reference, 2009.

[Lencioni 2002] Lencioni, Patrick. "Five Dysfunctions of a Team: A Leadership Fable." 2002.

[Mclellan, 2009] Mclellan, Paul., The Book that changed everything, Electronics Design, Strategy, News, February 11, 2009. http://www.edn.com/blog/920000692/post/1250039925.html]

[Moore, 1965] Moore, Gordon E., "Cramming more components onto integrated circuits", Electronics, Volume 38, Number 8, April 19, 1965.

[NASA, 2005] Simulation Based Acquisition Strategy (SBA) Implementation Strategy, 2005. http://aeronautics.arc.nasa.gov/assets/pdf/SBAStrategy_Final_w_signatures.pdf

[Nielsen, 1998] Nielsen, Jakob, "Nielsen's Law of Internet Bandwidth", 1998-04-05. http://www.useit.com/alertbox/980405.html, Retrieved on 2009-06-08.

[NRC, 2008] "Pre-Milestone A and Early-Phase Systems Engineering: A Retrospective Review and Benefits for Future Air Force Acquisition." Committee on Pre-Milestone A Systems Engineering, National Research Council. Washington, D.C.; The National Academies Press, 2008. http://www.nap.edu/catalog/12065.html.

[Royce, 1970] Royce, W. W., "Managing the Development of Large Software Systems: Concepts and Techniques," Proceedings WESCON, 1970, pp. 1-70.

[Schlager, 1956] Schlager, J., "Systems engineering: key to modern development". IRE Transactions EM-3: 64–66, July 1956.

[SSCI, 2009] "OUSD Study Workshop Technologies for Rapid Fielding Final Briefing." Systems and Software Consortium, Inc., October 16, 2009.

[Tehrani, 2000] Tehrani, Rich, "As We May Communicate", January 2000, http://www.tmcnet.com/articles/comsol/0100/0100pubout.htm, Retrieved on 2009-06-08.

[Turner, 2009a] Turner, Richard et al, "Evaluation of Systems Engineering Methods, Processes and Tools on Department of Defense and Intelligence Community Programs." SERC Phase 1 Technical Report (A013), 2009.

[Turner, 2009b] Turner, Richard et al, "Evaluation of Systems Engineering Methods, Processes and Tools on Department of Defense and Intelligence Community Programs." Final Technical Report SERC-2009-TR004, 2009.

[Turner, 2009c] Turner R, Pyster A, Pennotti M (2009), "Developing and Validating a Framework for Integrated Systems and Software Engineering", Stevens Institute of Technology, unpublished.

[Walter, 2005] Walter, C., "Kryder's Law". Scientific American (Verlagsgruppe Georg von Holtzbrinck GmbH), 2005-07-25. http://www.sciam.com/article.cfm?articleID=000B0C22-0805-12D8-BDFD83414B7F0000&ref=sciam&chanID=sa006. Retrieved on 2006-10-29.

[Watson, 2009] Ben Watson, Prasanta Bose, Rick Buskens, Trip Denton, Russell Kegley, Laura Pullum, and Daniel Waddington, A Research Agenda for the Systems and Software Initiative (SSI), Lockheed Martin Systems & Software Research Agenda - v2.2, January 2009.

[Wgsimon, 2008] http://en.wikipedia.org/wiki/Moore's_law, 2008. Retrieved on 2009-06-27.

Uncited References

Bar-Yam, Y. (2003). When Systems Engineering Fails-toward Complex Systems Engineering. IEEE International Conference on Systems, Man and Cybernetics. Volume: 2: p. 2021 - 2028.

Bohdan W. Oppenheim, B.W., Murman, E.M., Secor, D.A., Lean Enablers for Systems Engineering, Systems Engineering, Wiley Periodicals, November 2009

Dominguez, Jorge, The Curious Case of the CHAOS Report 2009, Retrieved 30th November 2009, from

http://www.projectsmart.co.uk/pdf/the-curious-case-of-the-chaos-report-2009.pdf

Finkelstein, A. (1993). Report of the Inquiry Into The London Ambulance Service.

GAO, Best practices: Increased focus on requirements and oversight needed to improve DOD's Acquisition environment and weapon system quality, GAO-08-294, General Accountability Office, Washington, DC, February 2008a.

GAO, Space acquisitions: Major space programs still at risk for cost and schedule increases, GAO-08-552T, General Accountability Office, Washington, DC, March 2008b.

GAO, Defense acquisitions: Assessments of selected weapon programs, GAO-09-326SP, General Accountability Office, Washington, DC, March 2009.

Oz, E. (1994). "When professional standards are lax: the CONFIRM failure and its lessons." Communications of the ACM Volume: 37(Issue: 10): p. 29 - 43.

Stengel, R., B. V. Voorst, et al. (1997). "An Overtaxed IRS." Retrieved 15th December 2009, from http://www.time.com/time/printout/0,8816,986145,00.html.

The Standish Group, The Standish Group Chaos Report - 1995. Retrieved 30th November 2009, from http://www.projectsmart.co.uk/docs/chaos-report.pdf OR www.standishgroup.com/sample_research/chaos_1994_1.php

The Standish Group, Unfinished Voyages: A Follow Up to the Chaos Report – 1995. Retrieved 30th November 2009, from http://www.standishgroup.com/sample_research/unfinished_voyages_1.php

The Standish Group, Chaos: A Recipe for Success - 1999. Retrieved 30th November 2009, from http://www.standishgroup.com/sample_research/index.php

The Standish Group, CHAOS Summary 2009. Retrieved 30th November 2009, from http://www.standishgroup.com/newsroom/chaos_2009.php

Section 4: Proposed Areas of Innovation

[Wymore, 2004] Wymore, A. Wayne, "Contributions to the Mathematical Foundations of Systems Science and Systems Engineering," Systems Movement: Autobiographical Retrospectives, The University of Arizona, Tucson, AZ, 2004.

Section 5: 3-Year Roadmap

Prioritization & Tradeoff Analysis

[Al-Said, 2003] Al-Said, M., "Identifying, Analyzing, and Avoiding Software Model Clashes", Ph.D. Dissertation, USC, 2003.

[Biffl, 2005] Biffl, S., Aurum, A., Boehm, B., Erdogmus, H., and Gruenbacher, P., (eds.), Value-Based Software Engineering, Springer Verlag, 2005.

[Boehm, 2000] Boehm, B., Port, D., and Al-Said, M., "Avoiding the Software Model-Clash Spiderweb," Computer, November 2000, pp. 120-122.

[Clements, 2002] Clements, P., Kazman, R., and Klein, M., Evaluating Software Architectures, Addison Wesley, 2002.

Concept Engineering

[Carlini, 2009] Carlini, J. "Technology Tools for Rapid Capability Fielding," DDR&E Tools Briefing, December 15, 2009.

[Cloutier, 2009] Cloutier, R., Mostashari, A., McComb, S., Deshmukh, A., Wade, J., Kennedy, D., Korfiatis, P., Investigation of a Graphical CONOPS Development Environment for Agile Systems Engineering, Final Technical Report SERC-2009-TR-003, October 2009.

Architecture and Design Analysis

[Blanchard, 2006] Blanchard, B.S. and Fabrycky, W.J. 2006. Systems engineering and analysis, 4th Ed. Pearson Prentice Hall, 2006.

[Browning, 2008] Browning, T.R. and Honour, E.C. 2008. Measuring the Life-cycle Value of Enduring Systems. Systems Engineering, Vol. 11, No. 3, 2008, pp 187-202.

[Crawley, 2004] Crawley, E., deWeck, O., Eppinger, S., Magee, C., Moses, J., Seering, W., Schindall, J., Wallace, D. and Whitney, D. 2004. The influence of architecture on engineering systems, MIT Eng Syst Symp, March 29–31, Cambridge, MA, 2004.

[Engel, 2008] Engel, A. and Browning, T.R. 2008. Designing systems for adaptability by means of architecture options. Systems Engineering, Vol. 11, No. 2, May 2008, pp 125-146.

[Fricke, 2005] Fricke, E. and Schulz A.P. 2005. Design for Changeability (DfC): Principles to enable changes in systems throughout their entire lifecycle. Systems Engineering, Vol. 8, No. 4, 2005 pp 342-359.

[Madni, 2008] Madni, A.M. "Architecture Follies: Common Misconceptions and Erroneous Assumptions," Fellows' Insight, INCOSE INSIGHT, Vol. 11, No. 1, pp. 33-34, January 2008.

[Madni, 2007] Madni, A.M. "Architecture Tradeoff Analysis: A Disciplined Approach to Balancing Quality Requirements," Ground System Architectures Workshop (GSAW), Architecture-Centric Evolution (ACE) of Software-Intensive Systems Presentation, March 27, 2007.

[Madni, 2010a] Madni, A.M. and Neill, C. Presentation on Architecture Design and Synthesis at SET RT-10 Workshop, January 2010.

[Ross, 2008] Ross, A.R, Rhodes, D.H., and Hastings, D.E. 2008. Defining changeability: Reconciling flexibility, adaptability, scalability, modifiability, and robustness for maintaining system lifecycle value. Systems Engineering, Vol. 11, No. 3, 2008 pp 246-262.

[Sangwan, 2007] Sangwan, R.S. and Neill, C.J. 2007. How business goals drive architectural design. Computer. Vol. 40, No. 8, August 2007. pp 101-103.

[Sangwan, 2008a] Sangwan, R.S., Neill, C.J., Bass, M, and El Houda, Z. 2008a. Integrating software architecture-centric methods into object-oriented analysis and design, Journal or Systems and Software. Vol. 81, Iss. 5, May 2008, Pages 727-746.

[Sangwan, 2008b] Sangwan, R.S., Li-Ping Lin, and Neill, C.J. 2008b. Structural complexity in architecture-centric software evolution, Computer, Vol. 41, No. 10, October 2008, pp. 99-102.

Design and Test Reuse and Synthesis

[Cloutier, 2006] Cloutier, R., Applicability of Patterns to Architecting Complex Systems, Doctoral Dissertation, Stevens Institute of Technology, Hoboken, NJ June 2006.

[Cloutier, 2007] Cloutier, R., Verma, D., Applying the Concept of Patterns to Systems Architecture, Systems Engineering Journal, Vol. 10, No. 2: 138-154, 2007.

[Cloutier, 2010] Cloutier, R., Muller, G., Verma, D., Nilchiani, R., Hole,. E., Bone, M., The Concept of Reference Architectures, INCOSE SE Journal, Volume 13 Issue Number 1, ISSN: 1098-1241, p. 14-27, January 2010.

Active System Characterization

[Helft, 2009] Helft, Miguel, New York Times, May 9, 2009.

[Hoffmann, 2007] Hoffmann, G.A.; Trivedi, K.S.; Malek, M.; , "A Best Practice Guide to Resource Forecasting for Computing Systems," Reliability, IEEE Transactions on , vol.56, no.4, pp.615-628, Dec. 2007.

[Karacali, 2007] Karacali, B.; Rao, B.; , "Network Instrumentation for End-to-End Measurements," *Integrated Network Management, 2007. IM '07. 10th IFIP/IEEE International Symposium on* , vol., no., pp.314-322, May 21 2007-Yearly 25 2007.

[Lu, 2006] Lu, C.; Lu, Y.; Abdelzaher, T.F.; Stankovic, J.A.; Son, S.H.; , "Feedback Control Architecture and Design Methodology for Service Delay Guarantees in Web Servers,"

Parallel and Distributed Systems, IEEE Transactions on , vol.17, no.9, pp.1014-1027, Sept. 2006.

[Pinzger, 2008] Pinzger, M.; , "Automated Web Performance Analysis," *Automated Software Engineering, 2008. ASE 2008. 23rd IEEE/ACM International Conference on* , vol., no., pp.513-516, 15-19 Sept. 2008.

[Wade, 2010] Wade, J.P. Presentation of Active System Characterization, SET RT-10 Workshop, January 2010.

[Wang, 2009] Wang, Kaiyu; Tian, Naishuo; , "Performance Evaluation of J2EE Web Applications with Queueing Networks," *Information Technology and Computer Science, 2009. ITCS 2009. International Conference on* , vol.1, no., pp.437-440, 25-26 July 2009.

Human-System Integration

[Madni, 2005] Madni, A.M., Sage, A. and Madni, C.C. Infusion of cognitive engineering into systems engineering processes and practices, Proc. of the 2005 IEEE International Conference on Systems, Man, and Cybernetics, October 10-12, 2005, Hawaii.

[Madni, 2008] Madni, A.M. Integrating human factors, software and systems engineering: Challenges and opportunities, Proc of the 23rd International Forum on COCOMO and Systems/Software Cost Modeling and ICM Workshop 3, Davidson Conference Center, University of Southern California, October 27-30, 2008.

[Madni, 2010b] Madni, A.M. "Integrating Humans with Software and Systems: Technical Challenges and a Research Agenda," INCOSE Journal of Systems Engineering, Vol. 13, No. 3, 2010.

[Madni, 2010c] Madni, A.M. SAE 549 Course, Lecture 6, "Human-System Integration: Implications for Systems Architecting and Engineering," University of Southern California, Spring Semester, 2010.

[Madni, 2010d] Madni, A.M. and Neill, C. Presentation of Human-Systems Integration, SET RT-10 Workshop, January 2010.

Uncited References

Booher H.R., Beaton, R., and Greene, F. Human Systems Integration, in A. Sage and W.B. Rouse (Editors), Handbook of Systems Engineering and Modeling, John Wiley and Sons, Hoboken, NJ, 2009, pp. 1319-1356

Defense Science Board, Defense Science Board Task Force on Patriot System Performance, Report Summary DTIC No. ADA435837, January 2005.

Department of Defense Handbook, Human Engineering Program Process and Procedures, MIL-HDBK-46855, January 31, 1996.

Madni, A.M. HUMANE: A knowledge-based simulation environment for human-machine function allocation, Proc of IEEE National Aerospace & Electronics Conference, Dayton, Ohio, May, 1988c.

Pew, R.W. and Mavor, A.S. Human–system integration in the system development process: A New Look, National Academy Press, 2007.

Sheridan, T.B. Humans and automation: System design and research issues, John Wiley Sons, 2002.

Agile Process Engineering

[Cass, 2000] Cass, A., Lerner, B., McCall, E., Osterweil, L., Sutton, S., and Wise, A., "Little-JIL/Juliette: A Process Definition Language and Interpreter," 22nd International Conference on Software Engineering (ICSE 2000), Limerick, Ireland, pp. 754-757, June 2000. (UM-CS-2000-066).

[Olbrich, 2009] Olbrich, S., Cruzes, D., Basili, V., and Zazworka, N., "The Evolution of Impact of Code Smells: A Case Study of Two Open Source Systems," Proceedings of the 3rd International Symposium on Empirical Software Engineering and Measurement (ESEM 2009), Lake Buena Vista, Orlando, Florida, October 2009.

[Turner, 2009b] Turner, Richard et al, "Evaluation of Systems Engineering Methods, Processes and Tools on Department of Defense and Intelligence Community Programs." Final Technical Report SERC-2009-TR004.

[Wise, 2006] Wise, A., "Little-JIL 1.5 Language," Report Department of Computer Science, University of Massachusetts, Amherst, MA 01003, October 2006. (UM-CS-2006-51).

[Zazworka, 2009] Zazworka, N., Basili, V., and Shull, F., "Tool Supported Detection and Judgment of Nonconformance in Process Execution," Proc. International Symposium on Empirical Software Engineering and Measurement (ESEM). Lake Buena Vista, FL, October 2009.

Modeling Environment Infrastructure

Uncited References

Batory, D., Johnson, C., MacDonald, B., and von Heeder, D., Achieving Extensibility Through Product-Lines and Domain-Specific Languages: A Case Study. ACM Transactions on Software Engineering and Methodology (TOSEM), Vol. 11, No. 2, April 2002, pp. 191-214.

Dagli, C.H. and Kilicay-Ergin, N., "Chapter 4: System of Systems Architecting", in System of Systems Engineering, M. Jamshidi (editor), Wiley & Sons Inc., 2009, p. 77-101.

Engel, A., and Browning, T.R., Designing systems for adaptability by means of architecture options. Systems Engineering, Vol. 11, No. 2, May 2008, pp 125-146.

Fricke, E., and Schulz, A.P., Design for Changeability (DfC): Principles to enable changes in systems throughout their entire lifecycle. Systems Engineering, Vol. 8, No. 4, December 2005, pp 342-359.

Gannon, T., Madnick, S., Moulton, A., Siegel, M., Sabbouh, M., Zhu, H., Semantic Information Integration in the Large: Adaptability, Extensibility, and Scalability of the Context Mediation Approach. MIT Sloan School of Management Working Paper 4541-05, May 2005.

Nilchiani, R., and Hastings, D.E., Measuring the value of flexibility in space systems: A six-element framework. Systems Engineering, Vol. 10, No. 1, January 2007, pp 26-44.

Rao, M., Ramakrishnan, S., and Dagli,C.H., "Modeling and Simulation of Net Centric System Of Systems Using Systems Modeling Language And Colored Petrinets – A Demonstration Using The Global Earth Observation System Of Systems" Systems Engineering Journal, Volume 11, Issue 3, 2008, Pages 203-220.

Roberts, D., and Johnson, R., Evolving Frameworks: A Pattern Language for Developing Object-Oriented Frameworks, Proc. Third Conference on Pattern Languages and Programming (PLoP 96), Allerton Park, IL, Sept. 4-6, 1996.

Ross, A.R., Rhodes, D.H. and Hastings, D.E., Defining changeability: Reconciling flexibility, adaptability, scalability, modifiability, and robustness for maintaining system lifecycle value. Systems Engineering, Vol. 11, No. 3, September 2008, pp 246-262.