2014-12-14

-ilities Tradespace and Affordability Project Phase 3

Jacques, David; Columbi, John; Ryan, Erin; Ender, Tommer; Peak, Russell; Sitterle, Valerie

Systems Engineering Research Center (SERC)

-ilities Tradespace and Affordability Project – Phase 3

Technical Report SERC-2014-TR-039-3

December 31, 2014

Principal Investigator: Dr. Barry Boehm, University of Southern California

Research Team:

Air Force Institute of Technology

Georgia Institute of Technology

Massachusetts Institute of Technology

Naval Postgraduate School

Pennsylvania State University

University of Southern California

University of Virginia

Wayne State University

This material is based upon work supported, in whole or in part, by the U.S. Department of Defense through the Systems Engineering Research Center (SERC) under Contracts H98230-08-D-0171 (RT 31, RT 46) and HQ0034-13-D-0004 (Task Order 013, RT 113). SERC is a federally funded University Affiliated Research Center managed by Stevens Institute of Technology.

EXECUTIVE SUMMARY

One of the key elements of the SERC's research strategy is transforming the practice of systems engineering and associated management practices – "SE and Management Transformation (SEMT)." The Grand Challenge goal for SEMT is to transform the DoD community's current systems engineering and management methods, processes, and tools (MPTs) and practices away from sequential, single stovepipe system, hardware-first, document-driven, point-solution, acquisition-oriented approaches; and toward concurrent, portfolio and enterprise-oriented, hardware-software-human engineered, model-driven, set-based, full life cycle approaches.

These will enable much more rapid, concurrent, flexible, scalable definition and analysis of the increasingly complex, dynamic, multi-stakeholder, cyber-physical-human DoD systems of the future. Four elements of the research strategy for SE Transformation are the following:

1. Make Smart Trades Quickly: Develop MPTs to enable stakeholders to be able to understand and visualize the tradespace and make smart decisions quickly that take into account how the many characteristics and functions of systems impact each other
2. Rapidly Conceive of Systems: Develop MPTs that allow multi-discipline stakeholders to quickly develop alternative system concepts and evaluate them for their effectiveness and practicality
3. Balance Agility, Assurance, and Affordability: Develop SE MPTs that work with high assurance in the face of high uncertainty and rapid change in mission, requirements, technology, and other factors to allow systems to be rapidly and cost-effectively acquired and responsive to both anticipated and unanticipated changes in the field
4. Align with Engineered Resilient Systems (ERS): Align research to leverage DoD's ERS strategic research initiative and contribute to it; e.g., ERS efforts to define new approaches to tradespace analysis.

"Systems" covers the full range of DoD systems of interest from components such as sensors and effectors to full systems that are part of net-centric systems of systems and enterprises. "Effectiveness" covers the full range of needed system quality attributes or ilities, such as reliability, availability, maintainability, safety, security, performance, usability, scalability, interoperability, speed, versatility, flexibility, and adaptability, along with composite attributes such as resilience, sustainability, and suitability or mission effectiveness. "Cost" covers the full range of needed resources, including present and future dollars, calendar time, critical skills, and critical material resources.

The primary focus of RT-113, ilities Tradespace and Affordability Project (iTAP) is on strategy 3, although its capabilities also support strategies 1, 2, and 4. It particularly focuses on the tradespace among a system's Ilities, also called non-functional requirements or system quality attributes. The ilities differ from functional requirements in that they are systemwide

properties that specify *how well* the system should perform, as compared to functions that specify *what* the system should perform. Adding a functional requirement to a system's specification tends to have an incremental, additive effect on the system's cost and schedule. Adding an ility requirement to a system's specification tends to have a systemwide, multiplicative effect on the system's cost and schedule. Also, ilities are harder to specify and evaluate, as their values vary with variations in the system's environment and operational scenarios.

Further, the satisfaction of their specifications is much harder to verify than placing an X or a URL address in a functional traceability matrix, as the verification involves up to the entire set of system functions. It also requires considerable effort in analysis across a range of environments and operational scenarios. As a result, it is not surprising that problems in satisfying ility requirements are underaddressed in early phases and are the source of many subsequent DoD acquisition program cost and schedule overruns. Also, with some exceptions such as pure physical systems and pure software systems, there is little technology in the form of scalable methods, processes, and tools (MPTs) for evaluating the satisfaction of multiple-ility requirements and their associated tradespaces for complex cyber-physical-human systems.

The increasingly critical DoD need for such capabilities has been identified in several recent studies and initiatives such as the AFRL "Technology Horizons" report (Dahm, 2010), the National Research Council's "Critical Code" Report (NRC, 2010), the SERC "Systems 2020" Report (SERC, 2010), the "Manual for the Operation of the Joint Capabilities Integration and Development System" (JROC, 2012), and the DoD "Engineered Resilient Systems (ERS) Roadmap" (Holland, 2012). The particular need for Affordability has been emphasized in several USD(AT&L) and DepSecDef "Better Buying Power" memoranda BBP 1.0 and 2.0 (Carter et al., 2010-2013) and the recent BBP 3.0 White Paper (Kendall, 2014).

iTAP Ongoing Contributions in Support of Better Buying Power 3.0

Here is a summary of the current and projected iTAP contributions in support of the objectives of Better Buying Power 3.0, relative to BBP 3.0 White Paper citations in italics. Details of the contributions are provided in the later section of the report.

Continue to set and enforce affordability constraints. Strengthen and expand "should cost" as an important tool for cost management. Building on previous results in RT-6 (Air Force Software Cost Modeling) and RT-18 (Valuing Flexibility), iTAP Phase 3 developed a framework and initial quantitative results for tradespace analysis of acquisition and total ownership should-costs vs. schedule, functionality, and reliablility, and an overall framework and initial population of sources of synergy and conflict among cost, schedule, and other system quality attributes. Future plans include upgrading the models to reflect emerging trends such as flexibility to accommodate increasingly rapid change, adaptability to accommodate autonomy and smart devices, usability to accommodate social networking and labor force evolution, and interoperability to accommodate increasingly interconnected systems of systems.

Employ appropriate contract types, but increase the use of incentive-type contracts. "formulaic incentives" show a high correlation with better cost and schedule performance. As above, Phase 3 has been developing cost-schedule-performance tradespace models to provide stronger formulas on which to define formulaic incentives.

Increase the use of performance-based logistics (PBL). As documented in Appendix A, Enclosure A of the 19 Jan 2012 JCIDS Manual, a survey of combat commenders on critical logistics attributes identified the key quality attributes for logistics performance as Responsiveness, Sustainability, Attainability, Flexibility, Survivability, and Economy, along with several other attributes. Phase 3 is developing an ontology of such attributes to provide a stronger basis for defining logistics performance.

Use Modular Open Systems Architecture to stimulate innovation. SERC has been collaborating with TARDEC and NAVSEA in Phase 3 to define sound properties for set-based design as a stronger way to achieve mission-supportive open systems architectures, and to define related set-based design processes.

Provide clear "best value" definitions so that industry can propose and DoD can choose wisely: providing industry with information on the value, in monetary terms, of higher levels of performance than minimally acceptable or threshold levels. As above, Phase 3 has been developing cost-schedule-performance tradespace models to provide stronger formulas on which to define the value of higher levels of performance.

Improve our leaders' ability to understand and mitigate technical risk: to minimize the likelihood of program disruption and to maximize the probability of fielding the desired product within reasonable time and cost. Phase 3 research has been extending, applying, and refining an iterative Epoch-Era approach to address sources of uncertainty and risk. A related SERC project, Quantitative Risk, is developing improved methods for identifying and quantifying leading risk indicators. The set-based design approach goes beyond design for acquisition, to use areas of requirements uncertainty as foci for architecting sets of requirements likely to needed post-acquisition.

PHASE 1 OBJECTIVES, APPROACH, AND RESULTS

The major objectives of the initial 5-month Phase 1 activity were to lay strong foundations for ITAP Phase 2, including knowledge of Department of Defense (DoD) ility priorities; foundations and frameworks for ITAP analysis; extension and tailoring of existing ITAP methods, processes, and tools (MPTs); and exploration of candidate Phase 2 pilot organizations for ITAP MPTs.

Four activities were pursued in achieving these objectives:

1. Ility Definitions and Relationships. Phase 1 included a discovery activity to identify and analyze DoD and other ility definitions and relationships, and to propose a draft set of DoD-oriented working definitions and relationships for the project.

2. iTAP Foundations and Frameworks. This effort helped to build iTAP foundations by elaborating key frameworks (architecture-based, change-based, process-based, means-ends based, value-based), anticipating further subsequent elaboration via community efforts.

3. Ility-Oriented tool demos and extension plans. This effort created initial demonstration capabilities from strong existing ITA analysis toolsets and explored piloting by user organizations in the DoD Services.

4. Program management and community building. This effort included coordinating efforts with complementary initiatives in the DoD ERS, and counterpart working groups in the International Council for Systems Engineering (INCOSE), the Military Operations Research Society (MORS), and the National Defense industry Association (NDIA).

The Phase 1 results for activities 1 and 2 included initial top-level sets of views relevant to ilities tradespace and affordability analysis that provided an initial common framework for reasoning about ilities, similar in intent to the various views provided by SysML for product architectures and DoDAF for operational and architectural views. The views included definitions, stakeholder value-based and change-oriented views, views of ility synergies and conflicts resulting from ility achievement strategies, and a representation scheme and support system for view construction and analysis.

Phase 1 also determined that strong tradespace capabilities were being developed for the tradespace analysis of physical systems. However, based on sources such as the JCIDS survey of combat commanders' tradespace needs, it found that major gaps existed between commanders' ility tradespace needs and available capabilities for current and future cyber-physical-human systems. The SERC also characterized the benefits and limitations of using existing tools to address ility tradespace issues, via collaboration with other leading organizations in the DoD ERS tradespace area, such as the Army Engineer Research and Development Center (ERDC) and TARDEC organizations, NAVSEA, the USAF Space and Missile Systems Command; DoD FFRDCs such as Aerospace, Mitre, and the Software Engineering Institute; and Air Force and Navy participants via the SERC Service academies AFIT and NPS.

PHASE 2 OBJECTIVES, APPROACH, AND RESULTS

As a result, the focus of Phase 2 was to strengthen the conceptual frameworks underlying ilities tradespace and affordability analysis, and to apply the methods and tools identified and extended in Phase 1 on problems relevant to DoD, using the information available from development of a large weapon systems and large automated information systems. The SERC worked with system developers directly and via participation and leadership in Government and industry working groups in such organizations as INCOSE, NDIA, and the Army-led Practical

Systems and Software Measurement organization, to gain a deeper shared understanding of the strengths and limitations of the tradespace tools and methods developed under Phase 1 and elsewhere.

Task 1: ITAP Foundations and Frameworks. Phase 2 activities expanded the set of ilities represented in the tradespace, organized them into a more orthogonal value-based, means-ends hierarchy, obtained initial results in identifying and quantifying the synergies and conflicts resulting from strategies to optimize individual ilities, and developed prototype tools for representing and applying the results.

Task 2. iTAP Methods and Tools Piloting and Refinement. The Ility-oriented tool demos performed in Phase 1 also led to Phase 2 interactions with DoD organizations, particularly TARDEC and NAVSEA, interested in their applicability in enhancing their systems engineering capabilities. These interactions led to refinements of existing methods and tools to address set-based vs. point design of ground vehicles and ships, and on extensions from physical systems to cyber-physical-human systems and to affordability analysis. Further interactions leading to piloting engagements included AFIT's use of the CEVLCC life cycle cost model and related T-X Training System Tradespace Analyses. The exploratory-use program involved advanced pilot training aircraft, simulators and course instructional elements. Its user organizations are the Air Force Life Cycle Management Center and the Air Education and Training Command.

Task 3. Next-Generation, Full-Coverage Cost Estimation Model Ensembles. A third area of engagement starting from exploratory discussions in Phase 1 was a new task to develop Next-Generation, Full-Coverage Cost Estimation Model Ensembles, initially for the space domain, based on discussions and initial support from the USAF Space and Missile Systems Center (SMC). Phase 2 work on this topic involved several meetings with SMC and the Aerospace Corp. with USC and NPS to set context and initial priorities. These included addressal of future cost estimation challenges identified in the SERC RT-6 Software Cost Estimation Metrics Manual developed for the Air Force Cost Analysis Agency, and prioritization of research efforts based on strength of DoD needs and availability of DoD-relevant data. Exploratory activities were pursued with respect to a scoping of full-coverage of space system flight, ground, and launch systems; hardware, software and labor costs; and system definition, development, operations, and support costs

Phase 3 Objectives, Approach, and Results Overview

Details of each task's past results, Phase 3 results, and future plans are in the detailed reports of each organization participating in the task.

Task 1: ITAP Foundations and Frameworks. Phase 2 refined an ilities semantic basis for change-related ilities. and developed prototype tools for formal analysis of the results. Phase 3 extended the ilities semantic basis for change-related ilities, resulting from continuing literature review of ilities, collaborative work on formalization of the basis, and experience in applying the basis in historical cases. Progress and adjustments to the basis have been made as a result of

feedback from other academic researchers, and specifically in MIT- UVa collaboration in their efforts on formalization and development of a REST (representational state transfer) web-based service implementation. This has resulted in an expanded and more explicit representation for the semantic basis, as well as motivating the need to create a translation layer for practical use of the basis. Phase 3 also refined the ility definitions, reviewed existing ility definition standards, developed an initial ilities ontology to address the current shortfalls in existing ility-related standard and guidelines. These generally do not address the reality that the ilities have multiple definitions varying by domain, and multiple values varying by system state, processes, and relations with other ility levels. Phase 3 also expanded the initial 4x4 synergies and conflicts matrix into a full 7x7 inter-ility-class synergies and conflicts matrix, and 7 smaller intra-ility-class synergies and conflicts matrices. Detailed results are in the USC, MIT, and UVa sections.

Task 2: Ility-Oriented tool demos and extension plans. Phase 2 effort created initial demonstration capabilities from strong existing ITA analysis toolsets and explored piloting by user organizations, via collaboration with other leading organizations in the DoD ERS tradespace area. Phase 3 broadened and deepened these initial contacts, including with such organizations as the as the Army Engineer Research and Development Center (ERDC) and TARDEC organizations, NAVSEA, the USAF Space and Missile Systems Command; DoD FFRDCs such as Aerospace, Mitre, and the Software Engineering Institute; and Air Force and Navy participants via the SERC Service academies AFIT and NPS. In particular, we have advanced our coordination with the ERS NAVSEA group, working with them to define the specific tradespace approaches and priorities for enhanced set-based design for ERS, that will complement and extend the tool and procedures they have been using. We anticipate access to a public-release copy of the ship design option datasets developed as part of their ERS set-based-design experiment. We initiated discussions with NAVAIR and with the DARPA/TARDEC Ground Experimental Vehicle – Technology (GXV-T) project regarding piloting iTAP MPT. We have begun to develop a network model of the interdependencies within and between the TARDEC Ground System Architecture Framework and Performance Specification Framework, and coordination with TARDEC in this research. TARDEC is actively engaged as a partner for co-development, piloting and transition into use. Detailed results are in the Wayne State, GTRI, Penn State, AFIT, NPS, and USC sections.

Task 3: Next-Generation, Full-Coverage Cost Estimation Model Ensembles. Based on the exploratory needs and data assessments in Phase 2, a Phase 3 workshop including Air Force, Navy, aerospace industry, and SERC researchers concluded that there were strong needs for better estimation of operations and support costs, but that the data available lacked adequate cost driver information, except in in the software area. The workshop recommended that the most promising initial areas to pursue would be for software development, systems engineering, and the use of systems engineering cost drivers to improve estimation of system development costs. Further research and workshops have identified further sources of data and some shortfalls in current models in these areas, and have developed requirements and draft frameworks for the next-generation models. These will be used in Phase 4 to develop and

calibrate prototype models for systems and software engineering cost estimation models, and to pursue research in the use of the systems engineering cost model to better estimate system development costs. Detailed results are in the USC, NPS, and AFIT sections.

Complementary Funding

During Phase 3, RT-113 contributed significantly to the SERC objective of obtaining complementary funding to its SERC core support. The Army Engineer Research and Development Center sponsored MIT research in SE knowledge capture, and provided over $1million to GTRI for tradespace tools research. GTRI also obtained significant from the Navy for tradespace tools research. USC and UVa obtained complementary funding from NSF to extend the ilities tradespace theory and foundations. AFIT and NPS provided complementary support of their SERC research via participation of their faulty and students in their SERC research.

REFERENCES

1. (Carter et al., 2010-2013) Carter, A., et al., Better Buying Power Memoranda, http://bbp.dau.mil/references.html.

2. Dahm, 2010) Dahm, W., Technology Horizons, AF/ST-TR 10-01-PR, 15 May 2010.

3. (Holland, 2012) Holland, J. "ERS Overview," Proceedings, NDIA SE Conference, October 2012.

4. (JROC, 2011) Joint Requirements oversight Council, "Manual for the Operation of the Joint Capabilities Integration and Development System," Updated 31 January 2011.

5. (Kendall, 2014) Kendall, F., "Better Buying Power 3.0 White Paper," OUSD(AT&L), 19 September 2014.

6. (NRC, 2010) Scherlis, W. et al., Critical Code: Software Producibility for Defense NAS Press, 2011.

7. (SERC, 2010) B. Boehm, J. Bayuk, A. Desmukh, R. Graybill, J. Lane, A. Levin, A. Madni, M. McGrath, A. Pyster, S. Tarchalski, R. Turner, and J. Wade, Systems 2020 Strategic Initiative, Final Technical Report SERC-2010-TR-009, August 29, 2010.

RESEARCH TEAM

Air Force Institute of Technology	Dr. David Jacques, Co-PI Dr. John Columbi Dr. Erin Ryan
Georgia Institute of Technology	Dr. Tommer Ender, Co-PI Dr. Russell Peak Dr. Valerie Sitterle Dr. Michael Curry Dr. Dane Freeman
Massachusetts Institute of Technology	Dr. Donna Rhodes Dr. Adam Ross, Co-PI
Naval Postgraduate School	Dr. Ray Madachy, Co-PI
Pennsylvania State University	Dr. Michael Yukish, Co-PI
University of Southern California	Dr. Barry Boehm, PI Dr. JoAnn Lane, Co-PI Dr. Bradford Clark Mr. Nupul Kukreja Mr. Jim Alstead
University of Virginia	Dr. Kevin Sullivan, Co-PI Ms. Xi Wang
Wayne State University	Dr. Walter Bryzik Dr. Gary Witus, Co-PI

TABLE OF CONTENTS

FIGURES AND TABLES

iTAP Results by Organization: Past Results, Phase 3 Results, and Future Plans

1 University of Southern California

1.1 Past Results: Phases 1 and 2 Results

Phase 1 focused primarily on building iTAP foundations by elaborating draft frameworks (process-based, architecture-based, means-ends based, value-based), and socializing these via team workshops anticipating further subsequent elaboration via community efforts. Phase 2 refined the draft frameworks and ility definitions, and prototyped an early version of an ilities Synergies and Conflicts matrix for guidance to systems engineers on the side effects on other ilities when increasing a given ility's level. It began an effort to define a next-generation, full coverage family of cost models, initially for the space community based on interest and support from USF/SMC. Phase 2 activities also included cost modeling support of SERC initiatives with NAVSEA and TARDEC.

1.2 Phase 3 Results

1.2.1 Foundations: Initial ilities Ontology

Based on reviews in Phase 3 of existing ility standards such as ISO/IEC 25010, whose implicit ontology definitions do not cover sources of variation in ility values, USC developed in Phase 3 an initial ilities ontology structure that enables specification and reasoning about these sources of ility value variation. Its structure is based on the IDEF5 Kind-Individual-States-Processes-Relations structure, but renames Kinds as Classes, and uses the means-ends relationship as the basis for the class hierarchy.

This ontology structure also simplified the initial Phase 3 top-level System-Ends classes from 6 to 4 by making Composability/Interoperability a means to the end of Mission Effectiveness, and by combining the overlapping Protection and Robustness end classes into a single Dependability end class. The States, Processes, and Relations elements of the ontology capture the primary sources of variation in ility values that are generally left undefined in traditional ility definitions. States capture ility value variations via variations in the system's internal state and the state of the external environment. Processes capture ility value variations via variations in the system's operational concept and execution processes. Relations capture ility value variations via variations in the system's synergies and conflicts with other ilities.

1.2.1.1 Initial Ontology Description

Introduction

The nature of the ilities is best understood with respect to their synonym of non-functional

requirements. Functional requirements specify the functions that the system shall perform, or basically what the system should do. Their effect on system cost is basically additive; adding a function adds the cost to develop it, plus another fraction for its contribution to system integration and test. A non-functional requirement specifies how well the system should perform its functions. Unlike functional requirements, its effect on system cost is system wide and multiplicative. In one example, changing the response time for a complex system with 2000 pages of functional requirements from 1 second to 4 seconds reduced the cost from $100 million to $30 million, in this case by avoiding the need for an expensive custom solution, and by showing that a 4-second response time was acceptable for 90% of the transactions and achievable via commercial technology[1]. However, functional requirements are much more emphasized in system acquisitions, via such practices as initial system functional review milestones, function-oriented system definition diagrams, functional work breakdown structures and associated earned value milestones, and data item descriptions, in which evidence of ility satisfaction is relegated to optional appendices, which are easiest to drop under budget and schedule pressures.

Concern with the ilities has considerable historical precedent. The writings of Vitruvius in the Roman era include considerations of soundness, utility, attractiveness, healthfulness, durability, malleability, proportionality, symmetry, and defendability[2]. Subsequent landmarks in the evolution of knowledge about the ilities include contributions to physical, economic, human, software, and combined ilities.[3.4.5.6.7.8.9.10]

The software standards community, particularly ISO (the International Organization for Standardization) and IEC (the International Electrotechnical Commission), has been active in developing standards for quality attributes, initially for software with the 2001-2004 ISO/IEC 9126 series[11] and the 2005-2008 ISO/IEC 25000-25030[12] series, and subsequently for systems and software quality models with the 2011 ISO/IEC 25010 standard[13]. Significant progress has also been made in the definition and analysis of physical-system quality attributes such as within the DoD Engineered Resilient Systems (ERS) initative[14], the MIT series of change-oriented quality attribute research results[15,16], and the Georgia Tech contributions to ility tradespace analysis[17].

In July 2012, the DoD ERS program and the DoD Systems Engineering Research Center (SERC) held a pair of workshops on systems tradespace exploration and analysis. Their key conclusions were that improved ility tradespace exploration and analysis capabilities were needed to avoid DoD system shortfalls and overruns; that affordability was an increasing concern for DoD systems; and that ERS strengths in physical systems tradespace analysis could be complemented by ongoing SERC research strengths in cyber-physical-human systems tradespace and affordability exploration and analysis, as reflected in references [1,7, 15, 16, 17, and 18]. This led to the establishment of a multi-year, multi-university SERC ilities Tradespace and Affordability Program (iTAP) initiative to provide, pilot, and evolve stronger foundations methods, process, tools, and models for ilities tradespace and affordability exploration and analysis. The SERC universities involved in iTAP are Air Force Institute of Technology, Georgia Institute of Technology, Massachusetts Institute of Technology, Naval Postgraduate School,

Pennsylvania State University, University of Southern California, University of Virginia and Wayne State University.

The remainder of the paper focuses on the need for an Initial Definition of an Ilities Ontology (IDIO), as a component of the iTAP foundations research. It begins with an assessment of current ility definitions and their shortfalls with respect to the nature of the ilities. It then discusses the nature of ontologies and the choice of ontology structure for the IDIO. It then elaborates on the content of the IDIO, with examples of its use; and concludes with an assessment of outstanding issues and needs for further research.

1.2.2 The Need for a System ilities Ontology

The primary need for a system ilities ontology is to enable more effective collaboration in system definition, development, and evolution across system stakeholders from different disciplines. A good example of the current Tower of Babel is in the diversity of definitions of the ility Resilience in Wikipedia[19]. These include differing definitions between and within the domains of Ecology, Energy Development, Engineering and Construction, Network, Organizational, Psychological, and Soil. Within the Ecology and Society domain, an additional set of definitions includes Original-ecological, Extended-ecological (several definitions), Systemic-heuristic, Operational, Sociological, Ecological-economic, Social-ecological, Metaphorical, and Sustainabilty-related[20].

The differences in these definitions are non-trivial. For example, the variants in resilience outcomes include Returning to original state; Restoring or improving original state; Maintaining same relationships among state variables; Maintaining desired services; Maintaining an acceptable level of service; Retaining essentially the same function, structure, and feedbacks; Absorbing disturbances; Coping with disturbances; Self-organizing; Learning and adaptation; and Creating lasting value. Trying to get interdisciplinary teams to create consistently resilient systems across such a variety of implicit understandings presents a formidable challenge.

It does not help that most sources of ility definitions limit them to a single hopefully one-size-fits-all definition of each ility. This is particularly the case for the ISO/IEC 25010 Systems and Software Quality Requirements and Evaluation (SQuaRE) Standard[13]. As a representative example, it defines Reliability as the, “degree to which a system, product, or component performs specified functions under specified conditions for a specified period of time.” As a standard, this is supposed to hold for any definition of “specified functions” and “specified conditions.” However, for agile methods, in which “specified functions and conditions” are sunny-day stories or use cases, a system will be judged to be reliable if it satisfies the specified sunny-day conditions, but fails on the rainy-day conditions. Further, the definition only focuses on performing functions and not on satisfying ility levels.

Another difficulty with one-size-fits-all definitions is that they do not accommodate multiple stakeholders with multiple value propositions. For example, if a system specification defines reliability as liveness to satisfy one stakeholder, it will be judged to be reliable if its liveness is

very high, even though it may deliver garbled and inaccurate output and not satisfy other stakeholders whose value propositions emphasize intelligibility and accuracy. ISO/IEC does not define Resilience, but if it also defined it in terms of "specified functions and conditions," as it does for other ilities such as Effectiveness and Usability, a system specified to be Resilient under one of the over-15 domain-varying definitions of Resilience above, would be judged to be resilient even though it did not satisfy the resilience definitions of success-critical stakeholders in other domains. Such one-size-fits-all definitions (also employed in other quality attribute guidance descriptions) thereby present hindrances to effective interdisciplinary collaboration.

ISO/IEC 25010 has other shortfalls as a candidate ontology for system quality attributes or ilities. It is inconsistent in its implied coverage of physical systems, including aircraft in its 4.4.11 definition of "system," but limiting its systems scope to software products and software-intensive computer systems in its Introduction and elsewhere. Although one would assume that any aspect of Quality in Use would be an aspect of Product Quality, these sections of the standard are often incompatible. Some attempt toward compatibility of the Quality of Use and Product Quality characteristics is provided at the beginning of Section 4.1, in which the Note under Table 3 defines the Usability Product Quality attribute in terms of the Quality in Use characteristics of Effectiveness, Efficiency, and Satisfaction. But this compatibility is undermined by the different subcategories of particularly Satisfaction and Usability. Satisfaction includes Trust, which overflows Usability and involves Reliability, Safety, and Security. It also includes Comfort, which is nowhere to be found in Usability. Usability includes Learnability and Operability, which are nowhere to be found in Satisfaction.

Other guidance descriptions for quality attributes have similar shortfalls as bases for a useful ontology across multiple system domains. In particular, most do not explicitly recognize or emphasize that the attribute values vary by environmental conditions, operational scenarios, and stakeholder value propositions. The initial ontology described next attempts to do this.

1.2.3 Ontology Definitions and Choices

For an ontology structure, this Initial Definition of an Ilities Ontology (IDIO) adopts the more recent 1993 Gruber definition[21] of ontology, and a variation of the often-used IDEF5 structure[22], which includes the elements Kind, Individual, Referent, Relation, State, and Process. IDIO substitutes Class for Kind, to capitalize on the concepts of class hierarchies and inheritance in object-oriented design and development. Class hierarchies in IDIO are organized in terms of stakeholder value propositions as system-ility end objectives, and means-ends relationships in terms of child-class ilities as means for achieving the parent-class ility end objectives.

The intent in IDIO is that the subclasses of each class are mutually exclusive and exhaustive, in which the means do not overlap each other, and combine to completely satisfy the ends of the parent class. For example, the three means for achieving the end parent class of Flexibility are Modifiability, Tailorability, and Adaptability. Modifiability accomplishes Flexibility by facilitating changes in the system's structure or state, via such sub-ilities as system Understandability,

Analyzability, Modularity, and Testability. Tailorability accomplishes Flexibility without changes in the system's overall structure or state, via such mechanisms as generics, design patterns, and plug-compatible receptors. Adaptability accomplishes Flexibility without human intervention via trend or anomaly analysis and self-modifying the system's structure or state to cope with threats or opportunities.

These parent-child relationships are reasonably mutually exclusive and exhaustive, but the ideal of such relationships is not always achievable. For example, such sub-elements as Modularity and Testability are also important in achieving Tailorability and Adaptability. They are also important to achieving other parent classes such as Dependability and Affordability. IDIO recognizes the necessity of such many-to-many relationships among parent and child classes, but unfortunately some standards such as ISO/IEC 25010 limit themselves to one-to-many parent-child relationships. It defines Testability as only a contributor to Maintainability, neglecting the importance of Testability in achieving other ility classes such as Functional suitability, Performance efficiency, Usability, Reliability, Security, and Portability.

Overall, we have found that the two upper levels of the IDIO class hierarchy can be reasonably represented by one-to-many means-ends relationships, but that there are numerous many-to-many relationships among these and the lower-level ilities such as Modularity and Testability. The current upper structure of the IDIO class hierarchy is shown in Table 1. As mentioned at the beginning of this Section, the overall upper level of the hierarchy is based on a system's success-critical stakeholders' value propositions. These of course vary by stakeholder, but at a macro level, they also vary by stakeholder role, such as end-user, usage manager, acquirer, investor, developer, supporter, maintainer, and interoperator. The rationale for this choice is based on the INCOSE Handbook definition of systems engineering[23] as "An interdisciplinary approach and means to enable the realization of successful systems," and the Theory W (Win-Win) Fundamental System Success Theorem[24], which states that: A system will succeed if and only if it makes winners of its success-critical stakeholders.

Table 1. Upper Levels of IDIO Stakeholder Value-Based ilities Means-Ends Hierarcrchy

Stakeholder Value-Based ility Ends	Contributing ility Means
Mission Effectiveness	Stakeholders-satisfactory balance of Physical Capability, Cyber Capability, Human Usability, Speed, Endurability, Maneuverability, Accuracy, Impact, Scalability, Versatility, Interoperability
Resource Utilization	Cost, Duration, Key Personnel, Other Scarce Resources; Manufacturability, Sustainability
Dependability	Security, Safety, Reliability, Maintainability, Availability, Survivability
Flexibility	Modifiability, Tailorability, Adaptability
Composite ilities	
Affordability	Mission Effectiveness, Resource Utilization
Resilience	Dependability, Flexibility

The key stakeholder role category in an operational system's usage lifetime is its set of operational stakeholders with respect to the system's operational mission. These include its end-users, usage managers, investors, supporters, maintainers, and interoperators. Their primary ility value proposition is the system's Mission Effectiveness with respect to their mission. For service organizations, such as defense or fire protection, the operators of ground, sea, or air vehicles will be concerned primarily with the vehicles' mission effectiveness in terms of range, payload, speed, maneuverability etc. Usage managers will be concerned with those ilities as well, but also in the high priorities for interoperability, understanding, timeliness, accessibility, and accuracy of their command, control, and situation understanding capabilities, such as are summarized in the survey of combat commanders in the 2012 JCIDS Manual for the Operation of the Joint Capabilities Integration and Development System[25].

Another class of success-critical stakeholders are those who are investing key resources (funds, property, materials, personnel, services, etc.) to define, develop, operate, and evolve a system. They will have high priority value propositions not only on mission effectiveness but also the relative returns on their investments. The combined ility is often called Cost-Effectiveness and the resource expenditures often called Affordability, but IDIO follows INCOSE, NDIA, and MORS in interpreting Affordability as a cost-effectiveness composite ility, and categorizes the expenditures as Resource Utilization. Frequently, the resources utilized in developing the system's initial operational capability are dominated by the cost of manufacturing the full production line of the system and of sustaining its operation across the system's life cycle. The stakeholders' concern with these resources is represented in Table 1 by the ilities of Manufacturability and Sustainability.

A further class of success-critical stakeholders are people who are not involved in the system's definition, development, and evolution, but who are depending on the system to avoid their loss of property and quality of life during its operational performance (driving a car, using a chain saw, providing credit card and social security numbers, etc.). As shown on Table 1, the means for achieving this Dependability end include protection from vulnerabilities from system adversaries (Security); from natural causes, defects and human errors (Safety); from failure to deliver needed capabilities (Reliability), and from persistence of such failure for long periods of time (Availability, and its enabler Maintainability). It also includes fault tolerance and the ability to gracefully degrade under stress, called Survivabilty in Table 1.

Finally, all of these stakeholders are concerned with the system's ability to continue to provide or improve the desired levels of Mission Effectiveness, Resource Utilization, and Dependability as the world around it continues to provide increasing sources of change in technology, competition, market demands, organizations, and leadership. As discussed at the beginning of this section, this end value proposition of Flexibility includes the mutually-exclusive and exhaustive means of Modifiability, Tailorability, and Adaptability.

However, Flexibility and the Maintainability component of Dependability are not mutually exclusive. Initially, the Dependability component was being called Repairability, which more

precisely is the component of Maintainability that links Reliability and Availability in terms of Mean Time Between Failures (MTBF) and Mean Time To Repair (MTTR) in the equation Availability = MTBF / (MTBF+MTTR). Repair of defects is only a part of the scope of responsiveness to change that could be called either Maintainability, Flexibility, or Evolvability, but as with Affordability, preserving compatibility with the common use of the triad Reliability, Availability, and Maintainability (RAM) looked preferable to introducing a new Repairability term. The choice of Flexibility over Evolvability was based on its implication of more timely and natural response to change made it a better counterpart with Dependability in defining the composite attribute of Resilience in Table 1.

In summary, Table 1 summarizes the class hierarchy of the primary stakeholder ility end value classes of Mission Effectiveness, Resource Utilization, Dependability, and Flexibility, along with their primary means-ends subclasses, and the primary composite ilities of Affordability and Resilience. In terms the IDIO ontology's Individuals, each of the ility subclasses can be considered as Individuals with respect to their variation by states, processes, and relations, as will be described next. Where there are commonalities across the subclasses, the end value classes can also be considered as Individuals.

1.2.4 States, Processes and Relations

All too often, ilities are specified as single numbers, when in general they vary by state (the system's internal state and its external environment state), process (operational scenario), and relations with other ilities. An example is provided below for Reliability, followed by more detailed discussions of the sources of variation.

Frequently in system acquisitions, a Reliability requirement will be specified as a single number, such as "The system shall have a Mean Time Between Failures (MBTF) of 10,000 hours." There are several things wrong with this ility requirement. Two of them are that it does not define what it means to fail, and that it does not recognize that different stakeholders rely on the system for different definitions of failure. Often the definition is interpreted as 10,000 hours of system liveness, and for example a system may be delivered with 10,000 hours of liveness, but which fails to deliver several messages per hour and delivers several garbled messages per hour: clearly a failure for some success-critical stakeholders.

4.1 States: Thus, liveness, message-delivery, and message-accuracy would be components of the example system's state that are success-critical for its stakeholders, along with any other success-critical ilities such as system response time. The system can undergo state transitions from being live to not-live and back, or being live and full-delivery to live and partial-delivery and back, etc. These would be aspects of the system's internal state, along with additional aspects such as its current capacity for handling additional transactions. The system's degree of Reliability would also depend on aspects of its external state, such as its operation in a desert, jungle, swamp, or ice field. Other external-state examples involve the nature of the workload the system needs to process. A good counterexample to such a consideration was the ility requirement for a large command-control system to have a 5-second response time in

peacetime and a 2-second response time in a crisis – when the number of active users, transactions per user, and complexity of transactions would be significantly higher.

Another serious concern with one-size-fits-all ility definitions is that not all ility shortfalls are equally important. Some examples from the reliability literature are that roughly 20% of the defects account for roughly 80% of the business value[26]; roughly 10% of the defects account for 90% of the downtime, roughly 20% of the defects account for 80% of the rework, and roughly 20% of the modules account for 80% of the defects[27]. Subsequent empirical data on stakeholder value-based prioritization of design and code inspections[28] increased the business value gained per person hour by a factor of 2, and that business value-based testing[29] increased business value gained from 58% to 93%. In general, just having the stakeholders prioritize the system ilities and use cases is sufficient for project activity prioritization purposes[30], although more quantitative methods are also available[31]. A similar approach to prioritizing command and control transactions would provide a way to improve crisis performance in the example above.

4.2 Processes: ility values will also vary by the nature of the processes or operational scenarios that the system needs to be involved in. A common Reliability-related example is the sawtooth curve of predicted remaining defects as the system goes through integration test, acceptance test, field test, and actual operations, approaching zero at the end of each phase, and then jumping back up as a new class of users and usage scenarios are encountered.

Process definitions are also useful in identifying additional enablers of reliability and availability. A good example is the problem report closure means-ends process summarized in the book *Maintainablity*[32]. It identifies the process steps Detection, Preparation for Maintenance, Localization and Isolation, Disassembly (Access), Repair or Removal and Replacement, Reassembly, Alignment and Adjustment, and Condition Verification (Checkout). These process steps lead to the definition of enabling ilities such as Diagnosability, Defect Localizability, Accessibility, and Repairability. In some cases, enabling ilities need to be disambiguated across different parts of the ility hierarchy. For example, Defect Localizability is an enabler of Maintainability, while Product Localizability is an enabler of Flexibility, in terms of means to facilitate and localize product specialization across different countries, languages, power supplies, identifier formats, etc.

Another example of process support for aspects of Dependability is the means-ends summary of process choices for software defect detection and removal used in the Constructive Quality Model (COQUALMO)[33,34]. In concert with the COCOMO II software cost model[34], it identifies different levels of investment in software defect detection and removal via automated analysis, peer reviews, and execution testing and tools, and produces resulting estimates of delivered defect density in terms of number of defects per thousand lines of code, along with associated costs and returns on investment via the Information Dependability and Value Estimation (iDAVE) model[35].

Table 2. COQUALMO Defect Detection and Removal Investment Rating Scales

Rating	Automated Analysis	Peer Reviews	Execution Testing and Tools
Very Low	Simple compiler syntax checking.	No peer review.	No testing.
Low	Basic compiler capabilities	Ad-hoc informal walkthroughs	Ad-hoc testing and debugging.
Nominal	Compiler extension Basic requirements and design consistency	Well-defined sequence of preparation, review, minimal follow-up.	Basic test, test data management, problem tracking support. Test criteria based on checklists.
High	Intermediate-level module and inter-module; Simple requirements/design	Formal review roles with well-trained participants and using basic checklists, follow up.	Well-defined test sequence tailored to organization. Basic test coverage tools, test support system. Basic test process management.
Very High	More elaborate requirements/design Basic distributed-processing and temporal analysis, model checking, symbolic execution.	Basic review checklists, root cause analysis. Formal follow-up using historical data on inspection rate, preparation rate, fault density.	More advanced test tools, test data preparation, basic test oracle support, distributed monitoring and analysis, assertion checking. Metrics-based test process management.
Extra High	Formalized specification and verification. Advanced distributed processing	Formal review roles and procedures. Extensive review checklists, root cause analysis. Continuous review process improvement.. Statistical Process Control.	Highly advanced tools for test oracles, distributed monitoring and analysis, assertion checking Integration of automated analysis and test tools. Model-based test process management.

4.3 Relations: The combination of Dependability and Resource Utilization in the iDAVE model is an example of the relations among the ilities, in that the choices of investments in defect detection and removal in Table 2 produce estimates of both delivered defect density and cost, enabling a tradespace analysis of how much defect removal is enough for different stakeholder value propositions for Dependability and Resource Utilization.

A common project practice for addressing a high-priority ility such as Security is to set up an Integrated Process Team (IPT) to ensure that the system is highly secure. Often, the IPT will be so focused on Security that it will propose strategies that have adverse effects on other ilities. As some examples from project practice, a Security IPT proposed a single-agent key distribution system to minimize probability of compromise, only to have a Reliability engineer identify this as a system-level single point of failure. It also proposed an elaborate multilayer defense that would have caused a 50% Speed reduction and some potential real-time deadline problems. Further, it proposed an elaborate authentication scheme that would have caused some Usability problems such as startup delays, delegation problems, and GUI complexity On the other hand, the Security emphases on integrity enhanced aspects of Reliability, and the proposed Security defenses against denial-of-service attacks enhanced Speed. In general, this implies that individual ility IPTs should be completed by a ilities IPT that addresses the synergies and conflicts implied by the individual IPT strategies. Ideally, such an ilities IPT, or systems

engineers in general, would have a knowledge base and some tools that would help them diagnose potential inter-ility conflicts and potential additional inter-ility synergies that could be pursued. Creating such a knowledge base and set of tools is a major objective of the SERC iTAP initiative.

Table 3 below provides an example of the kind of capability that the SERC project is working on. It shows above the main diagonal the synergies between improvements in one of the ilities Dependability, Flexibility, Interoperability, and Resource Utilization and improvements in the others. It shows below the main diagonal the conflicts between improvements in one of those four ilities and impacts on the others. It is currently in a Word table; current experiments at U. of Virginia and USC are focusing on tools to enable users to obtain detailed explanations, and ideally quantitative relationships, for the synergies and conflicts.

An example is provided in the two entries in Table 3 in a larger font. In the envisioned tool, clicking on the conflict "Increased reliability increases acquisition costs" in the lower left cell would bring up the green curve shown Figure 1 below and its explanation. On the other hand, clicking on the synergy "Increased reliability decreases total ownership costs" in the upper right cell would bring up the full Figure 1 and its explanation. The dotted red curve shows that very Low RELY software is 23% more expensive to maintain. The blue curve shows that for long-lived projects with 70% of their life-cycle costs in maintenance, low-RELY software will be more expensive over the life cycle. The black curve shows that if total ownership costs include a comparable amount of business losses due to low-RELY software, that Very High RELY investments are cost-effective. The quantitative relationships are based on the calibrated values of the Required Reliability cost driver in the COCOMO II cost estimation model[34].

Table 3. Example Second-Level System Ilities Synergies and Conflicts

	Dependability	Flexibility	Interoperability	Resource Utilization
Dependability		Domain architecting (using domain knowledge in defining interfaces) improves reliability and modifiability	Common, multi-layered services and architecture improve interoperability and reliability	Automated input, output validation
		Modularity (high module cohesion, low module coupling) improves modifiability and reliability	Domain architecting improves reliability, interoperability within the domain	Cost, increase reliability
		Nanosensor-based smart monitoring improves reliability, makes mods more effective	High-cohesion, low-coupling modules improve interoperability and reliability	Cycle ownership costs
				Increased reliability reduces life
				Product line architectures reduce
				Reduces human costs
Flexibility	Domain architecting assumptions complicate multi-domain system modifiability		High-cohesion, low-coupling modules improve modifiability and interoperability	Modularization around sources of change
	Reliability-optimized designs may complicate fault diagnosis, system disassembly		Modularization around sources of change improves modifiability and interoperability	Domain architecting enables domain
			Open standards, service-oriented architectures improve both modifiability and interoperability	High-cohesion, low-coupling modules
				Modifiability and decreases lifecycle cost
				Product lines, reducing costs
				Providing excess capacity improves
				Reduce life cycle costs
				Peduces life cycle costs
Interoperability	Data redundancy improves reliability, but updates may complicate distributed real-time systems interoperability	Domain architecting assumptions complicate multi-domain system interoperability		Common, multi-layered services and
	Optimizing on reliability as liveness may degrade message delivery, accuracy			Architecture reduce life cycle costs
				Interoperability, reduces cost of later systems
				Product line architecture improves
Resource Utilization	Formal verification adds cost	Domain architecting increases multi-domain system costs	Neglecting or deferring interfaces to co-dependent systems will reduce initial costs, but degrade interoperability	
	Hardware redundancy adds cost	Fixed-requirements, fixed-cost contracts generally produce brittle, hard-to-modify systems	Product line architecture increases cost of initial system	
	Increased reliability increases acquisition costs	Providing excess capacity improves modifiability but increases acquisition cost		
	Making easiest-first initial commitments reduces early costs but degrades later reliability, adds later costs			

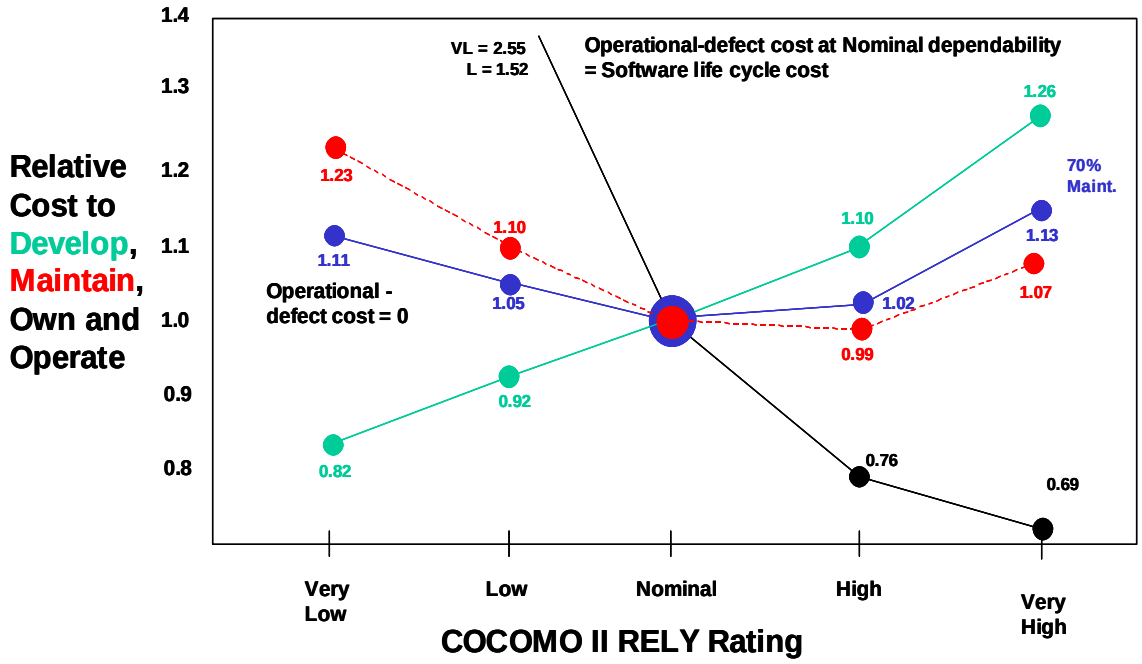


Figure 1 Software Life Cycle Ownership Costs vs Required Reliability(RELY)

The ultimate goal of the research in this area would be to provide such guidance on ility synergies and conflicts among all of the 26 second-level ilities in Table 3. Initial explorations of this goal concluded that the resulting 26x26 matrix would be too unwieldy, and too redundant, as there are many synergies among the second-level entries in each first-level category. For example under the Robustness first-level ility, every strategy to improve Reliability would also improve Availability, as would every strategy to improve Maintainability. Section 4.4 will show and discuss a less-redundant 7x7 matrix consisting of the four first-level ilities plus separate subsets of the complex second-level Mission Effectiveness ility to cover Physical Capability, Cyber Capability, and Interoperability ilities.

4.4 The 7x7 ility Synergies and Conflicts Matrix: The 7x7 matrix in Table 4 includes the three first-level ilities of Flexibility, Dependability, and Resource Utilization. plus expansions of the complex second-level Mission Effectiveness ility into separate Physical Capability, Cyber Capability, and Interoperability ilities along with a Mission Effectiveness category comprising the remainder of the Mission Effectiveness components.

Some additional synergy and conflict examples from the Flexibility and Dependability classes indicted in bold in Table 4 are:

Synergy: Domain Architecting. Special domain knowledge or domain ontologies enable simplification of domain artifacts and concepts. For example, specifying the Dependability aspects for a fixed wing aircraft can benefit from concepts common to fixed-wing aircraft aerodynamics, propulsion, controls, energy management, etc., These enable greater ease and consistency of definition, development, and modification of the desired aircraft, improving

both Dependability and Flexibility.

Conflict: Multi-Domain Architecting. Committing to several domain ontologies within a system of systems will complicate both Flexibility and Dependability, as the special domain

Table 4. Major-Category ilities Synergies and Conflicts Matrix

		Modularity	Many options	Agile methods	Spare capacity	Spare capacity	Loose coupling
		Self Adaptive	Service oriented	Automated I/O validation			Modularity
		Smart monitoring	Spare capacity	Loose coupling for sustainability			Product line architectures
		Spare Capacity	User programmability	Product line architectures			Service-oriented connectors
		Use software vs. hardware	Versatility	Staffing, Empowering			Use software vs. Hardware
							User programmability
lity							
	Accreditation		Accreditation	Automated aids	Fallbacks	Fallbacks	Assertion Checking
	Agile methods assurance		FMEA	Staffing, Empowering	Lightweight agility	Value prioritizing	Domain architecting within domain
	Encryption		Multi-level security	Value prioritizing	Value prioritizing	Redundancy	Service oriented
	Many options		Survivability	Automated I/O validation	Redundancy		
ibility	Multi-level security		Spare capacity	Domain architecting within domain	Spare capacity		
	Self Adaptive defects			Product line architectures			
	User programmability						
	Multi-domain modifiability						
	Autonomy vs. Usability	Anti-tamper		Automated aids	Automated aids	Automated aids	Automated aids
	Modularity slowdowns	Armor vs. Weight		Domain architecting within domain	Domain architecting within domain	Domain architecting within domain	Domain architecting within domain
	Multi-domain architecture interoperability conflicts	Easiest-first development		Staffing, Empowering	Staffing, Empowering	Staffing, Empowering	Staffing, Empowering
ctivenss	Versatility vs. Usability	Redundancy		Value prioritizing	Value prioritizing	Value prioritizing	
		Scalability					
		Spare Capacity					
		Usability vs. Security					
	Agile Methods scalability	Accreditation	Agile methods scalability		Automated aids	Automated aids	Automated aids
	Assertion checking overhead	Anti-tamper	Cost of automated aids		Domain architecting within domain	Domain architecting within domain	Domain architecting within domain
	Fixed cost contracts	Certification	Many options		Staffing, Empowering	Staffing, Empowering	Rework cost savings
	Modularity	Easiest-first development	Multi-domain architecture interoperability conflicts		Value prioritizing	Value prioritizing	Staffing, Empowering
ilization	Multi-domain architecture interoperability conflicts	Fallbacks	Spare capacity				
	Spare capacity	Multi-domain architecture interoperability conflicts	Usability vs. Cost savings				
	Tight coupling	Redundancy	Versatility				
	Use software vs. hardware	Spare Capacity, tool s costs					
		Usability vs. Cost savings					
	Multi-domain architecture interoperability conflicts	Lightweight agility	Multi-domain architecture interoperability conflicts	Cost of automated aids		Automated aids	Automated aids
pability	Over-optimizing	Multi-domain architecture interoperability conflicts	Over-optimizing	Multi-domain architecture interoperability conflicts		Staffing, Empowering	Domain architecting within domain
	Tight coupling	Over-optimizing		Over-optimizing		Value prioritizing	
	Use software vs. hardware						
	Agile Methods scalability	Multi-domain architecture interoperability conflicts	Multi-domain architecture interoperability conflicts	Cost of automated aids	Over-optimizing		Automated aids
ability	Multi-domain architecture interoperability conflicts	Over-optimizing	Over-optimizing	Multi-domain architecture interoperability conflicts	Physical architecture or cyber architecture		Domain architecting within domain
	Over-optimizing			Over-optimizing			
	Tight coupling						
	Use software vs. hardware						
	Multi-domain architecture interoperability conflicts	Encryption interoperability	Multi-domain architecture interoperability conflicts	Assertion checking	Over-optimizing	Reduced speed of Assertion checking	
ability	User-programmed interoperability	Multi-domain architecture interoperability conflicts		Cost, duration of added connectors	Tight vs. Loose coupling	Reduced speed of connectors, standards compliance	
						Tight vs. Loose coupling	

assumptions may have incompatibilities causing both ease-of change and dependability-of-change problems. For example, an aircraft domain ontology may assume a spherical Earth, while a ground vehicle ontology may assume a flat Earth, and a satellite vehicle ontology an oblate Earth. And it may be unclear whether the ground vehicle or satellite vehicle ontologies do or don't include aerodynamic effects. Again, the 7x7 matrix and its entries represent a necessarily incomplete and imperfect starting point. The intent will be to provide a Wikipedia or Systems Engineering Body of Knowledge (SEBoK)-like framework for the systems engineering community to discuss and improve the content.

4.5 Synergies and Conflicts Within the Major Categories: Another shortfall with respect to the synergies and conflicts among the major ility classes is that they are not perfectly uniform. If Dependability includes Reliability, Availability, and Maintainability, it is clear that many contributions to Flexibility will be synergetic with Maintainability, but may be conflicting with Reliability, as with user-programmable scripts. Thus it is important to complement the 7x7 cross-major-classes matrix with seven matrices addressing within-major-classes synergies and conflicts. This may seem like excess complexity, but it is the best compromise we have found between capturing important synergies and conflicts, and keeping things simple. It is certainly more practical to hat a 7x7 matrix with 42 entries plus 7 matrices averaging 4x4 elements and 12 entries, or 126 total entries; than it is to have a 26x26 matrix with 650 entries.

An example 3x3 matrix for intra-Dependability synergies and conflicts is shown in Table 5. At some cost in precision, it excludes Safety and Survivability as important but mostly duplicative in their relations with Security, Reliability, and Maintainability (another example of an ility tradeoff between exhaustiveness and simplicity).

Table 5. Intra-Dependability Synergies and Conflicts

	Security	Reliability	Maintainability
Security		Confidentiality, Integrity, Avalability	Certification
		Assurance Cases	Diagnosability
		Certification	Integrity, Avalability
		Failure Modes and Effects Analysis	Repairability
		Fault Tree Analysis	Smart Monitoring
		Recertification	Spare Capacity
Reliability	Non-redundancy (For Security)		Accessibility
	Redundancy (For Reliability)		Certification
			Diagnosability
			Repairability
			Smart Monitoring
			Spare Capacity
Maintainability	Accessibility	Armor	
	Compartmentalization	Recertification	
	Encryption		
	Recertification		

1.2.5 MULTI-STAKEHOLDER ILITY VALUE PROPOSITION RECONCILIATION

An ontology based strongly on stakeholder value propositions needs to recognize and deal with the fact that stakeholders' ility priorities will vary by their roles, responsibilities, and levels of need satisfaction in the Maslow need hierarchy[36]. As discussed under Table 1 in Section 3,

system end users, support operators, and administrators strongly need Mission Effectiveness. System sponsors and owners strongly need Affordability in terms of Mission Effectiveness and Resource Utilization. System maintainers strongly need Flexibility. External stakeholders affected by the system's operation strongly need Dependability as aircraft passengers and credit card holders.

The Synergies and Conflicts matrices presented above provide some of the support needed to understand the challenges of reconciling these value proposition conflicts, as does Multi-Attribute Utility Theory[37]. But further support is needed in terms of methods, processes, and tools to enable multi-discipline stakeholders to perform such reconciliations.

Current acquisition practices such as competitive bidding based on least-cost, technically acceptable selection criteria, followed by a fixed-price, fixed specification contract usually do the opposite. They generally force acquirers and bidders into a Conspiracy of Optimism set of commitments that lock them into unfulfillable contracts and subsequent adversarial relationships that often neglect the value propositions of other success-critical stakeholders.

Over the past 25 years, we have been addressing this challenge, beginning with the stakeholder win-win Theory W[38]. It states that "A system will be successful if and only if it makes winners of all of its success-critical stakeholders." (The INCOSE definition of Systems Engineering is, "an interdisciplinary method and approach for enabling the development of successful systems.")[39] The Theory W counterpart Successful System Realization Theorem includes four key process goals: (1) Identifying the system's success-critical stakeholders; (2) Determining their key value propositions; (3) Reconciling the value propositions into a mutually satisfactory (win-win) plans and specifications, and (4) Controlling the project to keep it in a win-win state[40]. Support of these goals also rests on major contributions to (1) dependency theory,[41] (2) utility theory,[42] (3) decision theory,[43] and (4) control theory,[44] and contributions to achieving the goals such as Getting to Yes[45] and a win-win approach to outsourcing called Vested Outsourcing.[46]

Also over the past 20 years, we have been developing and evolving collaboration tools that enable stakeholders from different disciplines to identify their value propositions or win conditions, and to negotiate mutually satisfactory or win-win agreements on the nature of a proposed system and its life cycle approach. There have been major challenges in doing this, such as providing supportive user interfaces for stakeholders having different knowledge or skill levels, and in determining criteria for prioritizing the agreed-upon capabilities. We have found that widely-used applications such as Facebook provide much better collaboration bases for diversified stakeholders, and have developed a Facebook-style tool called Winbook, which has significantly improved the speed, depth, and breadth of reaching concurrence across widely diversified stakeholders.[47] We have also been able to develop a capabilities prioritization approach based on the Minimum Marketable Features method,[31] and have led a successful activity to develop and apply 17 evaluation criteria to evaluate and select the most composite multi-stakeholder desirable of 17 alternative prioritization methods for a major information technology company.[48] The approach appears to have promise for addressing complex multi-criteria selection problems in general. We continue to experimentally apply it in an annual set

of 15-20 multi-stakeholder web services application projects as part of our Incremental Commitment Spiral Model approach to successful projects in the USC neighborhood area.[49]

1.2.6 Conclusions

Ilities or non-functional requirements are success-critical for many projects. They represent major stakeholder needs, but are a major source of project overruns and failures. They are a significant source of stakeholder value conflicts. They are poorly defined and understood, even in standards documents. Compared to functional requirements, they are seriously underemphasized in project management.

The initial ilities ontology presented here helps address these problems by clarifying the nature of system ilities. It uses a modified IDEF5 ontology framework, with ility classes represented in a stakeholder value-based, means-ends hierarchy. It identifies sources of ility value variation in terms of a system's states, processes, and inter-ility relations. The relations enable ility synergies and conflicts identification, providing the beginnings of satisfying a major need for systems engineers. Continuing SERC research at USC, MIT, and the University of Virginia is creating further tools and formal definitions that will strengthen the ontology and provide further support for systems engineers. Again, the ultimate intent will be to provide a Wikipedia or Systems Engineering Body of Knowledge (SEBoK)-like framework for the systems engineering community to discuss and improve the ilities knowledge base.

1.2.7 Acknowledgements

This material is based upon work supported in part by the U.S. Department of Defense through the Systems Engineering Research Center (SERC) under Contract H98230-08-D-0171. SERC is a federally funded University Affiliated Research Center managed by Stevens Institute of Technology. It is also supported by the National Science Foundation grant CMMI-1408909, Developing a Constructive Logic-Based Theory of Value-Based Systems Engineering.

1.2.8 References

1. Boehm B, Unifying software and systems engineering, *IEEE Computer* 2000; 3: 114-6.
2. Vitruvius, (Rowland I, Howe T trans), *Ten books on architecture,* Cambridge University Press; 2001.
3. Ruskin J, *The seven lamps of architecture,* Dover Publications; 1989.
4. Wellington A*, The economic theory of the location of railroads,* New York, NY, USA: John Wiley and Sons; 1887.
5. Pirsig R, *Zen and the art of motorcycle maintenance,* New York, NY, USA, William Morrow & CO., 1974.
6. Rubey R, Hartwick D, Quantitative measurement of program quality, *Proc ACM National Conference,* ACM; 1968: 671-7.

7. Boehm B, Brown J, Kaspar H, Lipow M, MacLeod G, Merritt M, *Characteristics of Software Quality*, NBS Technical Report,1973; Amsterdam: North Holland, 1978.
8. Gilb T, *Principles of software engineering management,* Reading MA: Addison-Wesley; 1988.
9. Barbacci M, Klein M, Longstaff T, Weinstock C, *Quality Attributes,* Technical Report CMU/SEI-95-TR-021: 1995
10. Clements P, Kazman R, Klein M, *Evaluating Software Architectures*, Reading MA: Addison Wesley; 2002.
11. ISO/IEC, *Software engineering – product quality standard 9126,* ISO/IEC; 2001-2004.
12. ISO/IEC, *Software engineering – software product quality requirements and evaluation (SQuaRE) standards 25000-25030*, ISO/IEC; 2005-2008.
13. ISO/IEC, *Systems and software engineering – SQuaRE system and software quality models standard 25010,* ISO/IEC; 2011.
14. Holland J, Engineered resilient systems (ERS) roadmap*, Proc. NDIA Sys Engr Conf*; 2012.
15. Ross A, Hastings D, The tradespace exploration paradigm*, Proc. INCOSE Intl Symp*; 2005.
16. Ross A, Stein D, Hastings, D, Multi-attribute tradespace exploration for survivability*, AIAA J Spacecraft Rockets;* 2014.
17. Sitterle V, Curry M, Freeman D, Ender T, Integrated toolset and workflow for tradespace analytics in systems engineering*, Proc INCOSE Intl Symp;* 2014.
18. Boehm B, Lane J, Koolmanojwong S, An orthogonal framework for improving life cycle affordability, *Proc. Conf Sys Engr Rsch;* 2013.
19. Wikipedia, *Resilience(disambiguation)* http://en.wikipedia.org/wiki/Resilience_(disambiguation), accessed 09/29/2014
20. Brand F, Jax K, Focusing the meaning(s) of resilience: resilience as a descriptive concept and a boundary object. *Ecology and Society* **12**(1): 23.; 2007
21. Gruber T, A translation approach to portable ontologies. *Knowledge Acquisition* 1993; 5(2):199-220.
22. Benjamin P et al., *IDEF5 Method Report*. Knowledge Based Systems, Inc.; 1994.
23. INCOSE. *INCOSE Systems Engineering Handbook*, INCOSE-TP-2003-002-03.2; 2012.
24. Boehm B, Jain A, A Value-Based Theory of Systems Engineering. *Proceedings, INCOSE Intl. Symp.;* 2006.
25. JCIDS, Manual for the Operation of the Joint Capabilities Integration and Development System, JCIDS; 2012: A-A-1-4.
26. Bullock J, Calculating the Value of Testing, *Software Testing and Quality Engineering*, 2000; 3: 56-62.
27. Boehm B, Basili V, Software Defect Reduction Top 10 List, *Computer,* 2001; 1: 135-137.
28. Lee K, Boehm B, Empirical Results from an Experiment on Value-Based Review (VBR) Processes, *Proc. International Symposium on Empirical Software Engineering*; 2005.
29. Li Q, Yang Y, Li M, Wang Q, Boehm B, Hu C, Improving Software Testing Process: Feature Prioritization to Make Winners of Success-Critical Stakeholders, *Journal of Software Maintenance and Evolution; 2012*
30. Thelin T, Runeson P, Wohlin C, Prioritized use cases as a vehicle for software inspections, *IEEE Software*, 2003; 4: 30-33.
31. Denne M, Cleland-Huang J, *Software by Numbers,* Prentice Hall; 2003.
32. Blanchard B, Verma D, Peterson E, *Maintainability,* John Wiley and Sons, 1995.

33. Chulani S, *Bayesian Analysis of Software Costs and Quality Models,* PhD Dissertation, Department of Computer Science, University of Southern California, 1999.
34. Boehm B, Abts C, Brown AW, Chulani S, Clark B, Horowitz E, Madachy R, Reifer D, Steece B, *Software Cost Estimation with COCOMO II,* Prentice Hall, 2000.
35. Huang L, Boehm B, How Much Software Quality Investment is Enough: A Value-Based Approach, *IEEE Software 2006; 3*, 23 (5): 88-95.
36. Maslow A, *Motivation and Personality*, Harper 1954.
37. Keeney R, Raiffa H, *Decisions with Multiple Objectives: Preferences and Value Tradeoffs,* Cambridge University Press 1976.
38. Boehm B, Ross R, "Theory-W Software Project Management Principles and Examples," *IEEE Transactions on Software Engineering*, 1989;15, (7): 902-916.
39. Boehm B., Jain, A., "A Value-Based Theory of Systems Engineering," *Proceedings, INCOSE IS,* 2006
40. INCOSE, *INCOSE Systems Engineering Handbook*, Version 3.2, 2012.
41. Burton R, Obel B, *Strategic Organizational Diagnostics and Design: The Dynamics of Fit*, Kluwer, 2004.
42. Debreu G, *Theory of Value*, Wiley, 1959.
43. Blackwell D, Girshick M, *Theory of Games and Statistical Decisions*, Wiley, 1954.
44. Brogan W, *Modern Control Theory*, Prentice Hall, 1974 (3rd ed., 1991).
45. Fisher, R., Ury, W., *Getting To Yes: Negotiating Agreement Without Giving In,* Houghton Mifflin, 1981.
46. Vitasek, K, Ledyard M, Manrodt K, *Vested Outsourcing,* Palgrave Macmillan, 2010 (2nd ed, 2013)
47. Kukreja N, Boehm B, "Social Networking-Based System Requirements Engineering," *Proceedings, Conference on Systems Engineering Research,* Elsevier: 2013
48. Kukreja N, Payyavula S, Boehm B, and Padmanabhuni S, "Value-Based Requirements Prioritization: Usage Experiences," *Proceedings, Conference on Systems Engineering Research,* Elsevier: 2013
49. Boehm B, Lane J, Kooolmanojwong S, Turner R, *The Incremental Commitment Spiral Model: Principles and Practices for Sysccessful Systems and Software,* Addison Wesley, 2014

1.3 Task 2: Methods and Tools Piloting and Refinement

USC participated in activities with WSU, NPS, TARDEC, and NAVSEA to apply SERC cost estimation capabilities to Army and Navy systems engineering analyses.

1.4 Task 3: Next-Generation, Full-Coverage Cost Models

Based on the exploratory needs and data assessments in Phase 2, a Phase 3 workshop including Air Force, Navy, aerospace industry, and SERC researchers concluded that there were strong needs for better estimation of operations and support costs, but that the data available lacked adequate cost driver information, except in in the software area. The workshop recommended that the most promising initial areas to pursue would be for software development, systems

engineering, and the use of systems engineering cost drivers to improve estimation of system development costs. Further research and workshops have identified further sources of data and some shortfalls in current models in these areas, and have developed requirements and draft frameworks for the next-generation models. An example of the cost driver elements and rating scales is provided next.

1.4.1 Draft COSYSMO 3.0 Rating Scales: Problem and Solution Understanding

Problem Understanding and Solution Understanding are the first two macro cost drivers proposed for the COSYSMO 3.0 model. Each is broken down into several sub-aspects to be rated and weighted-averaged into a composite rating level to be used to determine a composite macro cost driver value across a 7-level Likert scale. Their productivity ranges will be determined by a combination of Expert-Delphi determined values and of multiple regression of project data.

The use of composite cost driver parameters rather than individual parameters is to reduce the number of parameters in the model for both ease of use and ease of calibration of the parameters. Further, determining weighted averages provides a rational basis for creating interpolated rating levels such as 3.56, as compared to the big swings between integer-rated multipliers for the individual parameters in the earlier COSYSMO models. It also will generally provide a central tendency in the composite rating level as extreme ratings are averaged with more central or opposite ratings.

Problem Understanding includes understanding of the system's concept of operation, of functional requirements, and of non-functional requirements or ilities. Solution Understanding includes understanding of the most appropriate system architecture, of the most appropriate choices of non-developmental items (COTS products, cloud services, reused components), of the degree of understanding of the system-critical technologies, and of the degree of the system-critical technologies' maturity with respect to the system's scale and needed performance.

We considered combining Problem Understanding and Solution Understanding, as it is generally good systems engineering practice to work on these concurrently. But we decided to make them separate for two main reasons. One is to avoid complex macro cost drivers with numerous sub-aspects. The other is that often the estimators (and data contributors) are strong in problem understanding but not strong in solution understanding, or vice versa, and it was valuable to capture such knowledge and use it in developing and using COCOMO 3.0.

When we began developing 7-level rating scales for each of the sub-aspects of Problem Understanding and Solution Understanding, we found that the levels of understanding were essentially the same across the sub-aspects. For each, it was the case that understanding involved the levels of knowledge about the system's key factors: the system's domain (e.g.,

government/commercial, market sector, cyber/physical/human), its stakeholder values, its environment, its organizational relationships, or other system-influential factors.

The rating level scales then follow progressions along three major axes:

1. The highest level of understanding of each key factor held by at least one of the systems engineering team members.
2. The average level of understanding of the factors across the entire team.
3. The number of team members able to bridge multiple key factors.

Table 6 summarizes the rating level scales proposed to be used for all of the Problem and Solution Understanding sub-factors. The Bloom taxonomy from Wikipedia shown in Table 7 is used to define the rating scale levels: Knowledge, Comprehension, Application, Analysis, Evaluation, and Synthesis. I have taken the liberty of replacing the weak Evaluation question in Wikipedia beginning with "Questions like: Do you feel that" by a more pragmatic "Questions like: Evaluate sources of evidence that."

Table 6. Problem and Solution Understanding Sub-Factor Rating Scales

Rating Level \ Axis	1. Best Und'g Level	2. Avg. Team Und'g Level	3. Factor Bridging
Extra Low 1.0	Basic Knowledge of each factor covered	Minimal Knowledge of most factors	Some factor pairs at Knowledge level
Very Low 2.0	Comprehension, some Application of most factors covered	Knowledge of most factors, some Comprehension and Application of some factors	Some pairs at Comprehension level
Low 3.0	Application, some Analysis of most factors covered	Comprehension and Application of most factors, some Analysis of some factors	Some multi-factor bridging at Comprehension, Application levels
Nominal 4.0	Analysis, some Evaluation and Synthesis of some factors covered	Comprehension and Application of each factor, some Analysis of most factors	Some multi-factor bridging at Analysis level
High 5.0	Analysis, some Evaluation and Synthesis of most factors covered	Analysis of most factors, some Evaluation and Synthesis of some factors	Most factors bridged at Analysis level. Some at Evaluation level
Very High 6.0	Analysis, most Evaluation and Synthesis of most factors covered	Analysis of each factor, some Evaluation and Synthesis of most factors	Most factors bridged at Evaluation and Synthesis levels
Extra High 7.0	Evaluation and Synthesis of each factor covered	Most Evaluation and Synthesis of most factors	All factors bridged at Evaluation and Synthesis levels

Table 7. Bloom Taxonomy of Cognitive Understanding Domain

There are six levels in the Bloom taxonomy, moving through the lowest order processes to the highest:

Knowledge

Exhibit memory of learned materials by recalling facts, terms, basic concepts and answers

- Knowledge of specifics - terminology, specific facts
- Knowledge of ways and means of dealing with specifics - conventions, trends and sequences, classifications and categories, criteria, methodology
- Knowledge of the universals and abstractions in a field - principles and generalizations, theories and structures

Questions like: What are the health benefits of eating apples?

Comprehension

Demonstrate understanding of facts and ideas by organizing, comparing, translating, interpreting, giving descriptions, and stating the main ideas

- Translation
- Interpretation
- Extrapolation

Questions like: Compare the health benefits of eating apples vs. oranges.

Application

Using acquired knowledge. Solve problems in new situations by applying acquired knowledge, facts, techniques and rules in a different way

Questions like: Which kinds of apples are best for baking a pie, and why?

Analysis

Examine and break information into parts by identifying motives or causes. Make inferences and find evidence to support generalizations

- Analysis of elements
- Analysis of relationships
- Analysis of organizational principles

Questions like: List four ways of serving foods made with apples and explain which ones have the highest health benefits. Provide references to support your statements.

Evaluation

Present and defend opinions by making judgments about information, validity of ideas or quality of work based on a set of criteria

- Judgments in terms of internal evidence
- Judgments in terms of external criteria

Questions like: Evaluate sources of evidence that serving apple pie for an after school snack for children is healthy.

Synthesis

Builds a structure or pattern from diverse elements; it also refers the act of putting parts together to form a whole (Omari, 2006). Compile information together in a different way by combining elements in a new pattern or proposing alternative solutions

- Production of a unique communication
- Production of a plan, or proposed set of operations
- Derivation of a set of abstract relations

Questions like: Convert an "unhealthy" recipe for apple pie to a "healthy" recipe by replacing your choice of ingredients. Explain the health benefits of using the ingredients you chose vs. the original ones.

1.5 Future Plans

1.5.1 Task 1, Foundations

Based on the initial ontology developed in Phase 3, USC will converge the ility definitions and ontology content, including variations by domain for key DoD domains, and work with UVa and MIT to formalize the ontology definitions and structure and harmonize them with the MIT change-oriented ilities structure in Phase 4. In Phase 5, these will be extended to further domains, and guidance documents developed for their use.

1.5.2 Task 2: Methods and Tools Piloting and Refinement

Based on initial participation in Service piloting of cost estimation models and tools in Phase 3, USC will participate in further Service piloting in Phase 4 of further cost estimation models and tools being developed in Task 3, and refine the models and tools based on usage experience. Such improvements, piloting and refinement will continue in Phase 5. Complementary collaboration with Georgia Tech will extend the integration of the cost modeling tools with the Georgia Tech SysML architecting tools.

1.5.3 Task 3: Next-Generation, Full-Coverage Cost Models

Based on initial research on cost drivers for next-generation systems engineering and software cost models in Phase 3, Phase 4 will develop prototype systems engineering and software cost

models and tools for piloting and refinement, and extend the estimation capabilities toward full-coverage. This will primarily involve the following efforts to:

- Continue to involve groups of domain experts to review and iterate the definitions and develop first-order expert-judgment Delphi estimates of the CER cost driver ranges. Three Government-industry workshops in October 2013, February 2014, and April 2014, evolved a baseline identification and prioritization of candidate cost drivers, and an initial Delphi workshop is scheduled for October 2014.
- Evolve baseline detailed definitions of the cost driver parameters and rating scales for use in data collection.
- Gather initial data and determine areas needing further research to account for wide differences between estimated and actual costs.

2 MASSACHUSETTS INSTITUTE OF TECHNOLOGY

2.1 PAST RESULTS: PHASES 1 AND 2

2.1.1 TASK 1. ILITY FOUNDATIONS

MIT Semantic Basis.

In Phase 1, MIT built on prior research that aimed to develop a radical new approach for defining ilities, rather than simply proposing yet another set of definitions. MIT has proposed the use of a semantic basis that can serve as a framework for formulating ility "definitions." Such a basis would provide a common language that would inherently demonstrate how various ilities relate to one another and provide an opportunity for discovering new ilities as well as provide a new representation for meaning of various ilities beyond English definitions. Phase 1 matured the earlier work, resulting in a proposed semantic basis, made up of fourteen categories. This basis is believed to span the *change-type* ility semantic field and excludes the *architecture-type* semantic field that includes "bottom" ilities such as modularity (de Weck, Ross, and Rhodes 2012) and "architecture principles" (Fricke and Schulz 2005) such as simplicity. Beginning with the change agent and change effect as two categories for defining a change and the resulting applicable ilities, a larger set of categories are proposed for defining a larger set of possible changes for a system. The fourteen categories, which together form the semantic basis, are intended to collectively define a change in a system, thereby creating a consistent basis for specifying change-type ilities in formal statements. A system can be verified to display the quality described in the statement and therefore be traceable to a desired higher order system property. (An earlier version of this basis is described in Beesemyer 2012). The fourteen categories are: cause, context, phase, agent, impetus, impetus nature, impetus parameter, impetus destination state size, impetus aspect, outcome effect, outcome parameter, outcome destination state size, outcome aspect, level of abstraction, and value qualities of the change. Unique choices for each of these categories, when applied to a particular system parameter will formulate the change-type ility statement. The fourteen categories are illustrated in Figure 2.

The semantic basis aims to capture the essential differences among change-type ilities through specification of the following general change statement with regard to a particular system parameter:

> ***In response to "cause" in "context", desire "agent" to make some "impetus parameter change" in "system" resulting in "outcome parameter change" that is "valuable."***

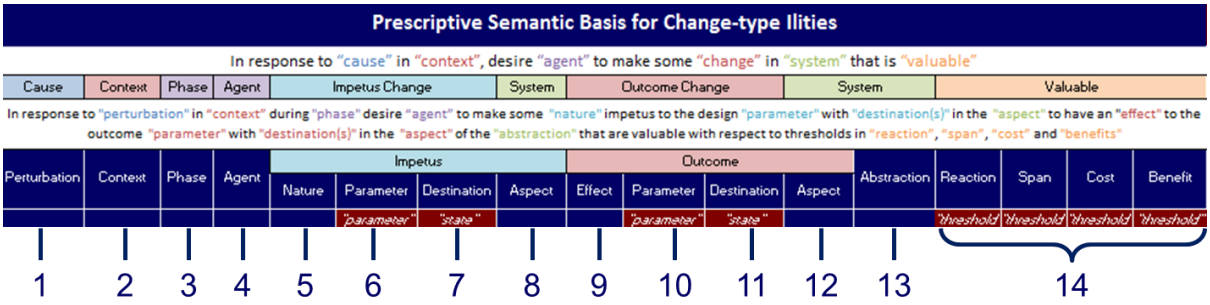


Figure 2. Change-type prescriptive semantic basis in 14 categories. (Ross, Beesemyer, Rhodes2011)

Application of the semantic basis begins with a user generating a change statement. The change statement is refined and assigned categorical choices within the basis, with the intention that the applicable ilities will emerge from the specified change statement. In this way, a user does not need to use a particular ility label a priori, thereby avoiding the semantic ambiguity in the terms. If the basis accurately and completely describes the underlying categories for change-type changes, then a user should be able to describe any change-type ility through the basis. Further description of the semantic basis can be found in the RT-46 Phase 1 Technical Report. The version of the semantic basis above served as the version for undertaking further research investigation in collaboration with UVA during Phase 2.

In Phase 2, based on a critique of the semantic basis provided by UVA, MIT convened an internal working group and provided further clarifications and reformulations to address the feedback. This working version has addressed a number of critiques from UVA on the MIT 14-dimension semantic basis, as well as accumulated MIT-internal critiques. Refined understanding of the semantic basis through use cases has also led to enhancements.

MIT provided UVA with the current working version, and gained feedback from UVA in regard to encoding the basis into a formal language as was done of the initial version of the 14-dimension semantic basis provided by MIT. In order to carefully consider the points in the UVA critique, MIT convened special working sessions with a number of students and researchers with prior experience with both ilities and the semantic basis specifically, to review UVA's interpretation and formulation. As a result, MIT recommended clarifications and some changes to the basis to fill gaps and provide clarification. The revised basis has 20 categories as developed during Phase 2; it is shown in the figure below.

Prescriptive Semantic Basis for Change-type Ilities

In response to "perturbation" in "context", desire "agent" to make some "change" in "system" that is "valuable"

Perturbation	Context	Phase	Agent	Impetus Change	Mech	Outcome Change	System	Valuable* (this category is not complete)

In response to "perturbation" in "context" during "phase" desire "agent" to make some "nature" impetus to the system "parameter" from "origin(s)" to "destination(s)" in the "aspect" using "mechanism" in order to have an "effect" to the outcome "parameter" from "origin(s)" to "destination(s)" in the "aspect" of the "abstraction" that are valuable with respect to thresholds in "reaction", "span", "cost" and "benefits"

Perturbation	Context	Phase	Agent	Impetus* (optional)					Mech	Outcome					Abstraction	Reaction	Span	Cost	Benefit
				Nature	Parameter	Origin	Destination	Aspect	Mechanism	Effect	Parameter	Origin	Destination	Aspect					
optional	circumstantial; required; general; optional	null	optional	null	required	optional	optional	null* (this is implied by "parameter")	Optional	null	required	optional	optional	null*	optional	required	required	required	required
"name"	*"name(s)"*		*"name(s)"*		*"parameter"*	*"state(s)"*	*"state(s)"*		*"name"*		*"parameter"*	*"state(s)"*	*"state(s)"*		*"name"*	*"threshold w/units"*	*"threshold w/units"*	*"threshold w/units"*	*"threshold w/units"*
none	circumstantial	pre-ops	none	decrease	level	one	one	form		decrease	level	one	one	form	architecture	sooner	shorter	less	more
disturbance	general	ops	internal	same	set	few	few	function		same	set	few	few	function	design	later	longer	same	same
shift	<empty>	inter-LC	external	increase	<empty>	many	many	operations		increase	<empty>	many	many	operations	system	always	same	more	less
<empty>		<empty>	either	not-same		<empty>	<empty>	<empty>		not-same		<empty>	<empty>	<empty>	<empty>	<empty>	<empty>	<empty>	<empty>
			<empty>	<empty>						<empty>									

Figure 3. The full revised semantic basis (gray row describes "name" label for that category).

MIT believes the semantic basis would be used differently in different use cases. During Phase 2 the team began working on defining various use cases, with a subset of the full category set. For example, the full basis might be used when trying to write a very specific requirement statement. But that use should not occur until AFTER analysis to determine what should be done. Early in the design phase, one would likely leave out the "valuable" categories (as these are subjective and depend on outside factors). Additionally, if one is trying to avoid fixating on a solution-centric approach, one might want to consider leaving out change mechanism (in order to allow engineers to propose their own alternatives). Several use cases are described in the Phase 2 Technical Report.

2.2 Phase 3 Results

2.2.1 Task 1. ility Foundations

In Phase 3, MIT continued efforts on the collaborative research and development of iTAP scientific foundations, in the development of more rigorous definitions and quantification of ilities and their relationships. Specifically, MIT focused on the continuing development of the semantic basis, usability testing of the basis, and collaboration with UVA researchers in formalism.

MIT Semantic Basis.

In Phase 3, MIT built upon the prior phase efforts on the prescriptive semantic basis for consistently representing ilities within a particular semantic field. The semantic basis, evolved and tested in Phases 1 and 2, was further developed and tested in Phase 3. Through successive refinement, the categories in the basis were derived from an earlier effort to define a 10, and later 14, dimensional basis for describing change-type ilities (Figure 4.

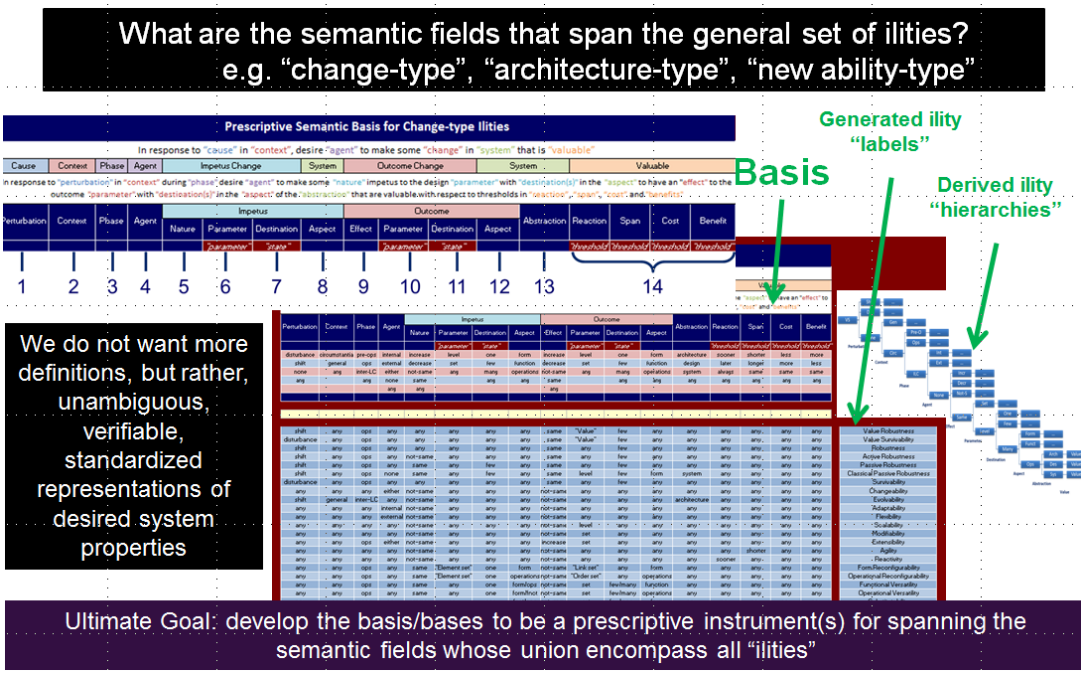


Figure 4. Basis began with 14 categories.

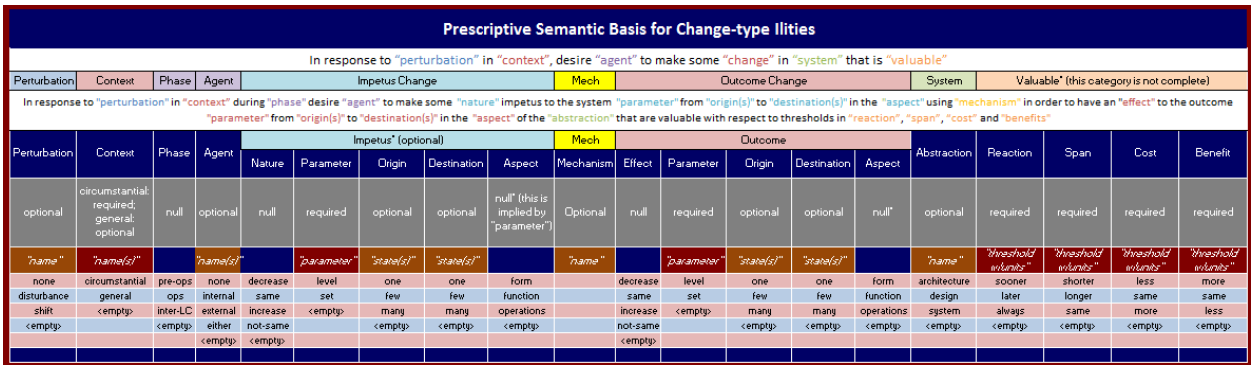
Subsequently this was extended to 20 categories based on feedback from collaborators, as well as internal review and testing. Beginning with change agent and change effect as two categories for defining a change, a larger set of categories in the full basis is proposed for defining a larger set of possible changes for a system. The twenty categories, which together form the semantic basis, are intended to collectively define a change in a system, thereby creating a consistent basis for specifying change-type ilities in formal statements. A system that can be verified to display the quality described in the statement can then have a traceable and consistent means for displaying a verifiable desired ility.

The twenty categories are: perturbation, context, phase, agent, impetus (nature, parameter, origin states, destination states, aspect), mechanism, outcome (effect, parameter, origin states, destination states, aspect), level of abstraction, and value qualities of the change (reaction, span, cost, benefit).

Unique choices for each of these categories will formulate the change-type ility statement. The twenty categories along with their associated choices are illustrated in Figure 5. The semantic basis aims to capture the essential differences among change-type ilities through specification of the following general change statement with regard to a particular system parameter:

> **In response to “perturbation” in “context” during “phase”, desire “agent” to make some “nature” impetus to the system “parameter” from “origin(s)” to “destinations” in the “aspect” using “mechanism” in order to have an “effect” to the outcome “parameter” from “origin(s)” to “destination(s)” in the “aspect” of the “abstraction” that are valuable with respect to thresholds in “reaction”, “span”, “cost” and “benefit.”**

Application of the semantic basis begins with a user generating a change statement. The change statement is refined and assigned categorical choices within the basis, with the intention that the applicable ilities will emerge from the specified change statement. For this use to work, particular combinations of choices in the basis must be assigned an ility term label. Such an exercise is similar to assigning definitions for ilities, albeit with an imposed closed form "little language" supplied by the basis. For example, if "agent" is set to "external" then the "flexible" label will apply. In this way, a user does not need to use, or know, a particular ility label a priori, thereby avoiding semantic ambiguity in the terms. If the basis accurately and completely describes the underlying categories for change-type changes, then a user should be able to describe any change-type ility through the basis consistently.



Prescriptive Semantic Basis for Change-type Ilities

In response to "perturbation" in "context", desire "agent" to make some "change" in "system" that is "valuable"

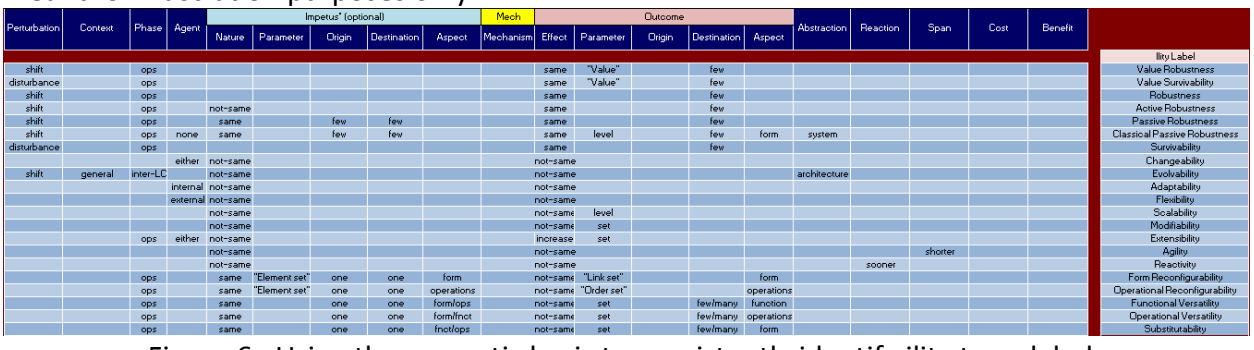
Perturbation	Context	Phase	Agent	Impetus Change	Mech	Outcome Change	System	Valuable* (this category is not complete)

In response to "perturbation" in "context" during "phase" desire "agent" to make some "nature" impetus to the system "parameter" from "origin(s)" to "destination(s)" in the "aspect" using "mechanism" in order to have an "effect" to the outcome "parameter" from "origin(s)" to "destination(s)" in the "aspect" of the "abstraction" that are valuable with respect to thresholds in "reaction", "span", "cost" and "benefits"

Perturbation	Context	Phase	Agent	Impetus* (optional)					Mech	Outcome					Abstraction	Reaction	Span	Cost	Benefit
				Nature	Parameter	Origin	Destination	Aspect	Mechanism	Effect	Parameter	Origin	Destination	Aspect					
optional	circumstantial: required; general: optional	null	optional	null	required	optional	optional	null* (this is implied by "parameter")	Optional	null	required	optional	optional	null*	optional	required	required	required	required
"name"	*"name(s)"*		*"name(s)"*		*"parameter"*	*"state(s)"*	*"state(s)"*		*"name"*		*"parameter"*	*"state(s)"*	*"state(s)"*		*"name"*	*"threshold in/units"*	*"threshold in/units"*	*"threshold in/units"*	*"threshold in/units"*
none	circumstantial	pre-ops	none	decrease	level	one	one	form		decrease	level	one	one	form	architecture	sooner	shorter	less	more
disturbance	general	ops	internal	same	set	few	few	function		same	set	few	few	function	design	later	longer	same	same
shift	<empty>	inter-LC	external	increase	<empty>	many	many	operations		increase	<empty>	many	many	operations	system	always	same	more	less
<empty>		<empty>	either	not-same		<empty>	<empty>	<empty>		not-same		<empty>	<empty>	<empty>	<empty>	<empty>	<empty>	<empty>	<empty>
			<empty>	<empty>						<empty>									

Figure 5. Change-type prescriptive semantic basis in 20 categories

In order to validate the proposed use of the basis, two activities need to be accomplished: (1) refinement of the basis itself, both in terms of complete categories and in terms of choices within those categories; and (2) if the use of particular ility terms is ultimately desired, a mapping between patterns in the basis and ility terms needs to be generated. If the latter is accomplished, then any usage of the basis for specifying particular change statements will result in consistently derived ility term labels. It is hypothesized that particular change-type ilities will correspond to particular choice(s) in this basis. In this way, a consistent method for specifying ilities can be pursued. An example of using the semantic basis for mapping ility terms is illustrated in Figure 6. The results in the figure are presently a work in progress and are meant for illustration purposes only.



Perturbation	Context	Phase	Agent	Impetus* (optional)					Mech	Outcome					Abstraction	Reaction	Span	Cost	Benefit	
				Nature	Parameter	Origin	Destination	Aspect	Mechanism	Effect	Parameter	Origin	Destination	Aspect						Ility Label
shift		ops								same	"Value"		few							Value Robustness
disturbance		ops								same	"Value"		few							Value Survivability
shift		ops								same			few							Robustness
shift		ops		not-same						same			few							Active Robustness
shift		ops		same		few	few			same			few							Passive Robustness
shift		ops	none	same		few	few			same	level		few	form	system					Classical Passive Robustness
disturbance		ops								same			few							Survivability
			either	not-same						not-same										Changeability
shift	general	inter-LC		not-same						not-same					architecture					Evolvability
			internal	not-same						not-same										Adaptability
			external	not-same						not-same										Flexibility
				not-same						not-same	level									Scalability
				not-same						not-same	set									Modifiability
		ops	either	not-same						increase	set									Extensibility
				not-same						not-same							shorter			Agility
				not-same						not-same						sooner				Reactivity
		ops		same	"Element set"	one	one	form		not-same	"Link set"			form						Form Reconfigurability
		ops		same	"Element set"	one	one	operations		not-same	"Order set"			operations						Operational Reconfigurability
		ops		same		one	one	form/ops		not-same	set		few/many	function						Functional Versatility
		ops		same		one	one	form/fnct		not-same	set		few/many	operations						Operational Versatility
		ops		same		one	one	fnct/ops		not-same	set		few/many	form						Substitutability

Figure 6. Using the semantic basis to consistently identify ility term labels

Using the basis. The twenty categories can be a bit overwhelming at first; alternative versions where the "optional" categories are left out, can be used at different stages of the design

lifecycle. For example, leaving out the “mechanism” column means that the statement leaves the mechanism ambiguous, allowing engineers to evaluate alternative mechanisms for meeting the statement. Likewise, leaving out the “impetus” categories results in an outcome-oriented change statement. In Phase 3, the team continued work on alternative use cases; these are being further defined and will help to generate specific examples.

Beginning with a desired system parameter change statement, the user makes choices across the categories, signified by the double quotation marks (“ ”) in the general form of the statement. It is important that this statement be specific to a chosen system parameter, as currently the change-type ilities are conceived as being defined only in relation to particular system parameters (i.e. there is no such thing as “generalized flexibility” as one can always find exceptions). Choices in the categories for the particular considered change result in a categorized change description. Optional categories do not have to be used if not of interest for the statement use.

As an example, a simple (optional categories excluded) statement could be: “in response to loud noises at night,” desire “owner” to be able to “change the level of volume” of “his stereo” “in less than one second.” This statement requests “flexible” (i.e., external agent is owner) “scalability” in “volume” and specifies the value proposition (i.e., what is deemed “valuable” is that it take less than one second to execute this change). Alternative stereos can be evaluated on this basis, with those stereos able to change more quickly as being more valuably scalable in this regard. If ility labels have been previously mapped to the basis, then one can automatically derive the applicable ility terms implied by the categorized change statement (i.e., flexibly scalable in this case). The distinction here is important since one could have generated an alternative statement that did not specify an external agent (i.e., resulting in “flexible” label), rather leaving the agent open or even internal (e.g., the stereo changes its own volume with associated design implications, and is labeled “adaptable” instead of “flexible”).

Each item in double quotes (“ “) corresponds to categories that differentiate the change-type ilities. As a prerequisite to specifying a change statement, one must have in mind the “parameter” of the “system” that is being affected. The “perturbation” refers to whether the change is in response to a perturbation of finite duration (disturbance), or one likely to last (shift), or no perturbation at all (none), or if it doesn’t matter (<empty>). The “context” refers to whether the response is desired in a particular case (circumstantial) or many cases (general), or it doesn’t matter (<empty>). The “phase” of the lifecycle when the response occurs has choices of pre-ops (before operations), ops (during operations), inter-LC (between lifecycles), or doesn’t matter (<empty>). The “agent” is the force that instigates the response through an impetus to the system; active response requires a change agent, which can be internal or external to the “system” boundary, while passive response doesn’t require an agent (none), or it may not matter (<empty>).

Changes within the statement are framed as “impetus” and “outcome” to capture how one might want to specify a range of changes in inputs (impetus), and the resulting range of changes in outputs (outcome). Both the impetus and outcome sections of the statement are

described by five categories: 1) the “nature” (impetus) or “effect” (outcome) of the change (decrease, remain same, increase, not-same, or doesn’t matter=<empty>); 2) what type of “parameter” change (in level, set or <empty>); whether there is a target number of 3) “origin” states and 4) “destination” states (each ranging from one, to few, to many to <empty>); and 5) the “aspect” referring to whether the change is in the form, function, operations, or doesn’t matter (<empty>). The “mechanism” specifies how the outcome is achieved.

The “abstraction” refers to the level of abstraction of the system, which can be at the architecture level (e.g., Boeing 737 aircraft), design level (e.g., 737-800), system level (e.g., 737-800, tail #: C-FTCZ), or doesn’t matter (<empty>). Lastly, a grouped set of categories describe how the change can be evaluated as valuable. This set of “valuable” categories relates aspects of value to thresholds in “reaction” (timing) and “span” (duration), “cost” (resources), and “benefit” (utility). “Value” is separated from the rest since it represents a coupled set of tradeoffs that can be used to judge the goodness described in a change statement. For example, one might be willing to accept later, slower, and more expensive if it provides greater utility, but if it provides less utility, then maybe it should be sooner, faster, and cheaper. Many combinations of these can result in ility statements representing valuable change.

A more complete version of the earlier change statement example could then be: In response to a loud noise (*perturbation*) late at night (*context*), during operations (*phase*) of system, desire owner (*agent*) to be able to ***impetus*** {increase (*nature*) the knob angle level (*parameter*) from one state (*origin*) to many states (*destination*) in the system form (*aspect*)} through turning the knob (*mechanism*) that results in the ***outcome*** {increasing (*effect*) the volume level (*parameter*) from one state (*origin*) to many states (*destination*) in the system function (*aspect*)}in the owner’s stereo system (*abstraction*) that takes less than 1 second (*valuable*).
As research progresses on the prescriptive semantic basis, several open questions remain: What types of ilities can be represented in the basis? Can or should the basis be expanded or modified? What are the appropriate basis categories? What are appropriate choices within each category?

With regard to mapping specifications within the basis to particular ility term labels, are there consensus patterns in matching ilities to the basis given particular definitions for each ility? Are there consensus patterns for given ility terms without provided definitions (i.e. is there some inherent, general meaning within an ility term that can be more consistently expressed using the basis than through traditional English phrase-based definitions)?

More generally, this research begins to structure the question regarding what semantic fields span the general set of ilities. Preliminary results (Ross et al., 2011) indicate that at least three semantic fields may exist in the general set of “ilities” including change-type, architecture-type, and new ability-type (the last kind includes such ilities as “auditability,” “learnability,” and “drinkability”). Identifying and classifying the current existing ility terms into appropriate semantic fields will serve to eliminate ambiguity in meaning, usage, and application, as well as allow for the explicit consideration of trade-offs within the semantic field. A consistent basis within a field can allow for direct comparison of its members; for example kinship terms clearly

distinguish between the meaning of uncle and cousin, even though a single person could serve in both roles. Using the prescriptive semantic basis approach also allows one to consider whether each semantic field can be represented with an internally consistent basis.

Revisiting the concept of relationships amongst the ilities, the basis can provide a first order approximation to clarify semantic differences amongst ilities within a particular semantic field. For example the difference between “flexibility” and “adaptability” is whether the change agent is external to or internal to the system’s boundary, respectively. The basis will also point out how a given change statement can display multiple ilities simultaneously. For example, agility is with regard to how quickly the change can be executed, so one could desire an agile, scalable change to describe a quick and level-increasing system parameter change. In this way, the difficulty in placing “agility” within the means-ends hierarchy study, described above, can be clarified, as any change could get labeled as agile, depending on its application and desires of the change statement writer. An additional investigation will be needed in order to clarify the relationships between different semantic fields. For example, how are members of the “architecture-type” semantic field related to the “change-type” semantic field? Preliminary work investigating design/architecture principles have begun to describe relationships between particular ilities and principles, but a unifying understanding of the relationship between semantic fields is not yet mature. Our working hypothesis is that “architecture-type” ilities are enablers for “change-type” ilities.

Given a stable, validated basis, can practitioners or academics use the basis to generate change statements, which will automatically label with the appropriate ilities? Do the ility term labels resonate with the users? The purpose of the research is not to generate more definitions, but rather, unambiguous, verifiable, standardized representations of desired system properties. One of the possible emergent results of this work may be the discovery of “new” ilities that do not yet have ility term labels, and yet may represent important desired system lifecycle properties, such as the distinction between functional versatility and operational versatility seen in Figure 7. Inverting the concept shown (i.e., achieving similar function with similar operations using multiple forms) results in a “new” ility we can label with “substitutability” and is a property displayed in computers, for example, where multiple different disk drives or monitors can be substituted for one another.

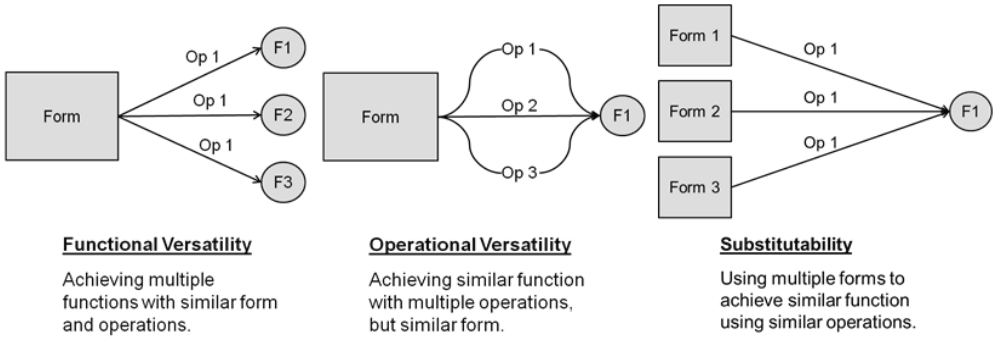


Figure 7 (a) Functional versatility; (b) operational versatility; (c) substitutability as suggested by the semantic basis

The ultimate goal of this research is to develop the basis or bases to be prescriptive instrument(s) for spanning the semantic fields whose union encompasses all "ilities." With such an instrument, practitioners can have a consistent and (potentially) complete list of possible ilities to consider for their systems, as well as a means to create verifiable requirements and system specifications. Academics can have a consistent basis for enhancing ility-related research and a means to advance the quantification of and clarification of relationships amongst ilities in general, as well as a means to educate future generations of engineering students so that one day ilities can become part of the lexicon of successful project managers.

To realize this goal, usability testing is an important activity. During Phase 3, MIT performed continued usability testing of the basis including the conception of a translation to assist in communicating the statement into more understandable language. As a first pass, color-coding was used to help parse the full basis change statement, as shown here:

In response to "perturbation" in "context" during "phase" desire "agent" to make some "nature" impetus to the system "parameter" from "origin(s)" to "destination(s)" in the "aspect" using "mechanism" in order to have an "effect" to the outcome "parameter" from "origin(s)" to "destination(s)" in the "aspect" of the "abstraction" that are valuable with respect to thresholds in "reaction", "span", "cost" and "benefits"

The full basis is used when trying to write a very specific requirement statement, and should not occur until after analysis to determine what should be done. In different use cases of the basis, variations on the full statement are useful.

A subset of the basis with 11 categories can be used if there is a constraint to make use of an existing/inherited mechanism, for example. This leaves open the "valuable" specification, but leaves in the "mechanism" category to constrain how the change should occur.

In response to "perturbation" in "context" during "phase" desire "agent" using "mechanism" to have an "effect" to the outcome "parameter" from "origin(s)" to "destination(s)" in the "aspect" of the "abstraction"

A subset of the basis with 10 categories is used early in the design phase, in order to not over specify the change mechanism. This allows engineers to propose/evaluate alternatives, or impetus. Leaving out the “valuable” part of the statement supports exploration. Later, when implications of ility statements are better understood, one can specify subjective thresholds on what makes the change “valuable”.

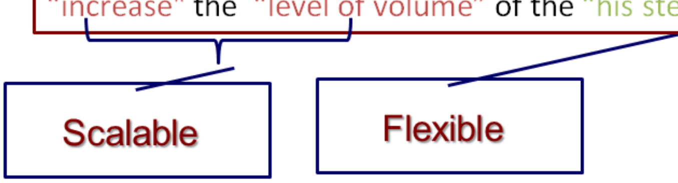
> In response to "perturbation" in “context” during "phase" desire “agent” to have an "effect" to the outcome "parameter" from "origin(s)" to "destination(s)" in the "aspect" of the "abstraction" that are valuable

A short example is shown below

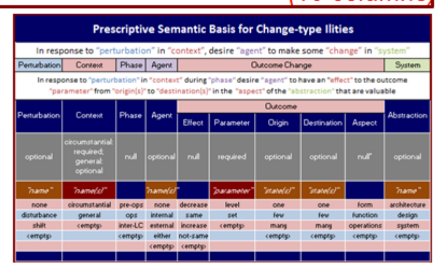
> In response to "perturbation" in “context” during "phase" desire “agent” to make some "nature" impetus to the system "parameter" from "origin(s)" to "destination(s)" in the "aspect" using "mechanism" in order to have an "effect" to the outcome "parameter" from "origin(s)" to "destination(s)" in the "aspect" of the "abstraction" that are valuable with respect to thresholds in “reaction”, “span”, “cost” and "benefits"

> In response to "perturbation" in “context” during "phase" desire “agent” to be able to "effect" the outcome "parameter" of the "abstraction" that is valuable

> In response to “loud noises" in “night”, during “ops" desire “owner” to be able to “increase" the “level of volume” of the “his stereo” in “less than one second”. *(10 columns)*



Desire stereo to be ***flexibly scalable*** in volume
in *less than one second*



Prescriptive Semantic Basis for Change-type Ilities

In response to “perturbation” in “context”, desire “agent” to make some “change” in “system”

Perturbation	Context	Phase	Agent	Outcome Change					System

In response to “perturbation” in “context” during “phase” desire “agent” to have an “effect” to the outcome “parameter” from “origin(s)” to “destination(s)” in the “aspect” of the “abstraction” that are valuable

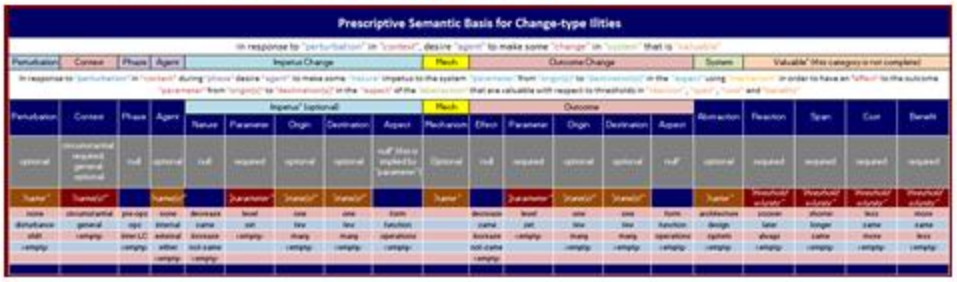
Perturbation	Context	Phase	Agent	Outcome					Abstraction
				Effect	Parameter	Origin	Destination	Aspect	
optional	circumstantial: required; general: optional	null	optional	null	required	optional	optional	null*	optional
"name"	"name(s)"		"name(s)"		"parameter"	"state(s)"	"state(s)"		"name"
none	circumstantial	pre-ops	none	decrease	level	one	one	form	architecture
disturbance	general	ops	internal	same	set	few	few	function	design
shift	<empty>	inter-LC	external	increase	<empty>	many	many	operations	system
<empty>		<empty>	either	not-same		<empty>	<empty>	<empty>	<empty>
			<empty>	<empty>					

Longer variations of this example include:

> In response to “loud noises" in “night”, during “ops" desire “owner” to be able to “increase" the “level of volume” of the “his stereo” in “less than one second”

> In response to “loud noises” at “night” during “ops“, desire “owner” to be able to ***impetus***
> {“increase" the “knob angle level" from “one state" to “many states" in the “system form“}
> through “turning the knob" that results in the ***outcome***
> {“increasing" the “volume level" from “one state” to “many states" in the “system function"}
> in the “owner’s stereo" that “takes less than one second”

Full basis: 20 columns



Full use of the semantic basis-derived change statement is clearly a more verbose version of the same change. But the illustration points out the consequences of varying the level of specificity. For example, one could have removed the "owner" as the change agent and left that category blank, allowing designers to develop a system that as other potential change agents (e.g. a software agent, allowing for adaptive volume control by the system itself). Similarly, alternative definitions and thresholds for "valuable" could have been used, and always reflect tradeoffs (e.g. maybe willing to wait 20 seconds for volume change effects if the dollar acquisition cost of the system is substantially less).

Examples collection

During this phase of research the team continued capturing descriptive change statements from historical systems, mapping them to the semantic basis-derived change statement. This also resulted in inferring related "ility labels" (i.e. the ility terms that we currently map to a subset of choices within the semantic basis). An example subset of these changes is shown here:

System Name	Change Description	Related Ility Labels
HMMWV	In response to improved IEDs and ambush tactics in urban warfare, desire AM General to design some increase in protection/armor in the HMMWV for quick valuable long term deployment.	Changeability, Flexibility, Scalability, Value Robustness,
B-52	In response to increased demand in alternative fuel sources in a context of climbing dependence on foreign fossil fuels, desire the Air Force to change fuel mixtures used in B-52s engines (synthetic fuels).	Changeability, Flexibility, Reconfigurability, Modifiability, Extensibility,
F-16	In response to need for longer range, desire ground crew to add external tanks for fuel to aircraft at hard points to increase fuel storage for increased range.	Changeability, Flexibility, Survivability, Scalability, Reconfigurability,
F-18 (Swiss)	In response to higher load cycles in a new operating environment (country), desire Swiss engineers to make some improvement to strength by replacing aluminum ribs with titanium ribs, that is always available and valuable.	Changeability, Flexibility, Scalability, Reconfigurability, Value Robustness,
S-92	In response to changing market needs in the civilian helicopter sector and FAA regulations, desire Sikorsky to evolve the S-70 into a new helicopter suitable for military and civilian purposes. Developed from the S-70 or Black Hawk family, the S-92 was planned to utilize as many components and subsystems from the highly reliable Black Hawk. The S-92 ended up with a redesigned a new dynamic component system, rotor, and gearbox.	Changeability, Flexibility, Modifiability, Evolvability, Value Robustness,
HMMWV	In response to "improved IEDs and ambush tactics" in "urban warfare", desire "Soldiers in the field" to "make a change to armor" in the "HMMWV" for immediate valuable deployment. This change refers to "Hill Billy" armor additions made by soldiers in the field to the humvees to help protect against small arms fire.	Changeability, Flexibility, Scalability, Reconfigurability, Value Robustness,
HMMWV	In response to "increased weight of the system" in "the new humvees that require more armor for protection", desire "AM General" to "design stronger chassis and better suspension" in the "HMMWV" for immediate valuable deployment.	Changeability, Flexibility, Scalability, Value Robustness,
A-10	In response to direct hits from armor-piercing and high explosive projectiles up to 23mm to the cockpit during close air support, desire the A-10 to withstand attack and protect pilot, always.	Survivability,
A-10	In response to direct hits small arms fire to fuel tanks during close air support, desire the A-10 to seal leaks in fuel system to minimize fires, explosions, and fuel supply depletion.	Survivability,
F-14	In response to changing aerodynamic environments, desire central air data computer to adjust wing sweep angle to improve lift to drag ratio and change aerodynamic characteristics of the aircraft.	Changeability, Adaptability, Scalability, Reconfigurability, Value Robustness,

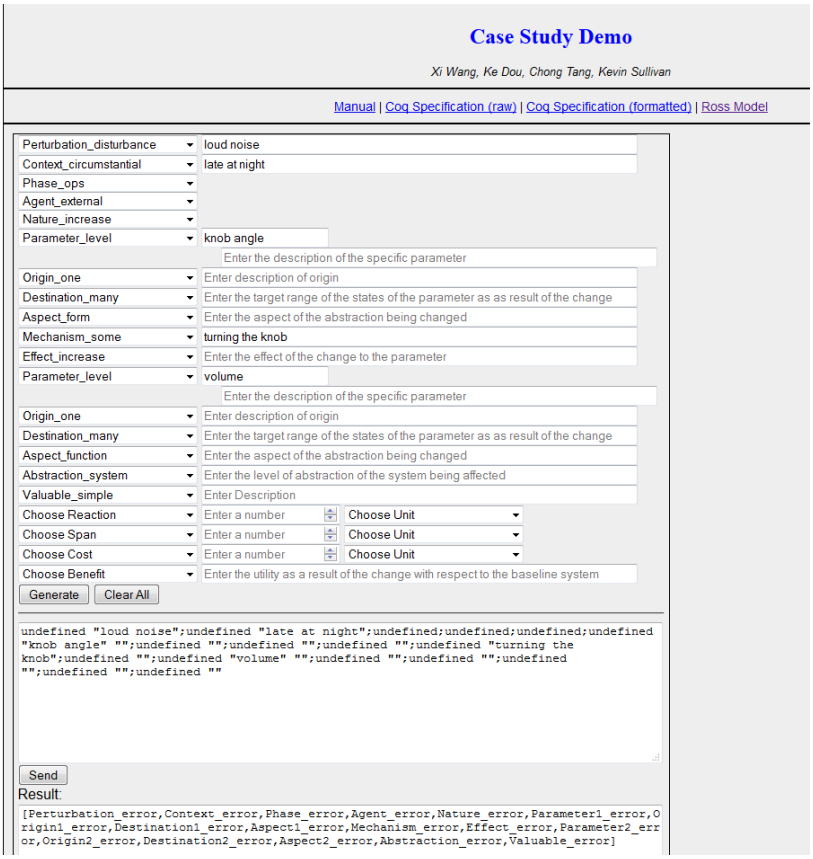
In response to "perturbation" in "context" during "phase" desire "agent" to have an "effect" to the outcome "parameter" from "origin(s)" to "destination(s)" in the "aspect" of the "abstraction" that are valuable

In response to "improved IEDs and ambush tactics" in "urban warfare", during "pre-ops" desire "AM General" to make an "increase" to the "armor" from "baseline" to "above baseline" in the "form" of the "HMMWV design" that is valuable

For existing changeable systems, this is a descriptive capture of a change statement. Based on data availability, these statements are captured at various levels of specificity. In reality, the most detailed version of the change statement could theoretically be populated given sufficient data.

Collaboration

During this phase the MIT team interacted with UVa in spiraling on a web service implementation of the basis. The interaction resulted in clarifications needed for the categories, including "naming" and optional fields. Challenges in basis usage highlighted the need for illustrative examples as well as motivating the need for the translation layer. A screenshot of the "Ross Model" (i.e. a slightly earlier version of the semantic basis) is shown below. More details on this effort are described in the UVa section of this report.



Active collaboration with UVA has continued in the effort to refine the semantic basis, as UVA worked formalization. It is recognized that the full verbose statement of the 20-category basis is unwieldy when viewed in "English", and that construction of "plain English" phrasing is partly customized based on particular statement. The research team believes a "translator" would be valuable in converting the basis category choices into English. We see this translation layer and underlying "little language" as key contributions of this work. Next phase research will develop the translation layer and seek collaboration with team for feedback and testing. Initial efforts were made on developing a concept for a translation layer, with examples of statements from basis.

Submitted Publications during this Phase

1. **Ross, A.M.**, and Rhodes, D.H., "Towards a Prescriptive Semantic Basis for Change-type Ilities," 13th Conference on Systems Engineering Research, Hoboken, NJ, Mar. 2015.
2. Dou, K., Wang, X., Tang, C., Sullivan, K., and **Ross, A.M.**, "Computational Foundations for a Science of Ilities, Tradeoffs, and Affordability," 13th Conference on Systems Engineering Research, Hoboken, NJ, Mar. 2015.

2.2.2 REFERENCES

1. Beesemyer J.C., Fulcoly D.O., Ross A.M., Rhodes D.H. Developing methods to design for evolvability: research approach and preliminary design principles. 9th Conf on Sys Eng Research. Los Angeles, CA, April 2011

2. Beesemyer, J.C. (2012), *Empirically Characterizing Evolvability and Changeability in Engineering Systems*, Master of Science Thesis, Aeronautics and Astronautics, MIT, June 2012

3. de Weck, O.L., Ross, A.M., and Rhodes, D.H. (2012), "Investigating Relationships and Semantic Sets amongst System Lifecycle Properties (Ilities)", *3rd International Engineering Systems Symposium*, CESUN 2012, TU Delft, 18-20 June 2012

4. 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, pp. 342-359

5. Ross, A.M., Beesemeyer, J.C., and Rhodes, D.H., (2011) "A Prescriptive Semantic Basis for System Lifecycle Properties", SEAri Working Paper WP-2011-2-1, MIT, Cambridge, MA, http://seari.mit.edu/papers.php.

6. Ross, A.M., Rhodes, D.H., and Hastings, D.E. (2008), "Defining Changeability: Reconciling Flexibility, Adaptability, Scalability, Modifiability, and Robustness for Maintaining Lifecycle Value," *Systems Engineering*, Vol. 11, No. 3, pp. 246-262

2.3 FUTURE PLANS: PHASES 4 AND 5

2.3.1 TASK 1. ILITY FOUNDATIONS

In FY2015, the MIT team will further refine the ilities semantic basis for change-related ilities, building on the work and using feedback from the UVA team. The team will continue working with UVA in their effort to develop a REST (representational state transfer) web-based service implementation as a means for formalization and testing of the basis. This will inform potential use cases and architecture needs for semantic basis evolution. The evolved semantic basis will be validated using historical examples as well as through user feedback. The historical examples will also be used to help illustrate usage of the basis.

The team will design and develop an interpreter layer for translating the semantic basis into a user-friendly interface. The developed interface will be tested with users in order to validate the usefulness of the semantic basis. The evolved semantic basis is intended to generate repeatable, rigorous change-related ilities statements, along with possible accompanying metrics that can be used for verification.

The team will continue working with UVA both for collaboration on development of the translation layer, as well as formalization of the underlying basis. A joint paper is intended as an outcome of this work. The team will also work with GaTech to find opportunities for using the basis to inform their work. The team will also collaborate with the broader foundations iTAP team members for opportunities to inform their ilities research.

In FY2016, MIT will develop a software-based implementation of the evolved semantic basis for potential deployment to government organizations. As the software is developed, further refinements to the translation layer will be implemented as necessary. Documentation with case examples will be created. Demonstrations and early adopter training will be conducted. Trial use will inform the development of next steps toward a broader transfer and implementation strategy. In FY2016, in support of the overall research on the foundations, the team will provide critical feedback and review of other iTAP team member research.

3 UNIVERSITY OF VIRGINIA

3.1 PAST RESULTS: PHASES 1 AND 2

UVa's Phase 1 efforts produced a web-based tool, called *doc-ility*, to maintain an evolving knowledge base of models of ilities and tradeoffs among them. Recognizing that the knowledge that would be disseminated by such a tool remained informal and unvalidated, UVa's Phase 2 efforts demonstrated the feasibility of formalizing of ility and tradeoff models and early results in synthesizing software services that expose in useful form the abstractions formalized in such artifacts. UVa also initiated a case study with MIT focusing on formalization and automation of the MIT group's work on a semantic basis for change-related ility definitions. The Phase 2 work produced an initial formalization of MIT's informal semantic basis for change-related ilities, and synthesized code implementing an abstract syntax tree data type and a semantic classifier for ility statements in MIT's language. UVa produced a trustworthy "compiler" for this language by hand-crafting Haskell code "around" the synthesized code for core data representations and functions. This work yielded a command line tool that can be run on desktop computers. UVa shared this work with the MIT group and hopes that the combination of MIT's original work with UVa's formal work will lead to a joint paper in the months ahead.

3.2 PHASE 3 RESULTS

A key UVa aim for Phase 3 was to develop a method for efficiently and reliably incorporating synthesized ility modeling and analysis code into *web-based tools* (like Doc-Ility). The overall goal was to make otherwise inaccessible work based on formal methods available on the Web to the SERC research team, and to prototype a method for making such work visible and useful to the broader systems engineering community. UVa viewed and views further development of a more precise and formal approach to understanding "ilities" as a key foundation for further development of a science of tradeoffs among ilities.

In Phase 3, UVa expanded its intentions to include not mere formalization but dissemination of formal models for purposes of evolving and eventual community-based validation of such ility models. UVa therefore sought to develop, and developed, infrastructure to support exposure of *formal* ility models not through a web *site* but through RESTful web services. In Phase 3, UVa thus developed, demonstrated, and partly validated the idea that one can not only formalize ility models in Coq but can then synthesize REST web services from such models of ilities, in order to enable community engagement around such models, driving model evolution and, we hope, eventual validation and adoption. A key is to enable rapid evolution of such web services as underlying mathematical-logical (Coq constructive logic) models evolve. UVa thus developed a method and tools to wrap Haskell code extracted from Coq ility specifications in Yesod-based REST web services.

The promise of this approach is to elevate the development of novel ility models from informal English-language statements to mathematically and logically precise, formal, and computable models, such that the key abstractions in such models can be disseminated to the community in a form (web services) that will support community engagement with and use of such models.
UVa tested the validity of this vision by running an experiment with MIT. UVa formalized and then synthesized web services for an evolving sequence of formal models of MIT's semantic theory of change-related ilities. UVa collaborated with the MIT group over the year to evolve multiple versions of an initial formal model of MIT's theory. UVa deems the demonstration of its formal-model-driven approach to ility theory development and validation a success. The MIT theory of change-related ilities is still evolving and is not obviously yet ready for production use, but the path toward validity and utility that UVa established seems very promising.

UVa also collaborated loosely with USC to ensure that UVa work was on track to be relevant to USC's ongoing efforts to formalize an ontology of system ilities and tradeoffs among ilities. In this period, UVa also continued its related efforts to formalize and automate frameworks for carrying out computational ility tradeoff analyses. This work included an initial demonstration of the instantiation of a generic functional programming approach to design space synthesis from relational logic specifications, the applications of design-space-specific ility estimation and measurement functions, and an early demonstration of the use of big data software (Hadoop and Spark) to carry out tradeoff analyses at increasing scale.

3.3 Future Plans: Phases 4 and 5

3.3.1 Task 1: Foundations

Based on initial experiments in Phases 2 and 3, Phase 4 will research and develop an initial formal basis for DoD-value-based ility definitions, including sources of variation (e.g., by domain, operational scenario, or stakeholder value proposition). Phase 4 will also apply the formal methods to strengthen the USC and MIT ility definitions, and extend the evolving DocIlity tool to address ility synergy and conflict relationships. Phase 5 will continue to apply, evaluate, strengthen, and extend the formal basis and tools developed in Phase 4. Additional details follow.

In work to date funded by the SERC iTAP project, UVa has developed and evaluated several innovative and synergistic approaches to developing foundations for both a science and an engineering practice of stakeholder-value-driven, ility definition, specification, tradeoff, and stakeholder impact and reconciliation analysis.

Value-Driven Evolution of Ility Science and Technology

Our overall goal is to achieve *value-driven evolution* of scientific theories and engineering technologies for understanding and managing ilities, tradeoffs, and stakeholder impacts in complex systems engineering projects and systems.

By *value-driven* we mean that theories should target and solve real problems, as evidenced by engagement of research and practitioner communities in the development, evolution, and eventual, *value-producing application* of such science and technologies.

The expected result of a successful application of our method to an ility, tradeoff, or value concept, *C,* is a 3-tuple, ($C_{Science}$, $C_{Technology}$, $C_{Evidence}$), where $C_{Science}$ is a mathematically, logically, and computationally precise and clear expression of the concept: a *theory*; (2) $C_{Technology}$ is an accessible and interoperable reference implementation of the theory in the form of a so-called REST web service, mechanically derived from the theory, both to avoid faulty implementations and to reduce the cost of theory evolution driven by user engagement with $C_{Technology}$; and (3) $C_{Evidence}$ is data that supports the proposition that the concept and theory have value in systems engineering research and/or practice, e.g., an active history of dialog about leading to the evolution and application of the concept over time,

Technical Approach

Our approach to fostering value-driven theory and technology evolution combines several elements in an attempt to achieve four key ilities in the resulting iTAP foundations: precision, accessibility, evolvability, and validity.

We aim for scientific *precision* in the expression of ility-related theories. To achieve it, we express our ideas in logically, mathematically, and computationally expressive, precise, mechanically checkable notations.

We aim for our theories to be *accessible* to researchers and practitioners who are far from familiar with such notations. To achieve accessibility, we implement our formal theories as interoperable, open REST web services.

We aim for rapid *evolvability* of theories and their corresponding implementations. Evolvability requires that we avoid costly reimplementation tasks each time a theory is found wanting and incrementally improved. We thus derive implementations automatically from formal models.

Most importantly, we mean to build *validated* foundations that are demonstrably useful to other researchers and eventually to practitioners. Our approaches to theory validation include both in-lab analysis as well as collaborative and experimental evaluation of theories with other researchers and eventually practitioners. Validity is in general expected to be an emergent

outcome of an incremental and evolutionary process of theory improvement catalyzed by the approach to theory expression, dissemination, and engagement that we have described here.

First, to overcome the tendency of discussions of ilities to remain rather vague and non-converging, we emphasize *precision:* that concepts and methods are eventually formulated with carefully checked mathematical, computational, and logical rigor.

The rest of this document summarizes results and emerging results to date and outlines how UVa proposes to build on them in 2015.

Activity #1: Making Formal Ility Theories Accessible

The goal of this work is to greatly accelerate both in-lab and collaborative external evaluation and evolution of scientifically formulated *"ility theories,"* evolving them from initial concepts to the point that they have demonstrated utility for research and development. This work will be described in a paper to be submitted to CSER 2015.

We have developed the capacity to automatically derive the core computational components of REST web services from formal ility theory expressions in the language of the Coq proof assistant. The result of applying this approach to a given theory is a REST web service and demonstration clients that allow one to engage with the theory by creating and analyzing new artifacts with respect to the ilities being modeled.

Our technical approach to this task includes synthesis of Haskell or Scala code directly from Coq specifications of ility theories, and embedding of this "extracted" code into web service frameworks, following by the writing of web-based client "apps" through which users can interact with and use the underlying theory. The web clients we have produced to date to demonstrate the approach provide (simple) model building capabilities (in web clients) and analysis capabilities (in web services), along with direct access to the underlying formal ility models. The theory is thus linked to useful tools making the theory broadly accessible.

3.3.2 PLANS FOR 2015

We have now demonstrated the use of this approach to formalize and evolve the semantic theory of change-related ilities of Ross et al. We plan to continue to develop and evaluate this approach: to support theories more complex than that of Ross-and-Rhodes models. We plan to extend their theory to include explicit system models (see next section) in terms of which change-related requirements can be expressed, and to leverage concepts of state change from computer science to bring computational rigor to the underlying theory. We plan to generalize our approach beyond their particular model. We also plan to develop sufficient software infrastructure to provide users of our web services with user accounts and functions to store

models and to provide compelling interactions with users. Our goal isn't merely -- or even mainly -- to build tools, but to use rapid synthesis of such tools from computational logic formulations of theories to close the theory-application-validation-evolution loop as previously discussed.

Plans for 2016 are summarized at the end of this section.

Activity #2: An Evolving Formal Theory of Change-Related Ilities

The goals of this task are (1) to evolve our current version of Ross's model to the point that it becomes demonstrable useful for articulating and classifying change-related ilities, and (2) by using our evolving version of Ross's model, to make fundamental advances in our approach to formalizing, automating, disseminating, testing, evolving, and ultimately validating scientific ility theories.

To drive initial development and testing of our approach, we took the early proposal by Ross et al. as a case study, as described above. Their goal, was to develop successful approaches to defining ility *terms*, particularly those relating to systems changeability -- terms such as *flexibility, evolvability, resilience*, and *adaptability*. The problem is that there are no precise and shared definitions of these terms, nor any evidence of convergence on agreement; and lack of agreement causes major problems in projects with requirements for such properties.

Ross et al. studied the long history of efforts to properly define such terms, and concluded that trying to impose definitions (typically carefully crafted natural language definitions) in a top-down manner, in hopes that diverse communities will rally around shared definitions, has proven unsuccessful and is unlikely to succeed in the future. They proposed as an alternative to develop a *semantic approach* to giving meaning to such terms.

Instead of making *ex cathedra* pronouncements about *preferred definitions* that *shall be used* (but aren't), they proposed a new idea. They started by studying change-related requirements statements that arise in practice. They created a *template* that to capture common structures and dimensions of variability in such statements. Finally, they defined rules for classifying how statements using this template map to *ility* labels. The language of changeability requirements is thus partitioned into (not necessarily disjoint) sublanguages under the various ility labels. It can also be said that they formulated syntax-based rules for *classifying* change requirements.

As an example, a requirement calling for an agent *external* to a system to be able to effect a given change would have a different set of ility labels than one calling for an agent *inside* the system boundary to be able to make a change: perhaps including *flexibility* as opposed to *adaptability.* Ility names are thus given meaning not in the form of top-down definitions but rather as syntactically and semantically partitioned classes of requirements statements.

The *template* of Ross and Rhodes was presented as a spreadsheet. Columns correspond to dimensions of variation in change requirements statements (e.g., whether a change agent is inside or outside the system boundary). The sequence of columns encode the structure of a requirements statement. In each row, columns are marked with one of several possible values (such as *inside* or *outside* in the column for where the change agent is situated relative to the system boundary). Some columns can be left blank. There are also free text fields in which the domain-specific contents of change requirement statements are expressed. The overall set of selection in a row form a pattern with an associated *output column* containing an ility term, such as *flexibility* or *adaptability.* The spreadsheet thus encodes, albeit imprecisely, a set of rules for classifying change-related requirements statements (cast using the template) as having one or more ility labels.

We observed that this informal and incomplete idea could be recast in formal computational terms. Instead of a template presented in a spreadsheet, we could defined a context-free grammar as a set of inductive data types and constructors. Given this formal language, we could then define semantic mapping functions associating any sentence in the language with associated ility labels (a syntax to semantics mapping, or *interpretation*).

Benefits include the ability to precisely reason about, *evaluate*, and *improve* the language grammar; fully mechanical checking of the *completeness* of the semantic rules; ending up with *a mathematically precise theory* rather than just prose and diagrams; having a basis for *automated synthesis* of a software implementation and globally accessible web service that exposes the theory to easy use and evaluation (rather than a model on paper); and having a theory that could be *evolved easily* as experience using it accumulates.

An experimental test of the adequacy of the expressiveness of the proposed language, we tested its ability to state change-related requirements in a seminal early paper in software engineering that focused on software evolvability as a critical property (the famous 1972 paper of David Parnas on information hiding modularity). We found the proposed language *not* yet adequate to satisfactorily capture such requirements. One of the key missing elements of the language is an approach to expressing systems models against which change-related requirements are being stated.

This work, to date, will be described in a paper that is to be submitted to CSER 2015.

Plans for 2015. Our plans for 2015 depart from the insight that it's really not possible to reason generally about ilities absent some kind of system modeling framework -- and that frameworks will vary from domain to domain and from ility to ility. We plan to develop new approaches to incorporating domain-specific systems modeling frameworks into both our domain-specific and general system ility modeling methods. We will drive this work from Ross's model as a starting point.

Such a system modeling framework is missing In Ross's current proposal. The model of the system about which change-related requirements are being stated is just implicit in the

requirements statements. More meaningful statements of requirements as well as any serious validation of systems as satisfying such requirements demands *some* approach to making system models explicit. Classical reliability is modeled in terms of fault trees, for example. Change-related ilities require their own system modeling approach(es). Design Structure Matrices, and our own *Design Impact Graphs*, are possible frameworks for modeling certain change-related ilities.

This work will extend Ross's language to make the language parameterized by a *type* of system models. Change-related requirements will be expressed in terms of (time-sensitive) state changes in such system models. We will explore this idea in a concrete way using what we call *design impact graphs* (DIGs), closely related to design structure matrices (DSMs) as a system model. Our synthesized web services will then support the expression of both system models of a kind appropriate in a given domain, as well as change-related requirements in terms of such models. We will then generalize from this particular case study to test the broader feasibility and utility of the approach.

We believe that these ideas can help to resolve a significant tension in the RT-113 project: it's hard to imagine a deep, useful theory of ilities and tradeoffs divorced from underlying system models. Such models need not be highly specific, but there must be some model. As another example, tradeoffs between fault-tolerant data availability, latency, and reliability in the face of possible network partitions requires something like a *quorum model*, in which one states the number of replicas that need to be written and the number of replicas that need to be read to get a value from a replicated store.

We plan to develop and evaluate a *generic* approach leveraging both parametric and ad hoc *polymorphism* (using e.g., typeclasses in Coq) to provide a common framework for modeling ilities against unrelated types of system models.

Activity #3: A Formal Framework for Value-Driven Tradeoff Analysis

The goal of this task is to expand a formal framework for automated design space synthesis and ility tradeoff analysis to link these technical issues to stakeholder value and stakeholder value reconciliation functions. This work is based in part on a paper published in ICSE 2014 on design space synthesis, ility measurement, and tradeoff analysis, but that did not include stakeholder value as a concern. The expected benefits of this task include the provision of both a mathematically precise framework and corresponding web-based technology, as per our earlier efforts. We will eventually produce a theory and technology to support modeling and analysis of multi-stakeholder value scenarios in which requirements and designs must be negotiated to produce satisficing outcomes for all success-critical stakeholders, as mediated by the suites of ilities that particular decisions are projected to produce.

Plans for 2015. We propose to develop a general but precise and easily specialized formal theory of linkages between incomplete specifications, design spaces whose elements vary in

terms of under-specified ilities, ility estimation functions on these spaces (domain-specific), tradeoff analysis functions, multi-stakeholder ility-to-value analysis, and stakeholder value reconciliation functions.

The core of this work will be a polymorphic, typeclass hierarchy expressed in Coq, flexible in terms of specification types, implementation types and design spaces, ility measurement functions, tradeoffs, and stakeholder value. The theory mean to (1) define the necessary constructs for scientific ility tradeoff and stakeholder value impact and reconciliation analysis; (2) provide a basis for synthesizing executable software frameworks specialized to specific domains, on top of which a broad range of domain-specific ility, tradeoff, and stakeholder impact analysis could be built; (3) support the provisioning of evolving web services to drive researcher and practitioner engagement with and evolution of the theory and technology.

The approach is partially validated based on a special-case proof of concept in one particular area of software engineering. This validation shows that an executable form of the framework can be instantiated to support specifications expressed in relational logic, design synthesis using relational logic model finders, ility estimation/measurement using a variety of published static and custom dynamic evaluation functions, and tradeoff analysis.

Activity #4: The Docility Index of Ilities, Tradeoffs, and Technologies

In work to date we have prototyped a database-backed, web site for cataloguing definitions of ilities, and underlying models of ilities and tradeoffs among them. It is intended to serve as a knowledge base providing researchers and practitioners easy access into the growing body of formal theory and theory-based technology for understanding, specifying and reasoning about ilities, tradeoffs, and stakeholder value. As a proof of concept, we populated the knowledge based initially with high-level tradeoff information taken directly from Boehm's proposed 8x8 grid showing synergies and conflicts between pairs of ilities.

Plans for 2015. Our most important goals for 2015 are to further develop the foundations of a science and engineering technology base to empower systems engineering to communicate and reason unambiguously, rigorously, affordably, and effectively about design spaces, ilities, tradeoffs, and stakeholder value. We will continue to work closely with our project member colleagues at MIT to develop far more precise methods for developing, communicating, disseminating, and validating theoretical work, through a combination of formal methods and software synthesis. We will also continue to work with our project colleagues at USC to bring the benefits of work prototyped with MIT to the task of formalizing a far more comprehensive range of ilities and tradeoffs.

The Docility tool was initially envisioned as a repository of definitions, propositions about generalized tradeoffs among ilities, and supporting evidence. While that aspiration remains easily feasible, we now see that it will be possible to populate it not only with English language definitions and tradeoff information, but with mathematically, computationally, and logically

precise models linked to reference implementations that make these theories accessible for community-based critique, evolution, validation, and eventual use. We plan to modify the tool to demonstrate this idea, for theories including our evolving theories of change-related ilities and ility-driven stakeholder value tradeoffs.

3.3.3 Overall Plans for 2016

Our plans for 2016 are focused on consolidating, disseminating, and validating gains made in preceding phases. We will produce versions of Docility populated with formally defined ility models and alternatively with informally defined ility definitions and tradeoffs based on USC research. We will demonstrate a suite of formally-defined ility terms, models (theories), and reference implementations, including validation of these models against realistic needs of industry and government.

This work is leading to an improved understanding of some specific ilities, particularly related to change-related requirements, but also to general approaches to building both a science and collection of interoperable, reference modeling and analysis services by which the science can be transitioned to the broader research community and to practitioners. As a side benefit, this work is providing one possible foundation for tool and method interoperability going forward.

4 WAYNE STATE UNIVERSITY

4.1 INTRODUCTION

Wayne State University research in 2014 was focused in two areas. One area was a "deep dive" into the needs and context of tradespace and affordability analysis in DoD system acquisition, in collaboration with the US Army, TARDEC. The second area was enhanced Set-Based Design (SBD) methods and applications with rigorous formulation and analytics, addressing challenges identified by NAVSEA in their trial applications of SBD methods. The SBD research is being coordinated with NAVSEA.

We coordinated with the TARDEC and NAVSEA end-users to articulate the key research issues and research objectives. We developed the technical approaches addressing the solution concept, research approach and the sequence of incremental steps. We reviewed the technical approaches with the end-users. We identified data needs and sources for practical demonstration and/or piloting the methods in end-user relevant application context, and began collecting the data. We began detailed research and development of the theoretical foundations, and practical methods, tools and procedures (MPT).

In 2015, we plan to continue research in both of these areas, and continue collaboration with the TARDEC and NAVSEA end users. By the end of 2015 we plan to have completed an initial demonstration of key MPT components in both of the research areas.

4.2 "DEEP DIVE" INTO TRADESPACE AND AFFORDABILITY NEEDS AND CONTEXT IN DOD SYSTEM ACQUISITION

During ground system conceptual design and engineering development, tradeoff decisions are made to balance functional performance, cost, risk, robustness, reliability, interoperability, growth margin, and other life-cycle characteristics. Tradespace analysis tools are needed to reveal and quantify the compound and ripple effects and interactions among performance requirements and system design options. The objective tradespace analysis tool is an executable model that represents the logical and causal relationships between and among subsystems and system performance characteristics. Logical relationships define compatibility requirements between interacting subsystems. Causal relationships quantify higher level system attributes as functions of lower level subsystem attributes.

Research at the US Army, TARDEC, has been developing a generic ground system performance specification framework, and a ground system standard product classification hierarchy. The current research seeks to integrate these two frameworks with a unifying framework of the logical and causal relationships. The tradespace framework identifies the relevant subsystem attributes and how they interact to govern product performance and life-cycle characteristics, and identifies the subsystem interfaces and compatibility characteristics that constrain and enable practical ground vehicles. The objective is a qualitative and quantitative framework

integrating system architecture and technology options with functional and life cycle performance to support design and performance requirement decisions. The goal is to integrate the ground system architecture and performance specification frameworks for interactive tradespace analysis and exploration. The research is motivated by, and grounded in, an understanding of the performance and affordability tradespace decisions and supporting analysis in ground vehicle development programs. Details of this research are contained in the attached report "Progress Toward a DoD Ground Vehicle Tradespace and Affordability Analysis Framework."

4.3 ENHANCED SET-BASED DESIGN

The research on enhanced SBD is organized into two areas.

The first area is to enhance SBD as a design and development process by developing a rigorous, principled and practical formalism, by developing a formalization of the notion "regions of design space" with useful properties for design enabling new, practical and relevant tradeoff analysis questions to be addressed, and by incorporating decision timing, change cost and uncertainty into the formalization. At the present time, SBD is a conceptual methodology without rigorous analytic formulation. This research attempts to develop this foundation. Computational design methods enable developers to automatically generate millions of possible combinations of design options and evaluate them on multiple performance and life cycle dimensions. This creates a conceptual challenge to the developers – to organize the millions of point designs into a simplified and useful view of design space supporting the SBD process. This research is developing a rigorous and useful approach to organize design space into regions, and that provide new analysis methods providing insight into the effects of system requirements on the topology of the design space regions. The research is also developing a structured approach addressing the sequence of design decisions and the interplay with the maturation and integration timelines of subsystem options. Details of this research are contained in the attached report "Design Space Regions, Geography and Topology for Set-Based Design."

The second area is to adapt the concept of SBD for long-lived DoD systems. In the traditional view, SBD leads to production point design. Long-lived DoD systems are expected and planned to undergo multiple upgrades and spawn variants over their lifespan. In this situation, the objective of system design is not simply the production design but is the production design with the set of affordable potential future upgrades and variants to meet changing operational needs and to exploit technology opportunities. Applying the concept of SBD to the continued development after the initial production design provides a methodology for tradespace and affordability decision making to explicitly consider affordability and tradespace not just of the initial design, but also of the potential family of future upgrades and variants. This provides a methodology and conceptual framework to address real issues in developing the product specification and making tradeoff decisions during development: How much reserve capacity

(aka “design margin”) should be engineered into the system to increase the range of affordable future upgrade capabilities? How can tradeoffs between modular and integral design be made, to balance current known and uncertain future needs and opportunities? When should specific capabilities or technologies be deferred to future upgrades? Details of this research will be contained in a future report “Set-Based Design for Long-lived Systems, Upgrades and Variants.”

4.4 Report “Progress Toward a DoD Ground Vehicle Tradespace and Affordability Analysis Framework.”

Paper accepted for the 2015 Conference on Systems Engineering Research written By Gary Witus, Walter Bryzik

During ground system conceptual design and engineering development many tradeoff decisions are made to balance functional performance, cost, risk, robustness, reliability, interoperability, growth margin, and other life-cycle characteristics. Tradespace analysis reveals and quantifies the compound and ripple effects and interactions among performance requirements and system design options. An analytic tradespace framework is an executable model that represents the logical and causal relationships among vehicle subsystems and system performance characteristics. Logical relationships define compatibility requirements between interacting subsystems. Causal relationships quantify higher level system attributes as functions of lower level subsystem attributes. Research at the US Army, TARDEC, has been developing a generic ground system performance specification framework, and a ground system standard product classification hierarchy. The current research seeks to integrate these two frameworks with a unifying framework of the logical and causal relationships. The tradespace framework identifies the relevant subsystem attributes and how they interact to govern product performance and life-cycle characteristics, and identifies the subsystem interfaces and compatibility characteristics that constrain and enable practical ground vehicles. The objective is a qualitative and quantitative framework integrating system architecture and technology options with functional and life cycle performance to support design and performance requirement decisions. The goal is to integrate the ground system architecture and performance specification frameworks for interactive tradespace analysis and exploration. The research is motivated by, and grounded in, an understanding of the performance and affordability tradespace decisions and supporting analysis in ground vehicle development programs.

Nomenclature

□	side slope rollover threshold
w	lateral wheelbase
h	height of the Center of Gravity
m	unsprung mass
k	suspension stiffness
T	suspension travel

4.4.1 INTRODUCTION

Different decisions and tradeoffs are made at the different stages of the DoD system acquisition life-cycle1. The tradespace evolves through the development life-cycle. Prior to the Material Development Decision, decisions and analysis are concerned with whether or not material development is needed, if so, the type of system, the missions, general capabilities, high priority characteristics, and variant configurations. Material Solution Analysis considers alternative acquisition strategies, e.g., upgrading an existing system, modifying a commercial system, or a new start. Material Solution Analysis assesses cost and feasibility of meeting the needs with alternative high-level system concepts leading to selection of a high-level design concept and a preliminary set of high-level performance specifications. These are the basis for the Technology Development (TD) stage. The purpose of the TD stage is to refine the requirements and design concept to ensure balanced, affordable, effective and low risk performance specifications and design concept for subsequent Engineering and Manufacturing Development (EMD). The specifications define constraints – the range of operating conditions, states and modes – and performance requirements for mission functions and life-cycle characteristics.

The TD stage develops a preliminary design. The TD stage also adjusts the performance specifications to balance time, cost, risk and capability. The specifications are refined and detailed in parallel with developing the design concept. As the design concept is developed, engineers are better able to assess the feasibility, cost and risk of meeting the specifications. It may not be possible to meet all the initial performance specifications with the evolving design concept. TD then needs to consider relaxing the specifications or changing the concept. Ideally, concept development decisions are informed by a thorough understanding of the effects on performance, and implications for the remaining design tradespace. The goal of the TD phase is to develop a balanced set of performance specifications matched to a conceptual design. The TD tradespace is the combination of the performance specifications and the conceptual design.

The focus of this research is on tradespace analysis during TD. Tradespace analysis requires an understanding of the elements of the performance specifications, the elements of the conceptual design, their properties, and the relationships among them.

The tradespace framework is an executable knowledge structure that identifies the elements, their properties, the logical and quantitative relationships among them. An executable model of the logical tradespace relationships identifies the path of ripple effects of decisions, and interdependencies among subsystem elements and system performance attributes. An executable quantitative tradespace model provides further capability to assess performance margins, sensitivities to preliminary design choices, and interactions among system elements – including ripple effects across the network of relationships. It can also help identify where relative valuation or prioritization among performance specifications can resolve ambiguities in subsystem valuation and performance choices.

This paper illustrates the practical context and motivation with a real-world cost and tradespace analysis from a current tactical vehicle program. It describes and illustrates the emerging

framework, addressing the performance specification framework, the ground system architecture model, the system and subsystem attributes, and the logical and quantitative relationships among them. It describes how the tradespace and affordability framework supports conceptual design and requirements balancing during the TD phase.

4.4.2 CONTEXT AND MOTIVATION

This section summarizes tradespace and affordability analysis conducted during the last year of the TD stage of the Joint Light Tactical Vehicle (JLTV)[2]. The tradespace and affordability analysis took place at a critical juncture in the development program, and was instrumental in successful Milestone B decision review to proceed to the EMD stage of acquisition. The tradespace and affordability assessment was conducted to provide the information and options needed to support program development, at the level of design granularity and modeling accuracy needed to support program justification. It was a pragmatic activity, not a theoretical exercise, and as such it both informed and motivated the ground vehicle tradespace framework.

The analysis was conducted under time pressure, and therefore required the analysts to use their best engineering judgment regarding what subsystem alternatives to consider, which performance characteristics to include in the analysis, which to address quantitatively and which to address qualitatively, and which causal chains and "ripple effects" to consider in the analysis. Quantitative analysis was conducted using parametric models of ground vehicle performance developed by the US Army and automotive engineers to predict ground vehicle mobility performance as functions of vehicle configuration and design attributes[3,4,5]. The parametric models are compatible with the assumptions of the NATO Reference Mobility Model standard for operational mobility effectiveness analysis, but are higher resolution as appropriate for design and configuration performance assessment. The parametric models identify the design attributes relevant to performance, and the analytic equations.

This example illustrates the types of considerations in practical tradespace and affordability analysis, and motivates the development of the ground vehicle tradespace and affordability analysis framework. The review of the JLTV cost-versus-performance tradeoff assessment elucidates the capacities and characteristics needed in a practical and relevant generic grounds vehicle tradespace analysis framework by addressing several questions:

1. What were the basic questions that frame tradespace and affordability analysis in the TD phase?
2. What types of quantitative tradeoff analyses were performed and what level of detail?
3. What types of factors were addressed qualitatively, and why?
4. What tradeoffs and impacts were not considered, but could be addressed in the tradespace model?

In the period from 2011 to 2012 the JLTV program was concluding its TD phase. The assumptions entering TD phase were that the performance requirements were firm, testing on competitive prototypes would show that performance requirements were achievable, and that

the production units would be affordable. As the program entered the final year of TD, operational testing on prototypes from different development teams showed that meeting all the requirements was challenging. Furthermore, the cost-tolerance had changed, lowering acceptable average unit production cost. In response, the JLTV program began a disciplined and systematic "cost-informed trades assessment" to reduce cost and rebalance the performance requirements to reflect affordability and relative priority. The basic questions were:

- What are the subsystem alternatives that could result in significant production cost reduction and by how much?
- What are the major performance characteristics that would be impacted?
- How much would the performance requirements need to be reduced to admit the alternatives?
- What are the other, unquantified, implications of the subsystem alternatives?
- What are the adjustments to the design concept needed to accommodate alternatives?
- What are the combined cost savings and performance compromises for packages of subsystem options?

The high cost subsystems were believed to offer the greatest potential for cost reduction. There was an initial hope that reduced cost could be achieved by trading off a single performance requirement, but that did not prove feasible since all of the high-cost subsystems impacted multiple performance characteristics. They affected mobility, occupant protection, payload capacity, system survivability, reliability, transportability, and net-readiness.

The high cost subsystems were propulsion/power-pack, hull/frame, auxiliary automotive electrical power, integration & assembly, and suspension/steering. Integration & assembly was removed from consideration because this would be required in any design, and was not a design decision parameter. Two of the subsystem trade studies are described below.

4.4.2.1 Engine Tradeoff Analysis

The engine is the high cost item within the propulsion/power-pack. Three engine alternatives were considered. All engines were mature and available.

The mobility characteristics chosen for quantitative analysis were top speed, speed-on-grade, and soft-soil mobility. Speed-on-grade was considered only at 5-percent grade, not across the full range of grades in the performance specification. Soft-soil mobility was addressed in terms of the minimum rating cone index (RCI) soil that the vehicle could traverse. The vertical step climb requirement was addressed qualitatively. Operation and sustainment (life-cycle) characteristics fuel consumption and reliability were assessed qualitatively.

The quantitative tradeoff analysis considered only the cost and horsepower attributes of the engine alternatives. The assessment did not address the impact on of engine size on cooling requirements or the cooling system. The effects of differences in engine size and weight were

not addressed. Relative to the mass of the rest of the vehicle, the effect of differences in engine weight on total mass and Center of Gravity (CG) location were deemed negligible from a mobility perspective. CG location is a key factor in vehicle stability. Engine size impacts open space in the engine compartment, affecting cooling and maintainability. The assessment did not address the impact engine size and weight on maintainability.

The quantitative tradeoff analysis results were summarized in a table showing cost and expected mobility performance for the three engine alternatives. The qualitative tradeoff analysis findings were included as side notes.
Qualitative assessment noted that lower power engines could not meet the original vertical step climb requirement, but did not estimate the vertical step climb ability at reduced power. Qualitative assessments noted that smaller engines consumed less fuel, and are less reliable because they run hotter for comparable load.

4.4.2.2 Suspension Tradeoff Analysis

The tradeoff options were between an active pneumatic suspension and traditional passive suspensions. The mobility characteristics chosen for quantitative analysis were ride quality and transportability. Ride quality is a limit on speed as a function of terrain – the maximum speed at which crew station absorbed power (vibration) and acceleration (shock) would be within human tolerance limits, as a function terrain roughness and obstacle height across the weight range from curb weight to gross vehicle weight. The analysis qualitatively assessed growth capacity for increased vehicle weight with future mission equipment and "margin" for emergency overloading. The impact of suspension choice on reliability of other subsystems was qualitatively addressed, but the reliability, availability, and maintainability of the suspension itself was not addressed. The analysis did not consider the impact of suspension choice on cornering stability, side-slope stability and hill-climb grade, under the range of CG locations for different variants, payload configurations, and applique "B" armor kits.

Active pneumatic suspension provides the ability to adjust the suspension stiffness to accommodate different vehicle weight and terrain conditions. Passive suspension is point-designed for a nominal weight and ride quality degrades away from the design point. Passive suspension could not meet the ride quality requirements (speed on terrain) across the full range from curb weight to gross vehicle weight. The quantitative analysis did not address how much the speed-on-terrain requirement would have to be reduced for passive suspension to meet ride quality requirements over the full range from curb weight to gross vehicle weight. The analysis did not address suspension travel. Suspension travel interacts with suspension stiffness to affect ride quality when the suspension displacement hitting the hard stops.

Transportability requirements included that the vehicle fit within the entry/exit space of the transport craft, and can traverse the transition from ramp to flat-bed without bottoming-out. Active suspension provides the ability to squat and to adjust fore-aft ground clearance, which are needed for transport on Maritime Prepositioning Force (MPF) ships. Passive suspension would fail to meet MPF transport requirements. The ability to squat also improves survivability by enhancing defilade posture, but this benefit was not addressed in the trade study.

Passive suspension is current technology. Active pneumatic suspension is not currently in the fleet and the logistics for maintenance would be a new burden. Passive suspension has lower cost, but higher weight. Active suspension has lower reliability. Active suspension can, potentially, reduce the shock and vibration on other components and interfaces, thereby increasing their reliability.

Passive suspension requires a decision regarding stiffness, which involves tradeoffs between on-road and off-road mobility. A softer suspension provides better ride quality and better handling on rough terrain, but a stiffer suspension provides better stability, handling and agility on road. An active suspension allows the stiffness to be adjusted to suit the vehicle weight and maneuver conditions. An active suspension also can improve side-slope stability and hill climb stability by stiffening the suspension on the downhill side and softening the suspension on the uphill side, although this benefit was not addressed.

Qualitative assessment noted that passive suspensions were heavier than the active suspension. But the implications of increased unsprung mass and total mass on other performance characteristics were not examined, e.g., increasing the unsprung mass increases fuel consumption on rough terrain, and increased shock to the unsprung components.

4.4.3 Ground System Tradespace and Affordability Analysis Framework

The major elements of the ground system tradespace and affordability analysis framework are (1) the performance specification framework, (2) the ground system decomposition, (3) the interface relationships among the subsystems, and (4) the dependency relationships of the higher-level system attributes on the lower-level subsystem attributes. The performance specifications are the top system-level attributes. The lower-level attributes from the logical and quantitative relationships.

4.4.3.1 Performance specification Framework

The performance specification (P-Spec) expresses the requirements for the system as a whole; it defines the performance characteristics of interest, and the specific levels required in the objective systems[6]. The P-Spec is a hierarchical organization of system characteristics that includes key performance parameters, key system attributes, and additional performance attributes. These are further decomposed into derived system requirements. The P-Spec specifies what the system does, at what level it has to perform, what constraints and conditions it must perform under, and what its life-cycle support properties are required.

The ground vehicle P-Spec framework being developed at TARDEC defines the performance attributes of the ground vehicle system but does not contain program-specific levels. The ground vehicle P-Spec framework is being developed by generalizing from specific P-Specs for a spectrum or vehicles. The activity is part of a knowledge-based systems engineering initiative to capture domain knowledge for future re-use. For any particular system, some of the generic P-Spec framework fields may be “not applicable”.

Figure 8 shows a portion of the organization of the ground vehicle P-Spec hierarchy. The performance specification framework defines data that quantifies the specifications, e.g.:

- Ride quality absorbed power requirements are expressed as a table of speed versus root-mean-square (RMS) terrain roughness at which the crew/passenger absorbed power must be to be less than or equal to 6 watts over the range from curb weight to gross vehicle weight
- Ride quality transferred shock requirements are expressed as a table of speed versus radius of "half-round obstacles" at which the crew/passenger acceleration must be less than or equal to 25 m/s/s over the range from curb weight to gross vehicle weight
- Cornering stability requirements are specified as a table of speed versus turning radius at which the vehicle will not roll over on hard, flat, level terrain
- Side-slope and hill-climb stability requirements are specified as a slope at which a stationary vehicle will not roll over or pitch backward, and can start with 10-percent fuel supply and can initiate motion
-

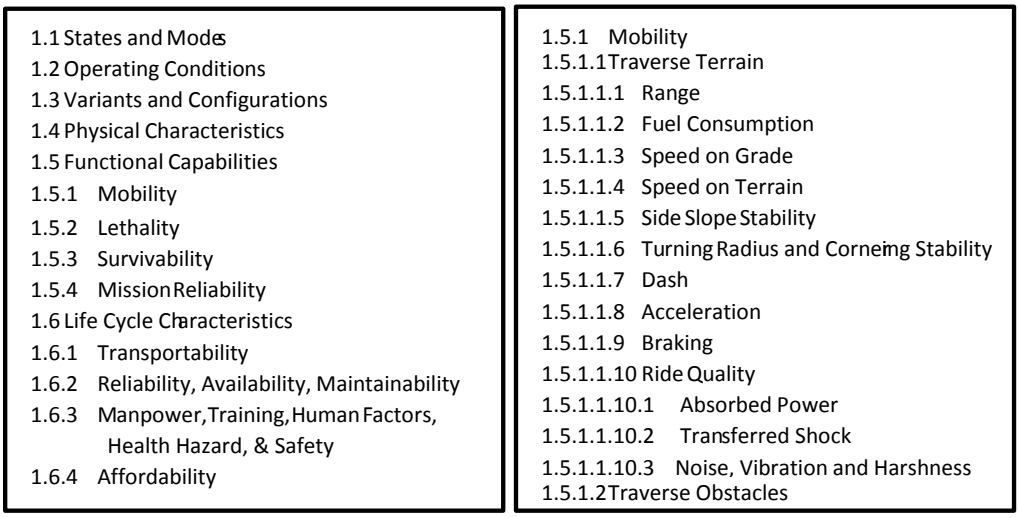


Figure 8: Partial Sample of the P-Spec System Attributes

The generic P-Spec is substantially mature, but is still evolving. Additional performance characteristics may still be added, and the particulars of performance specifications may be refined reflecting improved understanding of the relationship between vehicle design parameters and operational effectiveness.

Ride quality specifications do not currently address motion-induced sickness. Motion sickness is induced by low-frequency disturbance. Low frequency disturbance occurs at low-speed in vehicles with soft suspension and high moments of inertial, and is amplified by rough terrain. Motion-induced sickness is especially a concern for troop transport vehicles such as the Marine Personnel Carrier and the Armored Multi-Purpose Vehicle, and can result in significant degradation of troop ability upon dismount.

Stability requirements have traditionally been specified for idealized conditions, but not for realistic operational or limiting conditions. Off-road rollover events are caused by a combination of terrain (intermittent bumps and potholes, soft soil patches, slippery patches, etc.) on slopes, with steering, acceleration and braking. Future stability specifications might combine speed, turning radius, slope and terrain characteristics into a dynamic stability requirement, similar to the ride quality requirement tables.

4.4.3.2 Ground System Architecture

The ground system architecture framework[7] is being developed at TARDEC by generalizing from specific architectures of a wide variety of vehicles and following systems and software engineering architecture description guidelines[8]. The activity is part of a knowledge-based systems engineering initiative to capture domain knowledge for future re-use. The ground system architecture includes a hierarchical decomposition of the system into subsystem segments, referred to as the "Standard Product Classification Hierarchy" (SPCH). The SPCH maps onto the generic ground vehicle work breakdown structure in MIL-STD-881(C), but with more detail. The SPCH decomposition stops at the level at which components are purchased as integral items, and not designed as part of the ground vehicle development. Figure 9 shows a portion of the SPCH. The architecture is generic, and any given system may not have all of the elements.

01	Family of Vehicles
01.01	General Purpose (GP) Base Vehicle Platform
01.01.01	Hull / Frame
01.01.01.01	Frame or Clips (front &/or rear) Systems
01.01.01.02	Primary Structure and Base Armor
01.01.01.03	Bumpers & Fascias Systems
01.01.01.04	Transparent Armor
01.01.02	Suspension / Steering
01.01.02.01	Suspension System
01.01.02.02	Brake System
01.01.02.03	Central Tire Inflation System (CTIS)
01.01.02.04	Tires / Wheel Systems
01.01.02.05	Steering System
01.01.03	Power Package/Drive Train
01.01.03.01	Engine
01.01.03.02	Cooling System
01.01.03.03	Transmission / Transfer Case / Differential system
01.01.03.04	Power Generation Systems
01.01.04	Auxiliary Automotive
01.01.04.01	Fuel Systems
01.01.04.02	Heating, Ventilation, & Air Conditioning (HVAC)
01.01.04.03	Vehicle Controls
01.01.04.04	Automated Fire Extinguishing Systems (AFES)
01.01.04.05	Vehicle Diagnostic Systems & Software
01.01.05	Turret Assembly
01.01.06	Fire Control
01.01.07	Armament
01.01.08	Body / Cab

01.01.03.01	Engine
01.01.03.01.01	Engine Assembly-internal combustion
01.01.03.01.01.01	Crankcase-Block-Cylinder Head Assembly
01.01.03.01.01.02	Crankshaft
01.01.03.01.01.03	Flywheel Assembly
01.01.03.01.01.04	Piston and Connecting Rod
01.01.03.01.01.05	Valves Camshafts and Timing System Assembly
01.01.03.01.01.06	Engine Lubrication System
01.01.03.01.01.07	Engine Starting System-Other than electric
01.01.03.01.01.08	Manifold
01.01.03.01.01.09	Diesel Starting Control and Conversion Unit
01.01.03.01.01.10	Bearing or Shaft
01.01.03.01.01.11	Engine Brake
01.01.03.01.02	Engine Assembly-Gas or Steam Turbine
01.01.03.01.02.01	Compressor Assembly
01.01.03.01.02.02	Combustion Assembly
01.01.03.01.02.03	Turbine Assembly
01.01.03.01.02.04	Regenerator Assembly
01.01.03.01.02.05	Accessory Drive Assembly
01.01.03.01.02.06	Fuel or Steam Control
01.01.03.01.03	Lubricating System

Figure 9: Partial Sample of the Standard Product Classification Hierarchy

The SPCH includes both the subsystem decomposition elements and technology alternatives. It is not a simple hierarchy. It has both "AND" and "OR" nodes. At an "AND" node, all of the child nodes are elements of the parent node. For example, a power package consists of an

engine AND a cooling system AND a transmission AND an electrical power generation system. At an "OR" node, a subsystem branches into alternative technology options. For example, an engine could be diesel OR turbine OR hybrid-electric. Only one of the OR options is included in the final design.

The current version of the SPCH identifies some of the components' relevant attributes. The relevant attributes are those attributes that affect system performance and life-cycle characteristics, and those attributes that determine compatibility with other components. For example attributes of an engine include cost, rated horsepower, weight, size, low-end torque, and reliability. The attribute portion of the framework is still under development.

The values of the attributes cannot be chosen arbitrarily, but are constrained by technology to be within a practical range and relative values. Sometimes, there are a specific set of options, e.g., engine alternatives A, B and C. Sometimes there are a range of possible attribute values. The values of attributes may be correlated, and the correlation may be different for different technology alternatives. For example, for diesel engines in the power range suitable for a tactical truck, cost, horsepower, volume, and weight are close to linearly related. For turbine engines, cost, horsepower, volume, and weight are also close to linearly related, but with different slopes and intercepts than diesel engines.

4.4.3.3 Relationships and Attributes

There are two types of relationships: logical relationships between system architecture elements and quantitative relationships among the attributes of system architecture elements. The subsystem attributes are those characteristics involved in the logical and quantitative relationships.

Logical relationships address subsystem interfaces and compatibility. When there is an interface between two subsystems, a change to one may require a change to the other to maintain compatibility, or may open up options for lower cost alternatives, or may require a change to the interface. The logical relationships identify the design implication ripple effect pathways to consider in tradespace analysis.

The logical relationships describe the "topology" of the system. Logical relationships identify which subsystems interface with which other subsystems, and the type of interface, e.g., structural, thermal, electrical, hydraulic, data, etc. The logical relationships are developed by generalizing from the subsystem and component interfaces identified in maintenance manuals and design diagrams for representative systems. Figure 10 illustrates a portion of the vehicle topology surrounding the engine, noting the different types of interfaces. Different types of interfaces have different characteristics. For example, mechanical interfaces have attributes of yield strength, elasticity, damping, deadband, losses, excursion limits, in each degree of freedom. Other types of interfaces have analogous properties.

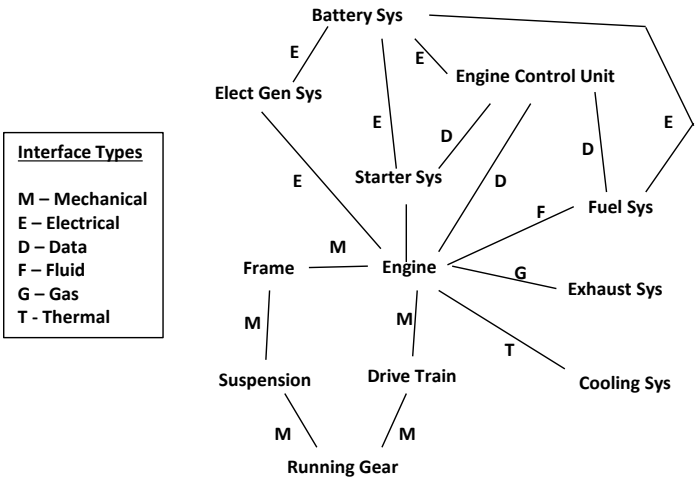


Figure 10: Example Subsystem Interface Diagram

Quantitative relationships express how the values of the attributes of lower-level system elements determine the performance attributes of higher-level system architecture elements. The network of quantitative relationships is necessary to estimate the impact on the system level performance of changes in the values of components' attributes. Subsystem attributes affect multiple system performance attributes, e.g., active suspension provides better ride quality and stability than a fixed passive suspension, but costs more and is less reliable.

The quantitative relationships are engineering approximations that have been developed over many years for different types of vehicles and vehicle technologies to support conceptual design and assessment[3,4,5]. They are parametric relationships suitable for representation with SysML parametric diagrams or similar tools. For example, with a passive suspension, static side slope stability, is a function of unsprung mass, height of the CG, lateral wheelbase, suspension stiffness, suspension travel (1). The static side slope limit is the angle at which the CG is directly above the down-slope wheel contact line due to slope and suspension compliance as the load of the unsprung mass shifts to the down-slope side. With an active suspension, the same physics is in effect, but the system stiffens the down-slope suspension and softens the upslope suspension to counteract the slope effects.

$$\theta = \operatorname{atan}\left(\frac{w/2}{h}\right) - \operatorname{atan}\left(\frac{w}{\min(T, m/k)}\right) \qquad (1)$$

The logical and quantitative relationships determine the relevant attributes of the system architecture elements. The relevant attributes of a system architecture element are the ones that are used in the equations describing the higher level performance parameters, and compatibility among subsystems. The attributes and quantitative relationships are developed by working top-down from the system-level attributes identified in the P-Spec framework.

4.4.4 TRADESPACE ANALYSIS

The fundamental tradespace consideration is the performance margin. The performance margin is the difference between the performance expected from a given conceptual design and the performance requirement. Where the margin is negative, expected performance does not meet the requirements. Where it is positive, there is "margin for error," a buffer against unforeseen operational needs, subsystem shortfalls and interaction effects.

During TD, the Program Management Office (PMO) can adjust the performance requirements in the P-Spec to balance acquisition and operation cost, reliability/availability/maintainability, interoperability, transportability/deployability and mission function capability subject to confidence in technical feasibility. The preliminary design is an "existence proof" that the P-Spec can be met. The P-Spec is the entry specification for the EMD phase. Tradeoffs and tradespace analysis continues during early EMD up to the Critical Design Review. The tradespace analysis framework is a tool to explore these interactions.

The ground system tradespace and affordability analysis framework as described in this paper enables developers, designers and other stakeholders to consider the following types of questions, supported by logical, qualitative and quantitative analysis with the confidence that indirect and ripple effects have been included:

- Within the range of feasible, practical technologies and design concepts, what are the tradeoffs among functional performance and life-cycle management characteristics?
- How do different conceptual design alternatives influence functional performance and life-cycle management characteristics?
- How resilient is the design - does the concept design have sufficient positive performance margins in key dimensions to provide growth potential for future modifications, to provide safety margin in the event that future operating conditions are more severe than planned for, and to provide design margin for error in the event that subsystem and interface performance is less than expected?
- Which elements of the performance margin vector are negative? How much do the requirements have to be reduced to be achievable with the preliminary design? What are the implications on other requirements of design changes to reduce the negative margins?
- Are the performance margins near zero for critical requirements, i.e., is the concept at risk of failing to meet critical requirements?
- What is the sensitivity of the performance margin vector to individual and combined changes to different options if immature technologies at are considered as potential future upgrades?

4.4.5 CONTRIBUTION, SIGNIFICANCE, LIMITATIONS, AND EXTENSIONS

The research described in this paper represents an approach to formalize knowledge regarding system requirements, system components, their attributes, and interactions. The framework supports practical relevant trade-off analysis between cost and capability in ground vehicle development programs. It is informed by and tightly linked with tradespace and affordability

analysis issues and assessments in current ground vehicle development programs, and with knowledge capture initiatives to learn and generalize from past experience.

The ground system tradespace framework extends and unifies the P-Spec and ground system architecture frameworks currently in use. It supports the types of analyses used in the JLTV cost-informed trades assessment process. It makes future similar analyses easier and more complete by having the knowledge structures and parametric models in place. It enables a more complete tracing of "ripple effects." It represents the ground vehicle topology compatibility of subsystems that interface with each other, and to trace potential repercussions of design changes. It traces the network of interactions with quantitative estimates of system performance.

As structured, the tradespace framework is set up to support traditional point-based design methods. It presumes that credible data on the component attributes is available. Extension to the framework would be needed to address uncertainty in the levels of the component attributes. Extensions to the framework would also be needed to support incremental design methodologies such as set-based design (SBD). In SBD, high level and major component decisions are made without specifying lower level choices. SBD enhancements are needed to provide proof-of-feasibility of development options. In SBD there is uncertainty regarding conceptual design choices that have not yet been made. SBD is still evolving, and effective application to ground vehicle development programs is only theoretical at this time.

4.5 REFERENCES

1. Under Secretary of Defence for Acquisition, Technology and Logistics, *Operation of the Defence Acquisition System – Interim Instruction 5000.02*. Office of the Secretary of Defence. November, 2013.
2. Schultz S, Johnson BR. Cost Informed Trades Assessment and Requirements Management Process. *Proceedings of the 6th NDIA GVSETS Symposium*, August 12-14, 2014.
3. Baylot EA, Burhman QG, Green JG, Richmond PW, Goerger NC, Mason GL, Cummins CL, Bunch LS. *Standard for Ground Vehicle Mobility – ERDC/GSL-05-6*. US Army Corps of Engineers Engineering Research and Development Center. February 2005.
4. Gillespie TD. *Fundamentals of Vehicle Dynamics*. Society of Automotive Engineers. 1992.
5. Wong JY. *Theory of Ground Vehicles 4th Edition*. Wiley. 2008.
6. Defense Standardization Program. *Guide for Performance Specifications SD-15*. Department of Defense. August, 2009.
7. Fett D, Prichett W, Richardson J. The JCGV Ground System Architecture Framework (GSAF). *Proceedings of the 5th NDIA GVSETS Symposium*, August 20-22, 2013.
8. *International Standard 42010 - Systems and software engineering – Architecture description*. ISO/IEC/IEEE. December, 2011.

4.5 Report "Design Space Regions, Geography and Topology for Set-Based Design."

Authors: Gary Witus, Steve Horton Rapp

The goal of this research is to enhance SBD as a design and development process by developing a rigorous, principled and practical formalism, by developing a formalization of the notion "regions of design space" with useful properties for design enabling new, practical and relevant tradeoff analysis questions to be addressed, and by incorporating decision timing, change cost and uncertainty into the formalization. At the present time, SBD is a conceptual methodology without rigorous analytic formulation. This research attempts to develop this foundation. Computational design methods enable developers to automatically generate millions of possible combinations of design options and evaluate them on multiple performance and life cycle dimensions. This creates a conceptual challenge to the developers – to organize the millions of point designs into a simplified and useful view of design space supporting the SBD process. This research is developing a rigorous and useful approach to organize design space into regions, and that provide new analysis methods providing insight into the effects of system requirements on the topology of the design space regions. The research is also developing a structured approach addressing the sequence of design decisions and the interplay with the maturation and integration timelines of subsystem options.

4.5.1 Introduction

This is a preliminary, interim draft report on extensions and refinements to Set-Based Design. Section 2 describes the system development process context and issues, including requirements uncertainty, technology uncertainty, decision sequences, decision timing, cost and change-cost dependencies. Section 3 compares and contrasts Set-Based Design (SBD) and Point-Based Design (PBD) methods. Section 4 introduces a formalism for regions of design space, including computational methods. Section 5 describes how analysis of the geography and topology of regions of design space can be used to enhance SBD.

4.5.2 System Development Context

DoD system development nominally proceeds through the stages of Materiel Solution Analysis (MSA), Technology Development (TD), and Engineering and Manufacturing Development (EMD). These initial development stages are followed by Production & Deployment, Operation & Sustainment, and Retirement & Disposal. In the MSA, TD, and EMD phases the functional performance and life-cycle requirements for the system are developed and refined in parallel with developing and refining the system concept. The two are co-developed in order to assure that the requirements are feasible, mutually compatible, and cost-effective. Each stage increases resolution and completeness. At the beginning of the EMD stage, the requirements and system concept are expressed as a product specification (P-Spec) and associated generic system architecture. These are the basis for the Request for Proposal for EMD bidders. During EMD the P-Spec and system architecture continue to be detailed and refined to balance

requirements, affordability, technology and integration reality. In principle, the requirements and architecture become increasingly "sticky" along the way with refinements and adjustments taking place at the in the details more than the higher structure. In theory, they are "frozen" at the Critical Design Review.

In practice, the process is less smooth than in theory. Requirements are changed in response to external factors – changes in military strategy, perceived threats and theaters, changes in the opinions of senior DoD planners and decision makers, changes in the cost-vs-capability balance, changes in risk tolerance and risk tolerance as development and test information is developed, failure of component technologies to perform up to expectation, unexpected performance deficiencies or interference during integration, etc. Requirements are rebalanced and sometimes deferred.

DoD system development takes longer than anyone wants. Faster design and development processes can produce faster results, but do not necessarily do so. If slower processes can more quickly adjust to external changes, they can produce shorter development times in the presence of external changes. Processes that are faster when external changes are negligible, can have "long tails" if they involved extensive rework. There is a tradeoff of the likely development time absent external events, with the mean time and variance given external impacts.

The EMD program has a schedule of events leading up to Operational Testing, and has to proceed regardless of potential external events. Design decisions have to be made, designs developed tested and integrated. The design decisions are a partially ordered set, i.e., some decisions have to be made before others. For example, the choice of wheels versus tracks on a ground vehicle has to be made before the decision about the wheel configuration and suspension. Sometimes the timing of decisions is forced by the lead time to design and integrate particular technologies. For example, an active hydra-pneumatic suspension could have an 18 month lead time, whereas a passive mechanical suspension could have 4 month lead time. In this case, at 18 months from initial testing, program management has to decide whether or not to pursue the hydra-pneumatic suspension. The order of decisions can also have an effect on EMD time and cost. Consider a ship with ducting running from one zone to another. If the bulkheads are completed before the ducting, the ducting options may be limited and require cutting the bulkhead during manufacturing. If the ducting is solved first, then the bulkheads are designed and built around the ducting. The time and cost are different in the two cases.

When changes are made to the system concept, whether due to external events or discoveries during EMD, the design rework has an impact on time and cost. In general, the more complete a design is, the greater the time and cost impact will be – more subsystems and interfaces have to be adjusted, recalibrated, and retested.

4.5.3 PBD AND SBD

PBD is a design approach that tries to arrive at complete, specific design as quickly as possible. Once there is a concept or design, "customers" in the Service command, DoD, and Congress will think they know what they will be getting, and will be more confident that the Program Office knows what they are doing. Once there is a specific concept or design, cost estimates will have less uncertainty than before.

PBD is generally conducted by making design choices that limit the options that have to be considered, restricting the remaining design space. PBD generally tries to home in on a specific concept or design, confirm that it is technically feasible, affordable and meets or comes close to meeting the performance and lifecycle requirements. The requirements can often be relaxed or deferred as part of the tradeoffs of time and cost. During the process of homing in on a point solution, local tradeoffs and decisions are made to improve outcome on one or other dimensions.

PBD may make design decisions before they are required, to advance the process. PBD may make design decisions based on the expectation of future availability and performance of a technology, in order to advance the processes. This potentially exposes PBD to the risk of rework if the technology is not available on time or does not perform to expectations. There might have been information to make a more cost-effective decision, had the process waited until the last minute to make the selection.

SBD is a design approach that attempts to keep options open as long as possible, only making design decisions when they are required due to lead times and decision precedence. SBD attempts to minimize rework in response to external events and information by deferring commitment. The less rework there is, the faster and more economically the design can be adjusted in reaction to the external changes.

SBD operates on the principle of relaxing and restricting constraints (requirements) and making the minimal design decision, i.e. eliminating subsystem options from consideration. When a decision has to be made, the SBD process tightens constraints (allowing different stakeholders to have their input), until there is only one decision outcome that meets the constraints.

Both SBD and PBD apply at all stages of development, from high level requirements and concepts, to preliminary design, to critical design, to engineering and manufacturing design. Both SBD and PBD seek to find a feasible solution, meeting all current constraints. At any point during development, there are decisions that have not yet been made – the design at any point in time implies a set of alternative outcomes.

PDB and SBD differ in terms of the principles used to time decisions and make decisions. PBD potentially leads to faster design and design concepts in the absence of surprises, with tighter cost estimates, and increased confidence on the part of high-level decision makers. In the presence of possible surprises, SBD potentially leads to lower time and cost, and lower time and

cost variance. PBD is appropriate when the requirements can be assumed to be fixed. SBD is appropriate when the requirements are assumed to be more fluid than not.

SBD is seen as a potential methodology to quickly and economically adapt to changes in stakeholders constraints and the P-Spec with minimal disruption to the design process. SBD is also seen as a potential methodology to lead to better design solutions than traditional point-based design (PBD), since PBD might make choices before they needed to be made and close of design branches offering superior performance.

The philosophy of SBD is to hold off making decisions as long as possible, thus keeping as wide a range of options open as long as possible. The philosophy of PBD is to come to solution as quickly as possible. PBD "prefers" to make design decisions that minimize the remaining alternatives and options. SBD "prefers" to make design decisions that maximize the remaining alternatives and options. Whether using PBD or SBD, at any point during design, there are decisions yet-to-be-made, and so, in some sense there is a set of future options. PBD seeks to restrict the set of options as quickly as possible, to arrive at a concept or design, whose cost, reliability and other life-cycle characteristics can be estimated, whose performance can be tuned, that can be explained to decision-makers, and defines successful development for the program. SBD seeks to keep the design space of options open as long as possible so that changes in constraints and requirements can be accommodated with minimal impact on the acquisition program, and to avoid early convergence on a dominated, sub-optimal solution.

PBD attempts to come to a design concept quickly. This is in part an attempt to reduce the overall development time. It is also part of an attempt to reduce the uncertainty in time, cost and performance estimates. The idea is that the sooner a design becomes well defined, the sooner reliable cost and performance estimates can be made. Also, it is easier to explain a point design to funding and decision agencies than it is to explain that the detailed concept is yet-to-be-determined, but is somewhere in a set of options.

Given static constraints, PBD is expected to arrive at an articulated solution and cost estimates sooner. Given dynamic constraints, SBD is expected to arrive at solutions with lower variance in the time. In the larger view, we need to consider the costs and benefits of different methodologies, and when they should be applied. Consider uncertainty in system development. Is one less risky than the other (sunk cost at risk, value at risk, conditional value at risk)? Uncertainties include need (adversaries who choose conditions, tactics and equipment to attack our limitations and avoid our strengths, plus "random" nature of the world), and opportunity (adaptive component maturation investment to fill unmet needs, and unexpected opportunities).

The goals of this research are to establish a rigorous and principled foundation to express, compare and contrast SBD and PPD methodologies, to develop an analytic formulation for SBD, to develop an analytic approach to address the sequence and timing of design decisions, and to extend the SBD analytic MPT to provide information and guidelines to analyze alternatives. The objectives are to formalize SBD, PBD, and their differences and commonalities; to develop a

rigorous, principled, practical and relevant approach to organize design-space into "regions" in a way that is relevant to SBD, to integrate sequencing and timing considerations into SBD decision making, and to provide analytical methods for new insights for decisions and analysis during system development.

The scope and context encompasses MDD to MS-C: progressive resolution/instantiation of the system concept and range of time and costs vs capabilities. The DoD acquisition lifecycle is organized around progressive refinement of the system requirements, without specifying the design. At the same time, DoD acquisition practices are oriented producing reliable time, cost and performance estimates. The uncertainty in the estimates is greater when the design is uncertain ("set-based") than when there is a specific concept.

At any point in time, PBD has greater variance in time and cost given change in requirements and opportunities but lower expected time and cost, vice SBD. Over time, considering change dynamics, SBD may have lower variance and expected value.

4.5.4 A Formalism for Regions of Design Space

The section describes a sequence of concepts leading up to a rigorous and useful characterization of regions of design space that is compatible with SBD and which supports novel and valuable analysis operations during SBD. At this point in time the formalism is not complete, and there a several questions and issues that remain to be addressed.

The system architecture can be represented as an AND/OR graph with dependencies. At an AND node, all of the child subsystem have to be specified for the parent node to be completely specified. At an OR node, exactly one of the child nodes can be selected and must be selected to instantiate the OR node. At the bottom level of the graph are component options. The nodes one level up from the bottom are all OR nodes. Dependencies are relationships that cross branches of the graph identifying mutually incompatible choices, and significant performance or cost interactions. In a strictly logical AND/OR graph, AND and OR nodes alternate. In practice, because the nodes correspond to real subsystems, the architecture can have AND nodes followed by AND nodes matching the organization of the real system. The TARDEC Standard Product Classification Hierarchy is an example of the AND/OR graph for the generic ground vehicle architecture. A partial example is:

- Vehicle *AND*
  - Chassis *OR*
    - Monocoque hull
    - Frame
  - Suspension *OR*
    - Torsion bar
    - Hydra-pneumatic
    - McPherson
  - Running Gear *OR*
    - Tracks
    - Wheels *AND*
      - Configuration *OR*

        - 4-by-4, 6-by-6, etc.
      - Tire diameter OR …
  - Engine *OR* …
  - Dependencies *AND*
    - Suspension/Torsion bar & Running Gear/Wheels are incompatible
    - Suspension/McPherson & Running Gear/Tracks are incompatible

The system architecture can be "flattened" into a genomic representation. The genomic representation simplifies developing the design space region formalism. In the genomic representation, every OR node one level up from the bottom in the AND/OR graph is a gene site. The alleles at the a gene site are the component options at the corresponding OR node. Each gene site also has a "not present" allele to represent the case where the branch leading to that OR node was not chosen at a higher level. The genomic representation has completeness and consistency checks when it is mapped to the AND/OR graph: exactly one branch is instantiated at each OR node, every branch is instantiated at each AND node, and there are no "orphan" components (components alleles at an OR node that was not chosen).

A design solution in the genomic representation has an allele choice at each gene site, and it meets the consistency and completeness checks.

A feasible solution is a design solution that meets all the performance and life cycle requirements on all dimensions.

A *feasible region* is a set of *connected* feasible solutions. Two feasible solutions are 1-connected if they differ at exactly one gene site, so a single allele swap converts one to the other. Two feasible solutions are N-connected if (a) they are not N-1 connected, and (b) one is 1-connected to a solution that is (N-1)-connected to the other. Feasible regions can be grown from a feasible solution. Every feasible solution in a region is connected to every other solution in the region on a path of single gene changes such that every solution on the path is in the region.

A region differs from a set in that every pair of points in the region are connected on a path of "single nucleotide polymorphisms (SNPs)" that stays within the region at every stage, whereas sets are simply collections of point solutions that may not all be connected within the set.

A region is convex if for any two solutions in the region, every "mix and match" combination of their alleles is in the region. If two solutions in the region are N-connected, then every solution on every path of length N that connects the two is in the region.

Convex regions have a compact representation: the set of alleles at each gene site allowed in that region. Every combination is in the region (it is a "box" in design space). Feasible convex regions have a nice property: allele choices (within the region set) at each gene site can be made independently without violating feasibility.

Convex feasible regions can be grown from a feasible point solution by adding alleles one-at-a-time at gene sites, but only adding those that preserve convexity. Making the convex region growing unique, requires on unambiguous rule for which allele at which gene site to add next, e.g., add the allele at the gene site that maximizes region availability.

The availability of a component is the forecast probability that the component will be available when it is needed for design, integration and testing. Availability can change over time due to component maturation. Design solution availability is simply the product of the availabilities of the choices at all the gene sites. Region availability is the probability that at least one solution in the region is available.

Convex region availability has a simple computation. Availability at a gene site is the probability that at least one allele is available. The probability that a solution in the region is available is the product of the availabilities over all the gene sites. This assumes that the availabilities of alleles are independent, i.e., the availability of one allele at one gene site does not affect availability of any other allele at any gene site.

4.5.5 Analysis of Design Space Region, Topology, and Sensitivity to Requirements in SBD

To think about topology and geography, consider islands and shorelines at the continent, country and vacation spot resolution versus changing sea levels, and variable tides and sea states. As the water level changes (requirements restrict), connected islands and convex cores may split or join. In practice, the challenge is even more complicated. "Islands" may be connected at low-tide via a causeway, but not at high-tide or high-sea-state. Geographers have not resolved how to address temporal (seasonal; daily) variations vs connection (truck load limits, highway/bridge capacity, failure in trunk lines, etc.) in mapping the topology of regions. (In theory, this can all be resolved to the cost of connection – capacity, bandwidth and time to implement – except that these are unknowns and may be unknowable.)

The formalism presented in the previous section provides rigorous, principled, and computationally-feasible methods to analyze the topology of design space vis-à-vis the architecture and product specifications at two levels: (a) organization into disjoint, but self-connected, feasible regions, and (b) organization into convex feasible regions. Convex feasible regions are a refinement of feasible regions. In the organization of design space into a unique collection of feasible regions, each feasible region is self-connected (each point solution inside it is connected by SNPs to each other solution), and the feasible regions are distinct and non-intersecting. In the organization of design space into a unique collection of convex feasible

regions, each convex feasible region is *compact*, and the convex feasible regions are distinct in the sense that they do not have overlapping interiors, although they are not strictly disjoint since they can share points at their boundaries. The convex feasible regions have useful properties for SBD. Changes at a gene site (restricted to the allele set of the region for that gene site) can be made independent of any other gene site.

Feasible regions and convex feasible regions have descriptive characteristics that makes them useful for SBD – to gain insight into the topology of design space, to select significant regions to focus on, to characterized regions with point solutions for the purpose of exposition and nominal cost estimation, to identify tipping points where changes in requirements change the topology, and gain insight from the sensitivity of region properties to requirements changes and design decisions.

From the perspective of SBD, the interesting topic is "how do changes in requirements affect the topology and geography of design space?" The formalization provides the ability to answer these questions, but also more fundamental questions: "how much can the requirements be changed without changing the topology of design space?" - I.e., "where are the tipping points where feasible regions appear/disappear, split and merge?"

Other relevant questions that can be addressed include:

"What point design represents the regions (e.g., can generate the region and is, in a sense that is to be determined, the most resilient)?"

"How large is the region (e.g., number of point solutions, breadth on the minimum dimension)?"

"How dense is the region (ratio of size of the minimal circumscribing convex region to the non-convex circumscribed region)?

"How compact is the region (ratio of size of the inscribed convex region to the non-convex region)?"

"How sensitive are the characteristics of a region to requirements? To design decisions?"

"How sensitive is topology of design space to requirements? To design decisions?"

These questions, and others, e.g., based on change-cost versus number of independent questions, can be addressed in this framework, but further research is needed to articulate the issues, benefits, and methods.

5 GEORGIA INSTITUTE OF TECHNOLOGY

5.1 ACTIVITY 1. PAST RESULTS: PHASES 1 AND 2

5.1.1 PHASE 1 RESULTS

During Phase I, GTRI investigated relevant, existing tools and their ability to capture –ilities in a tradespace environment. The investigations were initially limited to those toolsets developed by SERC members involved in the ITAP Phase 1 work. GTRI investigated prior research in the area of web-based analytical tools for systems engineering decision-making. Of these, the GTRI-developed Framework for Assessing Cost and Technology (FACT) was identified as a promising toolset example that integrates a model based systems engineering (MBSE) approach and methodology to enable tradespace analysis. GTRI investigators therefore identified key aspects of FACT that could be used to help capture and analyze –ilities as their definitions matured during the course of the multi-year ITAP effort.

FACT is an open architecture web-based tool developed to enable collaborative tradespace exploration for early phase design of complex systems (initially military ground vehicles, but since extended to other applications) [Browne et al. 2013; Ender et al. 2012; O'Neal et al. 2011]. FACT embodies a web services based environment that enables models to be interconnected, providing a rapid exploration of the design tradespace in support of systems engineering analysis. FACT is model agnostic and capable of linking disparate models and simulations of both government and commercial origin through the application of community established data interoperability standards.

Further, FACT focuses on interoperability and data sharing with the emphasis centered on metadata. FACT was designed on a philosophy of open architecture to enable extensibility. To achieve this, and avoid the encumbrance of licensing fees limiting its use or tethering it to a single manufacturer over its lifetime, FACT was built using open source software components, and the US Government is free to use, modify and distribute it. FACT's development followed guidance from the Department of Defense, mandating that it be web-based and accessible from common computer workstations, be built entirely from open source software, and offer an open and extensible architecture [Assistant Secretary of Defense 2007; Assistant Secretary of Defense 2009].

FACT provides decision support tools to help manage risks of cost, schedule, and performance through a rapid analysis of alternative technology and materiel using surrogate models, or equation regression representations of more complex M&S tools. It is designed primarily to provide tradespace analysis during conceptual design. Other stages of the system lifecycle can benefit from the FACT process, but the conceptual design phase is where both good and bad decisions have the greatest impact on cost and performance. Although FACT's development

was originally envisioned for vehicle acquisition programs, the overall process and application is independent of vehicles and can be applied to any system-of-systems.

SysML was highly leveraged as the point of reference for the data schema implemented as FACT was initially developed. SysML[1] is a general-purpose graphical modeling language for MBSE applications that supports the specification, design, analysis and verification of a broad range of systems. It is a subset and extension of the Object Management Group's Unified Modeling Language (UML), the industry standard for modeling software-intensive systems, giving systems engineers the ability to represent system requirements, structure, behavior and properties using a formal diagram syntax.

The Phase 1 evaluation found that a FACT-like framework and methodology might well incorporate extensions to the SERC team's methods, especially as they are defined to capture -ilities tradespace of interest. During the Phase 1 timeframe, however, FACT existed as a specific development for application to ground vehicles for the USMC. The team determined that a more flexible and scalable integrating toolset might be better suited to support research and integration of –iliities defined as critical to support DoD and other acquisition and design processes.

5.1.2 PHASE 2 RESULTS

Phase 2 therefore extended the investigation to developed, open-source, web-based analytical tools outside of the ITAP effort. The goals were to identify the best tools and develop a flexible, integrated way to support analyses from various starting points in a workflow. In so doing, GTRI's contributions for the Phase 2 effort focused on two primary fronts: (1) integration of a toolset and workflow process to guide early stage design refinement, and (2) directly using this toolset and workflow to enable designers to begin assessing some key aspect of resiliency as related to evaluating design alternatives. The toolset was designed and then built to use open source technologies, supporting a workflow that would guide early stage design refinement while being extendable to more rigorously detailed design exploration. Conceptually, this built heavily from the FACT work cited previously but differed in that the aims were to create a flexible, open architecture, and integrating workflow that would support early-stage analytical research and method maturation across ITAP extensions as they mature.

The tools identified for use and integration in to a workflow to enable efficient generation of a tradespace and subsequent rational reduction of the design alternatives included SysML, described previously, and OpenMDAO. Recognizing widespread use of SysML across the DoD (to include the SERC's sponsor community), GTRI leveraged previous research for authoring SysML models and using those models to execute design tradespace exploration [Browne et al. 2013]. The integration of these capabilities was extended to allow feedbacks within the design process and allow compatibility with NASA's OpenMDAO framework.

[1] http://www.omgsysml.org/

OpenMDAO is an open-source Multidisciplinary Design Analysis and Optimization (MDAO) framework developed by NASA Glenn and Langley Research Centers [OpenMDAO, 2014]. It has been developed for use as an integrated analysis and design environment that can be applied to many systems engineering applications. It is capable of linking multiple disparate models or other analysis tools in a single design structure matrix that can map system design variables to performance attributes in a manner similar to Phoenix Integration's Model Center[2]. Applying OpenMDAO's built-in library of solvers, optimizers and design of experiments generation tools allows for rapid generation of design tradespaces of greater fidelity and complexity than in some previous efforts.

A tradespace is defined as a collection of design variables and system attributes, different levels of which characterize each design alternative for a given system. A model or collection of models is a mathematical representation of the system and any external variables necessary to map the input variables to output variables. Commonly, input variables are chosen to be system design variables while output variables are defined to be system attributes. This relationship may, however, be reversed depending on the mapping. Variables may be intrinsic to the system or dependent on conditions external to the system (e.g., cargo space versus miles per gallon). Some form of cost is also typically derived from the characteristics that describe each system design alternative. This is illustrated in Figure 11 where X and Y are used to describe input variables and output variables respectively.

OpenMDAO is a key element for tradespace generation. OpenMDAO wrappers can be created which extend the analysis performed to externally hosted models. The software then governs the interaction with these models by way of managing the input and output metadata, especially in terms of units and allowable ranges, and orders the execution of the various analytical blocks, which are then executed sequentially to generate a complete tradespace.

As an initial proof of concept an open source, web-based toolset was developed to couple a rationally guided workflow to existing analysis methods for design tradeoff evaluation. This toolset leverages existing open source web frameworks such as Python based Django[3] and Javascript based D3[4] to enable the rapid development of a complex, database-driven website. The OpenMDAO framework is used to facilitate complex analysis by linking together the separate models used to describe the behavior and performance of the system of interest. In addition to the use of several open source software frameworks, the toolset also takes advantage of the SysML modeling language to specify the parametric constraints that define system performance.

The proof of concept was specifically developed to evaluate and begin to test how an –ilities related analytical construct might be used within the toolset and workflow. One of the more mature concepts was that of Epoch Era Analysis (EEA) by the MIT ITAP collaborators, which was

[2] http://www.phoenix-int.com/software/phx-modelcenter.php
[3] https://www.djangoproject.com/
[4] http://d3js.org/

actually developed prior to ITAP. EEA began as a high-level framework to identify, structure, and evaluate the impact of different or changing dimensions on expected suitability or performance of systems [Beesmeyer, Ross, and Rhodes 2012; Ross and Rhodes 2008]. These dimensions may include broad environmental contexts such as weather, political or financial scenarios, and environmental or operational characteristics, as well as stakeholder needs across short-term variation ('epochs') or longer term, time-ordered sequences ('eras'). Different analytical methods were then incorporated into EEA over time (as discussed in subsequent publications by this group) to help quantify one aspect or another.

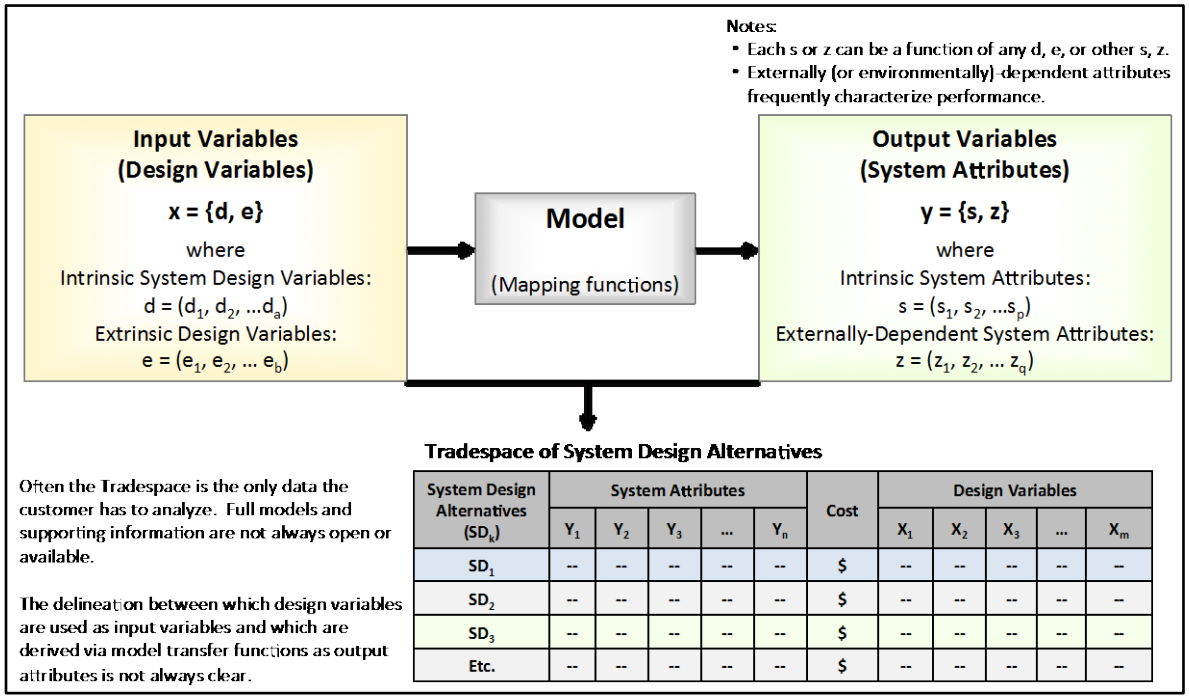


Figure 11: Defining a Tradespace

The GTRI team decided that a streamlined modification of EEA might serve as an excellent test construct for the process. Frequently, DoD users do not have sufficient data to flesh out the full concept of EEA as described in the literature. Also, analysts often are not able to specifically elucidate and specify a precise timeline with respect to how demands of a system might change much less how the relevant data might change over time. Consequently, the GTRI team focused on how to create a highly flexible analytical construct that could readily scale with vastly increasing tradespace sizes and yet still capture demands of competing stakeholders or changing performance requirements in an intuitive manner.

The end result of this effort was the development of a *Needs Context* construct. A significant concern during early phases of acquisition, or during the Pre-Milestone A analysis of the DoD Acquisition process is the resiliency of a system design across simultaneously competing or sequentially changing requirements on its performance attributes. A Needs Context is a scalable, applied methodology to capture certain dimensions of resiliency related to how well a system performs its functions in the face of requirements perturbations. It is defined based on

flexible subsets of performance attributes relevant to the stakeholder(s) and ranking of those attributes within each.

The motivation for the Needs Context is that choices must be made based on what is valued most by stakeholders, recognizing that some stakeholders may have a greater influence. A Needs Context can represent different or directly competing objectives for a system's performance for:

- Different stakeholders, each with different or competing priorities in parallel
- Changes in requirements over time (future performance requirements differ in series)
- Different mission profiles that necessitate different performance objectives, whether in parallel or in series

Value of a given system attribute is scaled against objective and threshold requirement levels, using the Key Performance Parameter (KPP) concept to promote comparability across analyses. Value of a system design alternative is then assessed using concepts from multi-attribute utility theory (MUAT), synergistic with the concept of evaluating *Robustness of Fielded System Capabilities and Capacity with respect to Operational Requirements* in terms of broad utility. Since each Needs Context may be defined using different attributes, and/or different valuations and preference weightings, Needs Context utility will have a different value for each system design alternative *k* (**SD_k**) within each individual Needs Context, i.e.:

$$[U_k = \alpha * v_j(Y_{jk}) + \beta * v_m(Y_{mk}) + \cdots + \gamma * v_n(Y_{nk})]_{Needs\ Context\ i}$$

…where **U_k** denotes the overall utility of system design alternative *k* (**SD_k**) for a given Needs Context, ***Y_{jk}*** represents system attribute *j* for system design alternative *k*, {**α, β, γ**} are weights derived from preference rankings or other means, and each **v_j** is a value function expressing the relative value of the given system attribute level to a stakeholder. Value functions are typically linear or exponential expressions but may be any monotonic function. Cost is also a function of system design alternative characteristics, though it depends on other influences and variables as well. Utility and cost are expressed as related dimensions, linked by an underlying **SD_k**. The Needs Context methodology adds a dimension of analysis to the classical utility representation as shown in Figure 12. The toolset, workflow, and example operationalized construct were described and published as part of INCOSE 2014 [Sitterle, Curry, Freeman, and Ender 2014].

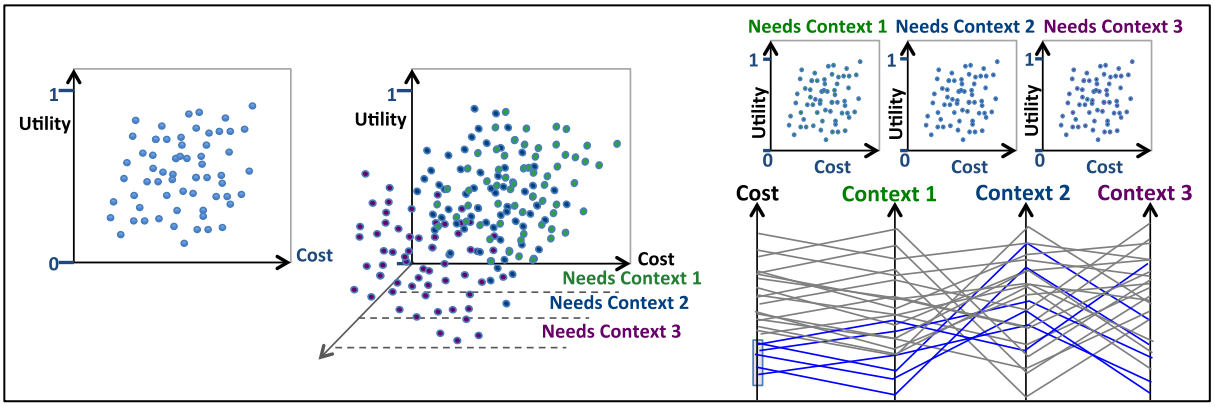


Figure 12: Classical 2D vs. Needs Context 3D Utility

5.1.3 PHASE 2 INSIGHTS

Specifically, Phase 2 sought to define and develop a capability through which new analytical methods may be explored, refined, and linked together in a design space environment. More complex analytical constructs and their subsequent synthesis into systems engineering decision aiding processes could be investigated and matured in this framework prior to integration into existing customer processes. This allows for customized performance attributes, especially as relating to hard-to-define "-ilities", to be investigated and matured for specific design domains.

Phase 2 efforts produced several insights. For one, starting with formally defined system model architectures has implications for structuring an integrated toolset for tradespace exploration. Firstly, performance-based attributes (e.g., braking distance) require a system model to couple with basic engineering representations of the system's operation and/or operational environment. The model of the system alone is insufficient to produce all quantitative system attributes that are typically important to the decision making process.

Most standard forms of utility evaluation derive from normalizations of the current design space with a single value function for each performance attribute. The impact is that utility is then not comparable from one analysis to the next when different performance attribute ranges are generated from differences in input variable ranges or system architecture. Similarly, using single value functions for each performance attribute implicitly assumes non-competing preferences across different stakeholders, mission profiles, etc.

The Needs Context construct described here avoids both of these limitations. By scaling to defined requirements, given the same contributing attributes (defined the in same way) and the same requirement levels, the broad utility measures captured in a Needs Context are comparable across analyses. The flexibility to define different requirement levels for a given performance attribute in each Needs Context also allows for simultaneous visualization and evaluation of competing objectives.

5.2 ACTIVITY 2. SUMMARY OF PHASE 3 RESULTS (TRADESPACE MPTS)

Building from the insights gained during Phase 2, the Phase 3 Tradespace Methods Processes and Tools (MPT) effort focused on 2 primary fronts. The first was more academic in nature, and investigated a more rigorous grounding as well as a few augmentations to the Needs Context method from Phase 2. The second focused on more robust integration of the processes and tools to support future extensibility to additional methods and –ilities-related analyses.

The first step was to more completely document the rationale for specifying the Needs Context in the way chosen during Phase 2 and to more firmly place that into context with the other ITAP work. There is already an enormous body of work focused on developing decision support methods and a tradespace toolset framework architecture in support of the Department of Defense's (DoD) analysis of various large-scale and complex engineering systems. This includes research and development of methodologies to conduct Analysis of Alternatives (AoA) relevant

to evaluating different dimensions of resiliency for these systems. Many of these dimensions are quite strongly related to the –ilities being investigated under the ITAP effort and the newer DoD priority for Engineered Resilient Systems (ERS).

Towards this end, tradespace analysis is of great importance and requires development and maturation of executable and scalable analytical constructs. These constructs must be implementable within the context of a larger workflow, and in tandem with each other, to guide trade space exploration and evaluate ERS resiliency concepts. In order to more clearly understand precisely what the Needs Context evaluates, and therefore help the broader community understand ways in which it either could or should not be used in conjunction with other constructs, we sought to more fully define its focus based on existing ontologies.

The need to evaluate systems according to requirement needs, changes in those requirements, and against some sort of human (or stakeholder) value has been well established in the literature. While by no means an exhaustive list, the following excerpts from literature trace this understanding well:

> *Challenges in requirements elicitation include:*
> - *Scope and context to reflect true user needs*
> - *Fostering understanding among different communities affected by the development of a given system*
> - *Requirements volatility (i.e., Requirements change over time.)*
>
> *[Christel & Kang 1992]*

> *"There is… an explicit recognition that designs are created to provide value to humans, and design decisions are made in some attempt to maximize that value. Decision-based design recognizes the need for human intervention in the design process in four areas: determination of human values, determination of the relationships to be included in analytical models, assessment of probabilities for random events, and creativity through the creation of design options coupled with judgment on which options should be given consideration for implementation."*
>
> *[Hazelrig, 1998]*

> *"Knowledge about evolution, and likely future requirements is critical to incorporate functional and performance options within an architecture."*
>
> *[Schultz & Fricke 1999]*

> *"Robustness characterizes a system's ability to be insensitive towards changing environments. Robust systems deliver their intended functionality under varying operating conditions without being changed."*
>
> *[Fricke & Schultz 2005]*

> *"Value robustness is the ability of a system to continue to deliver stakeholder value in the face of changing contexts and needs."*
>
> *[Ross & Rhodes 2008 (IEEE SysCon)]*

"To achieve value robustness, design systems "… using natural value-centric timescales, wherein the context and expectations define the timescales."

[Ross & Rhodes 2008 (INCOSE)]

The Needs Context analytical basis and workflow derive strongly from concepts espoused by Hazelrig (1998), conceptually expanded by Fricke and Schultz [1999 & 2005], and further delineated by Ross and Rhodes [2008] as EEA. It was constructed to specifically help evaluate robustness in the face of changing or competing stakeholder needs. As such, the Needs Context relates directly to the ERS concepts of a system being "trusted and effective in a wide range of contexts" and exhibiting "broad utility" [Goerger et al 2014].

A broad review on flexibility performed by the AFIT team of ITAP collaborators placed many concepts relevant to resiliency into an ontological framework [Ryan et al 2013]. This work comprehensively analysed systems engineering literature to arrive at a set of proposed, consistent definitions for flexibility, agility, adaptability, and robustness. The authors found consistency with the analytical developments by Fricke and Schultz [2005]. Robustness, for example, was subsequently specified as a measure of how effectively a system could maintain a given set of capabilities in response to external changes after it has been fielded.

The Needs Context analytical construct therefore builds from robustness as defined by Ryan et al. [2013] and the concept of broad utility advocated by Goerger et al. [2014] to create a requirements-based evaluation of a non-cost value of system design alternatives. *Robustness of Fielded System Capabilities and Capacity with respect to Operational Requirements* is a more complete descriptor to aid accuracy and utility of a Needs Context as a building block in future analyses.

We then investigated the relevance of placement of the Needs Context within a larger analytical workflow. The Needs Context process begins by defining any number of Needs Contexts, each defined by some individualized subset of the performance attributes defined for the system. The attributes within each Needs Context are assigned threshold and objective requirement values as well as a preference ranking (i.e., preference across attributes within the given Needs Context) or weighted by any other means. The same performance attribute may be used in multiple Needs Contexts and characterized differently by threshold, objective, and rank/ weight values in each one.

The Needs Context is essentially an analytical filter to rapidly identify the system design alternatives that provide the most value to Stakeholders and/or mission needs in a transparent, scalable manner that preserves future comparability. As such, it may apply to a raw, freshly generated tradespace or a tradespace that has already been filtered by some other means. The Needs Context approach and algorithms are robust to either scenario and may be applied at any point in an analytical workflow. It is important to understand, however, what the Needs Context does by way of selecting design alternatives based on a Broad Utility evaluation of a specific Robustness dimension. This will allow proper systems engineering of an entire

analytical workflow such that the information presented to an analyst is consistent with the goals and specifications of ERS.

Consider that a typical tradespace for early-stage design alternatives of defense systems may be quite large. There may be thousands of alternative designs and hundreds of design attributes if not more in some cases. Taken together with the immense flexibility inherent in the Needs Context construct, it can be challenging for an analyst to have a good understanding of what impact a given Needs Context produces in terms of valuing the design alternatives. We therefore need a way to evaluate the big picture of this impact on a per-Needs-Context basis before moving on to evaluation of multiple Needs Contexts and their tradeoffs simultaneously.

One simple and clear way to achieve this is to create a raw count histogram of the top X% of utility scores for a given Needs Context, and overlay these scores with the raw count histogram from the "full tradespace" before application of the Needs Context filter. 25% is a good example number for this approach to be informative. Using X% = 25% as an example, this does not refer to those design alternatives with utility scores above 0.75 but rather the top 25% of all utility scores for the given Needs Context. This histogram overlay will present a visual graphic to an analyst showing the effect of the given Needs Context before accepting those results and proceeding to the next Needs Context definition or analysis of all Needs Contexts. This visualization may be displayed alongside a heatmap showing the Spearman correlation coefficients from the "full tradespace". These correlations will help analyst quickly see positive or negative relationships for attributes. That way, if 'best' values are lost for a given attribute that was not defined in a Needs Context, we can see how it related to other attributes and therefore why. This knowledge will allow the analyst to decide if this is acceptable or if the Needs Context should be adjusted. The intent is to provide a quick and intuitive "big picture understanding for each Needs Context.

The overall workflow for this process is depicted in Figure 13. The precise equations and math behind these steps have been provided previously in [Sitterle et al. 2014] and GTRI's contributions to the Phase 2 ITAP Final Technical Report.

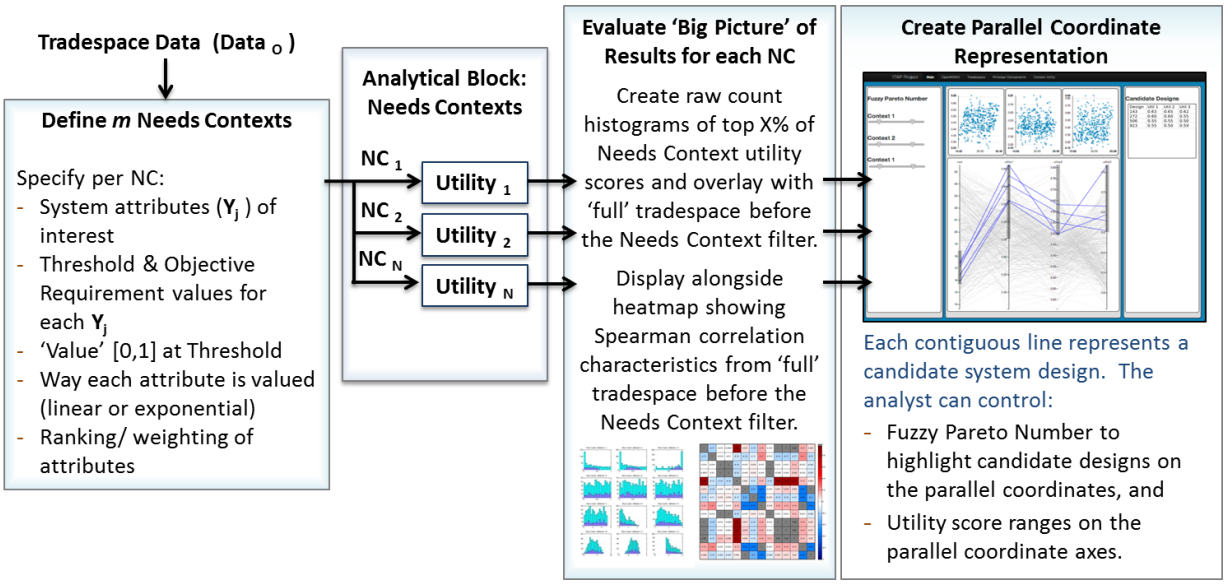


Figure 13: Graphical depiction of Needs Context Steps and Workflow

The GTRI team next investigated different methods that might help modify this approach if the attributes specified as part of the value equation were not truly independent from a stakeholder perspective. Most valuations methods still use some version of simple additive weighted (SAW) measures – including multi-attribute utility theory (MUAT) methods – which frequently assume that no interaction between the valued attributes exists even when that has not been verified. Fuzzy methods evolved in part to cope with potential interactions as well as a means of handling uncertainty surrounding them. These methods include non-additive subjective and objective functions such as ordered weighted average (OWA) and fuzzy integrals. The Choquet integral is part of this latter category and has been used previously in the context of multi-criteria decision-making. [Grabisch and Labreuche, 2010)]

Like SAW measures of utility, the OWA and Choquet integral operators are aggregation functions. However, a fundamental difference is the reordering process used for the fuzzy methods. For example, a weight in an OWA aggregation function is not associated with a specific argument but rather with its ordered position in the aggregate [Grabisch 1996; Ben Hassine-Guetari, Darmont, and Chachat 2010]. In our case, this would correspond to ordering the values of specified attributes (i.e., as defined by the attribute's value function scaled as described previously) for a given design alternative, with the weights dependent on the placement of the value in that reordering. This approach does not, however, take into account that a decision maker may have very distinct priorities for which attribute is the most important that are not reflected by the valuation of that attribute.

The Choquet integral in its discrete form uses a similar reordering basis and substitutes the classical weight vector with a monotone fuzzy set function called a capacity. These fuzzy integrals can represent interaction between criteria because a weight of importance (i.e., a capacity) is attributed to every subset of criteria. The fuzzy weights assigned to the various

subsets of *n* total attributes may be supra-additive, additive, or sub-additive. However, for an alternative with *n* attributes, (*2n – 2*) coefficients must be either specified by the analyst or determined via some other means to evaluate a Choquet interval [Grabisch & Rouens, 2000; Fu, Hall, & Lawry, 2005]. The number of coefficients involved in the fuzzy integral model grows exponentially with the number of criteria to be aggregated. And, the attribute valuations and hence integral equation changes with each alternative according to the reordering of the specified attributes.

The exponential complexity of evaluating a design alternative "valuation" using Choquest integrals makes the method exceeding difficult to use for large or even moderate numbers of contributing attributes. Additionally, while methods including partial preference ranking and information theoretic functions have been employed to calculate the numerous fuzzy weights on the subsets and ease the burden otherwise on the analyst, these do not eliminate the need to reorder the attribute values that populate the integral structure for each alternative. Many of these methods also depend on some quantifiable characteristics present in the data (i.e., the tradespace) at that point in the analytical process. For example, using a tradespace generated measure such as one of the various correlation measures between the attributes may be employed to determine supra, simple, or sub-additivity for the capacity relationships. Unfortunately, this would eliminate the direct comparability of one tradespace analysis to the next since such measures will change not only from one tradespace to the next but also with any other analytical "filtering" method applied earlier in the process.

In light of these experiences, the GTRI team next chose to consider various ways in which uncertainty could be brought into the additive function for broad utility and carried forward as part of the analytical process. In an additive function of weighted valuations on attributes, uncertainty could come in on either the valuations or the weights. Uncertainty on the value functions would be expressed in terms of uncertainty associated with the KPP scaling parameters, the objective and threshold requirement levels that determined the valuation. The potential space for valuation this could open, however, did not preserve the original intent of this construct and process.

Turning therefore to uncertainty on the weights, we considered two distinct cases: (i) ordinal ranking of attributes was provided but without certainty, or (ii) no ordinal ranks at all were provided. (Recall that a Needs Context by definition specifies only a subset – though any subset – of attributes present in the tradespace. It is so defined in order to provide evaluations of alternatives based on what is most critical to stakeholders.) Applying uncertainty to the weights for the additive value function creates a stochastic version of the analysis achieved via an efficient Monte Carlo on a specified random distribution for (i) and constrained according to an ordinally defined convex polytope for (ii). This approach is well developed and applied to multi-attribute additive utility problems throughout the literature. The approach was well defined by Charnetski and Soland [1976 & 1978], Parnell [1999] and then expanded upon tremendously by [Lahdelma et al. 1998; 2003; Tervonen, Tommi, and Lahdelma, 2007]. The benefit of including uncertainty in this manner is that it allows us to move away from

deterministic evaluations without requiring in depth knowledge about models or other sources of uncertainty that may not be available to the analyst.

The second primary focus of Phase 3 was to implement the processes and tools in a robust way such that the resulting framework can be leveraged in future Phases to further support additional methods and –ilitiy-related analyses. One principle tenet to accomplish this focus is to recognize that end users have different initial conditions to their problems and require a framework that enables them to inject themselves into different points along the generalized Systems Engineering workflow.

For example, one user may already have a high level SysML model describing their problem while another may have a set of analysis models that can be executed to generate a tradespace. In fact, a user may have previously generated their tradespace data using outside analysis codes but need to perform additional analyses as part of a trade study to extract design alternatives that satisfy their particular needs.

While the framework is informed by this desired end user flexibility, specifically for Phase 3 the Needs Context analysis formulated in Phase 2 was decomposed into smaller reusable functional building blocks. These functional blocks were then implemented in the Python programming language as steps in a data pipelining tool. Now particular Needs Contexts analyses can be performed by composing these blocks together into a pipeline and then passing in a tradespace to analyze. Because of the flexible framework, the pipelining step is separate from the tradespace generation step to allow a user to inject a previously executed tradespace data set from outside the framework. In the future, additional blocks can be developed along a similar paradigm that can be composed with the Needs Context analysis to address alternative –ilitiy-related metrics and analyses.

In addition to the development and implementation of the data pipelining additional effort has been to expand on the initial web tool from Phase 2. The web tool explored in Phase 2, and in other work, offers a collaborative and distributed way to interact with the flexible framework being developed. This interface development work has been to offer additional visual feedback to the user about the particular Needs Contexts and tradespace under consideration and support filtering of the tradespace based on each design alternative's utility.

5.2.1 Phase 3 Insights on Tradespace Analysis

A major understanding from this process is that generating a tradespace from various models is not a trivial task if the goals are to achieve flexibility, scalability (often via properly orchestrated modularity), and efficiency of the process. Also, a use case has a specific path through the networked workflow. Driving the tool development with a generalized workflow helps ensure we can meet the requirements of future use cases.

Similarly, a Needs Context is not the same as a use case. In an operational sense, the latter is a defined scenario that captures various exogenous conditions under which the system must achieve desired performance. Use cases are therefore typically unique to the defined operational scenario and variable levels/ ranges defined. In contrast, a Needs Context is defined from a stakeholder perspective and based entirely on performance requirements and prioritization of those requirements. It may indeed help evaluate a use case, but the definitional basis is distinct.

In addition, -ilities are often defined according to life cycle stage or blur across several; care must be taken to operationalize appropriately. Specifying the precise way in which any analytical construct applies to tradespace analysis and also its specific lifecycle context is critical to future synthesis with other methods. Composability and traceability of constructs is key to future maturation using other methods in tandem. MPTs must be modular, efficient, and scalable for same reasons as above.

Uncertainty was investigated here and applied based on methods found in the literature in order to help elucidate how to propagate that through a series of analyses using some of the processes and tools described. The key is not in applying a well-understood concept as much as understanding how to blend various sources of uncertainty across constructs in a scalable but also meaningful way that can be intuitively brought into visualizations to guide the analyst. Methods must pay attention to computational feasibility as the dimensionality of the problem increases; else the method should incorporate some sort of dimensionality reduction.

5.3 Activity 2. Summary of Phase 3 Results (SysML Based Cost Modeling)

This section describes the SysML-based cost modeling work in ITAP being led by Russell Peak (GT) and Jo Ann Lane (USC), as well as Ray Madachy (NPS) who recently joined in this collaboration. One key ITAP research goal is to create a framework that enables a wide variety of -ility models to come together to support system trade studies (including cost modeling as one aspect of affordability). In this particular task we are bringing together four main bodies of work (BWj) toward this goal:

- (BW1) The FACT work (which T. Ender et al. bring to the RT113 team) provides front-end trade study capability.
- (BW2) The MIM work (which R. Peak et al. bring to the RT113 team) provides fine-grain associativity capability to connect diverse models (including leaf-level design models and analysis models), as well as knowledge representation patterns to fold in all kinds of "-ility" models. This could potentially enhance the backend of FACT (ultimately leading to a more generalized method beyond MIM). [Peak *et al.* 2010]
- (BW3) The cost/effort modeling work (which B. Boehm, J. Lane, R. Madachy, et al. bring to the RT113 team) is one key type of "-ility" model that can be represented in the above

framework (e.g., to incorporate cost analysis with other analyses and trade study aspects involving diverse comprehensive "-ility" considerations). This includes cost modeling for systems engineering (COSYSMO), COSYSMO for systems of systems (SoS), software development effort modeling (COCOMO), and related work. [Lane, 2009; Madachy, 1997; et al.]

Other ITAP team members can potentially provide expertise with other models that can be similarly represented (e.g., as others have recently implemented via SysML for manufacturability, environmental sustainability, and end-of-life recyclability [Romaniw and Bras, 2010; 2011] and [Culler, 2010]).

- (BW4) The overall SysML/MBE/MBSE technology area (which R. Peak, T. Ender, et al. represent on the RT113 team) provides a practical means to embody and deliver the above technology and concepts. [www.omgsysml.org]

See the ITAP Phase 2 report (for CY2013) for further background context and examples of the above bodies of work and their envisioned combination in ITAP.

5.3.1 Approach Toward SysML-Based Cost Modeling within MIM and FACT

During Stage 1 of this task (Oct-Dec 2013) the focus was both (i) to define the big picture context as highlighted above, and (ii) to implement an initial example (Case Study 1).

During Stage 2 (Jan-Dec 2014), which this current report covers, we have focused on the following:

- Enhancing and refining the generic SysML-based cost modeling building blocks from Stage 1. These building blocks and their underlying cost modeling principles are generic and thus can be applied to practically any system (not just Case Study1 or Case Study2).
- Updating the Case Study1 implementation where needed as one means to re-verify the updated building blocks.
- Implementing a new case study, Case Study2 [Lane, 2010], which has more complexity and exercises more facets of the above cost modeling building blocks.
- Exploring additional technologies that can make interacting with these models more intuitive and comfortable for people who do not have much background with SysML.
- Investigating how to interface with the growing body of SysML-based system models. The goal is to wire together a SysML-based system model with a SysML-based cost model, including automating the calculation of the cost drivers and size drivers that a COSYSMO model requires. In contrast, today the typical practice is to manually estimate those values and provide them as manual inputs into a COSYSMO model.
- Identifying potential additional future applications, both in terms of analysis capabilities and case studies. For example, we should be able to support risk analysis (Madachy 1997, 2013) by leveraging the same cost and size driver building blocks we have already created.

The sections below highlight this Stage 2 work.

5.3.2 KNOWLEDGE CAPTURE VIA GENERAL-PURPOSE SYSML BUILDING BLOCKS

As given in the ITAP Phase 2 report (Dec 2013), in Stage 1 we took the COSYSMO-SoS cost modeling concepts described in [Lane, 2009] and captured them as general-purpose SysML building blocks. During Stage 2 we refined these building blocks, including correcting and generalizing some calculations (which were not exercised in Case Study1). We also identified additional useful parameters that can be derived from the existing parameters, such as metrics regarding the quantities of various types of requirements at different SoS levels. We implemented a few of these new parameters in Stage 2, and we plan to implement more during Stage 3. The resulting building blocks are highlighted in Figure 14. The next section illustrates how these building blocks are applied in two particular case studies.

Figure 15 graphically illustrates the comparison between a traditional COSYSMO spreadsheet (for a single system) and a DNA signature (equation graph) that can be auto-generated from the COSYSMO building blocks implemented in SysML. This DNA signature structure holds true regardless of the case study or actual system that the building blocks are applied to. The cost driver calculations and cost effort multiplier variable are seen in the typical dandelion roll-up pattern in the lower-right portion of the DNA signature. The size driver calculations and equivalent size variable roll-up are seen in a sparser dandelion pattern in the upper-left portion of the DNA signature. The bottom-line estimate of systems engineering effort is calculated via the equations in the middle that utilize these cost effort multiplier and equivalent size variables.

In case the reader is not familiar with the concepts of DNA signatures (equation graphs), Figure 16 summarizes the notation and corresponding SysML elements from which they are generated. Briefly, a red shape represents an equation, a blue shape represents the local variables utilized in that equation, and a yellow shape represents the block properties that are connected to those variables.

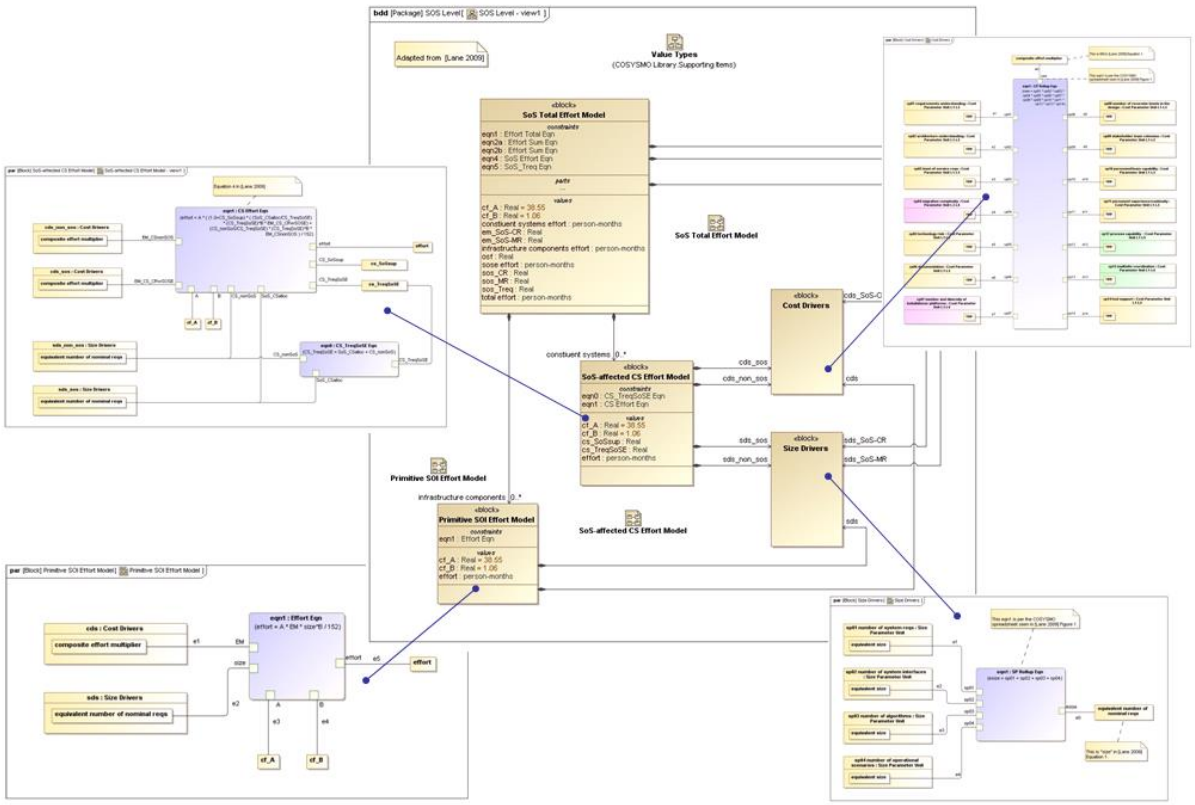


Figure 14: Implementation of COSYSMO-SoS cost/effort modeling concepts as general-purpose SysML building blocks — selected SysML diagrams.

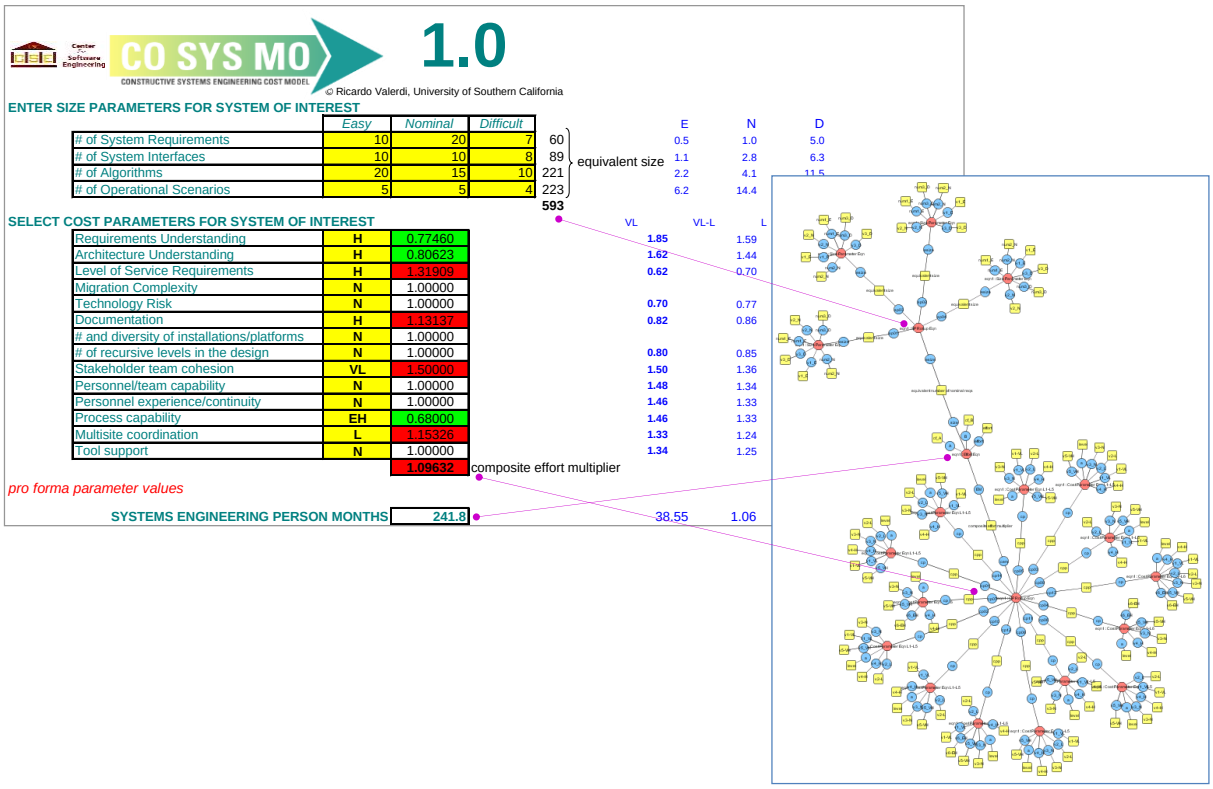


Figure 15: COSYSMO concepts as (i) traditional spreadsheet (on left) compared to (ii) SysML-based DNA signature (on right).

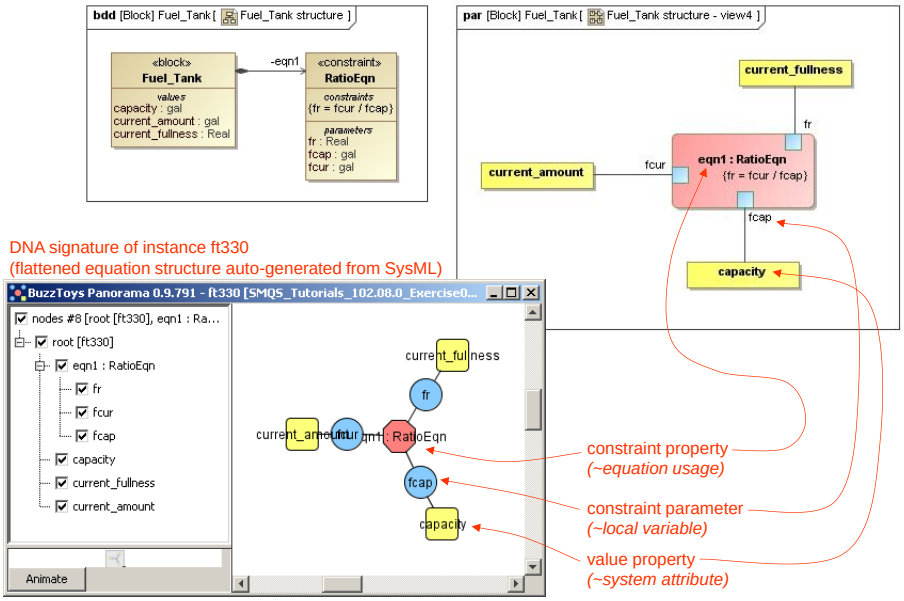


Figure 16: DNA signature nomenclature and corresponding SysML elements (illustrated via a Fuel Tank tutorial example).

5.3.3 Overview and Comparison of Case Studies

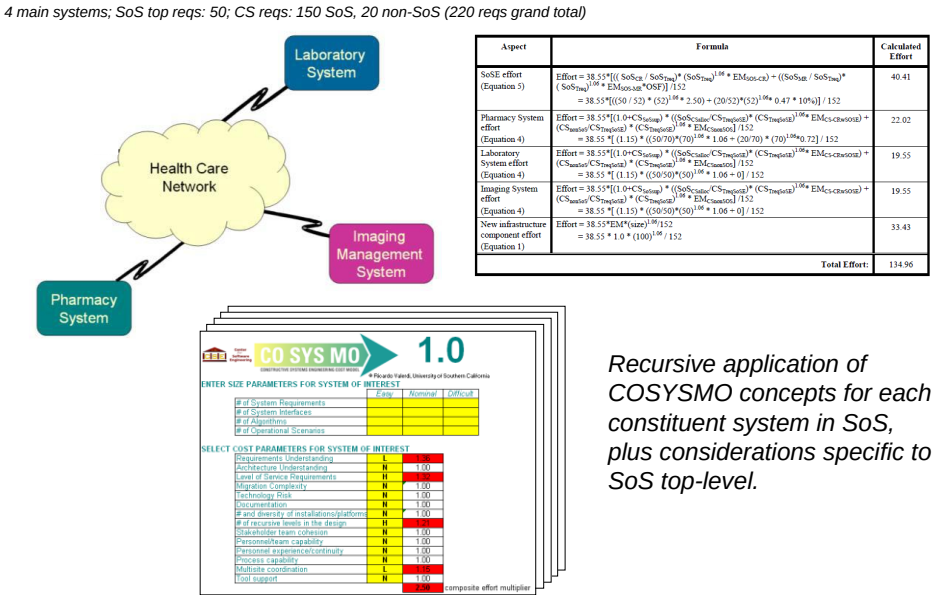
The composite ITAP team leveraged two existing (non-SysML) case studies to illustrate the overall approach and to verify the results. Both associated papers [Lane 2009; and Lane 2010] are comprehensive and include specific equations and numbers, which makes it feasible to readily verify the new SysML implementations. The original papers utilize COSYSMO cost/effort estimation concepts at both the SoS and single system levels.**Error! Reference source not ound.** Table 8 shows some of the metrics for each case study, where you can see that Case Study2 has two more systems versus Case Study1, as well as more SoS-level requirements and more constituent system (CS)-level requirements, and thus more resulting cost model variables and equations. The next two sections cover each case study individually.

Table 8: Comparison of case study complexity (healthcare SoS Case Study1 versus healthcare SoS Case Study2).

SoS Case Study	# of Systems	# of Requirements	# of Vars.	# of Eqns.	Notes
1 [Lane 2009]	4	220 grand total: 50 SoS top reqs; CS reqs; 150 SoS, 20 non-SoS	1166	204	Baseline complexity.
2 [Lane 2010]	6	680 grand total: 130 SoS top reqs; CS reqs; 375 SoS, 175 non-SoS	1830	320	More facets and complexity.

5.3.4 CASE STUDY 1 – HEALTHCARE SOS (BASELINE COMPLEXITY)

This section highlights the healthcare SoS Case Study1 implemented during Stage 1 (see ITAP Phase 2 report from Dec 2013). During Stage 2 we also enhanced its implementation and added additional views as presented here. Some of the original equation table and spreadsheet views from [Lane 2009] are given in Figure 17, while Figure 18 shows the new SysML implementation execution and results summary. In Figure 19 the results comparison indicates successful verification of the SysML implementation. Figure 20 highlights the SysML-based implementation and the resulting top-level DNA signature (as well as DNA signatures for selected sub-elements in the case study). These initial results graphically illustrate the repeated and recursive patterns that one would expect from an SoS cost model based on COSYSMO concepts.



Aspect	Formula	Calculated Effort
SoSE effort (Equation 5)	Effort = 38.55*[((SoS_{CR} / SoS_{Treq})* (SoS_{Treq})$^{1.06}$ * EM_{SOS-CR}) + ((SoS_{IR} / SoS_{Treq})* (SoS_{Treq})$^{1.06}$ * $EM_{(SOS-IR}$*OSF)] /152 = 38.55*[((50 / 52) * (52)$^{1.06}$ * 2.50) + (20/52)*(52)$^{1.06}$* 0.47 * 10%)] / 152	40.41
Pharmacy System effort (Equation 4)	Effort = 38.55*[(1.0+CS_{ToStep}) * ((SoS_{CSdisc}/$CS_{TReqToSE}$)* ($CS_{TReqToSE}$)$^{1.06}$* $EM_{CS-CRwSOSE}$) + (CS_{nonSoS}/$CS_{TReqToSE}$) * ($CS_{TReqToSE}$)$^{1.06}$ * $EM_{CSnonSOS}$] /152 = 38.55 *[(1.15) * ((50/70)*(70)$^{1.06}$ * 1.06 + (20/70) * (70)$^{1.06}$0.72] / 152	22.02
Laboratory System effort (Equation 4)	Effort = 38.55*[(1.0+CS_{ToStep}) * ((SoS_{CSdisc}/$CS_{TReqToSE}$)* ($CS_{TReqToSE}$)$^{1.06}$* $EM_{CS-CRwSOSE}$) + (CS_{nonSoS}/$CS_{TReqToSE}$) * ($CS_{TReqToSE}$)$^{1.06}$ * $EM_{CSnonSOS}$] /152 = 38.55 *[(1.15) * ((50/50)*(50)$^{1.06}$ * 1.06 + 0] / 152	19.55
Imaging System effort (Equation 4)	Effort = 38.55*[(1.0+CS_{ToStep}) * ((SoS_{CSdisc}/$CS_{TReqToSE}$)* ($CS_{TReqToSE}$)$^{1.06}$* $EM_{CS-CRwSOSE}$) + (CS_{nonSoS}/$CS_{TReqToSE}$) * ($CS_{TReqToSE}$)$^{1.06}$ * $EM_{CSnonSOS}$] /152 = 38.55 *[(1.15) * ((50/50)*(50)$^{1.06}$ * 1.06 + 0] / 152	19.55
New infrastructure component effort (Equation 1)	Effort = 38.55*EM*(size)$^{1.06}$/152 = 38.55 * 1.0 * (100)$^{1.06}$ / 152	33.43
	Total Effort:	134.96

Recursive application of COSYSMO concepts for each constituent system in SoS, plus considerations specific to SoS top-level.

Figure 17: Healthcare SoS Case Study1 and its main systems: original document and spreadsheet views. [Lane, 2009] [5]

[5] Technically Case Study1 also has a Patient Management System, but it has no updates or affects at the SoS level in Case Study1 and is thus not illustrated here.

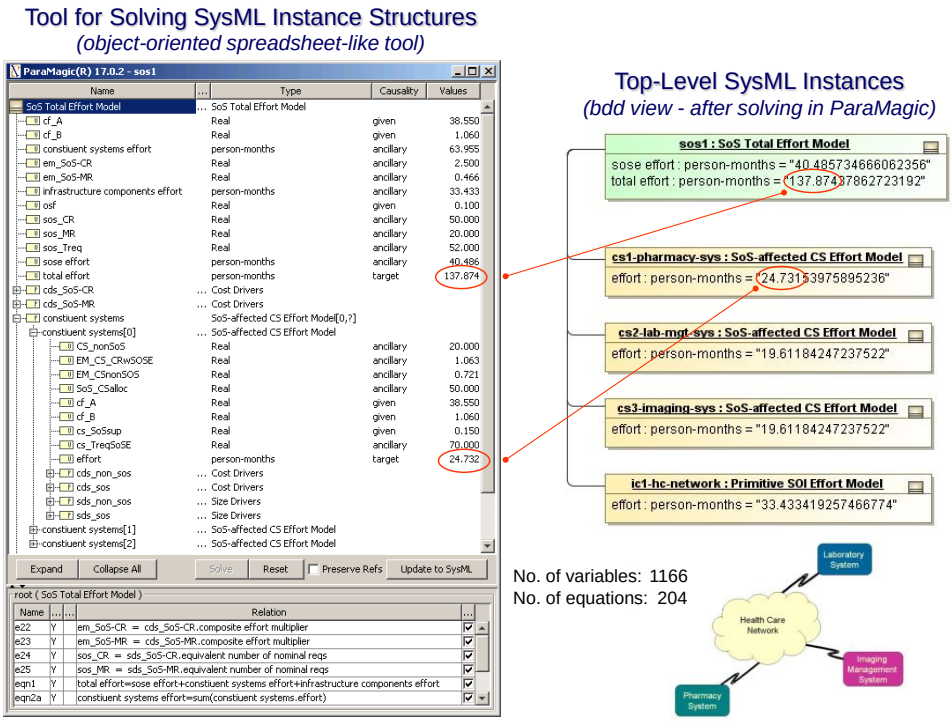


Figure 18: Healthcare SoS Case Study1: execution of the SysML model and its calculations.

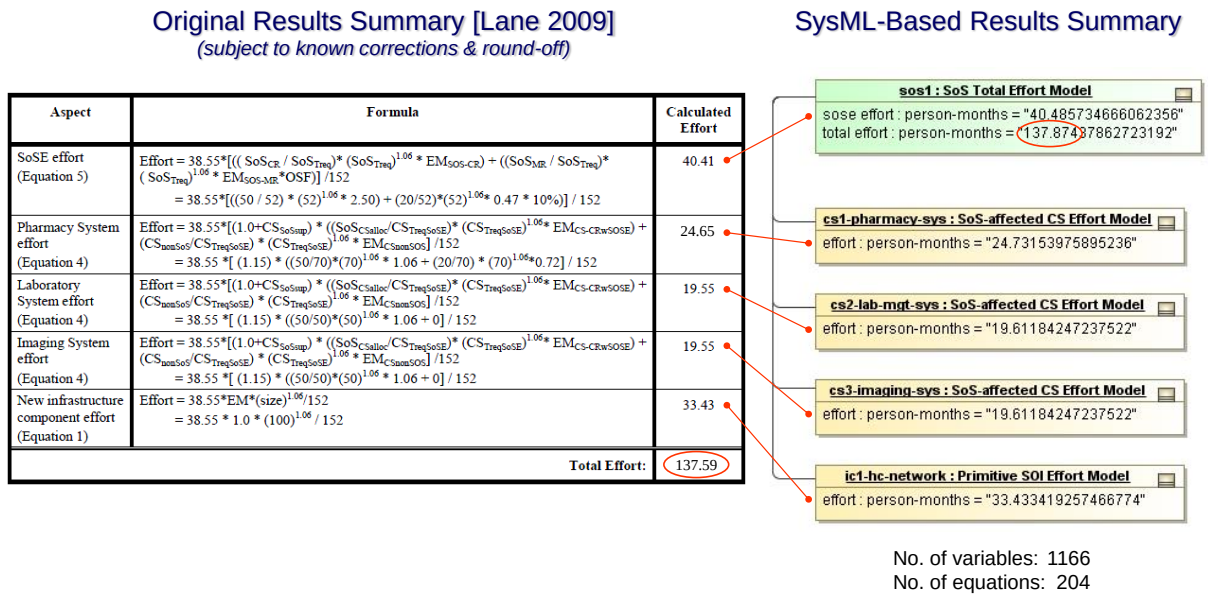


Figure 19: Healthcare SoS Case Study1: verification of SysML model results compared to original results.

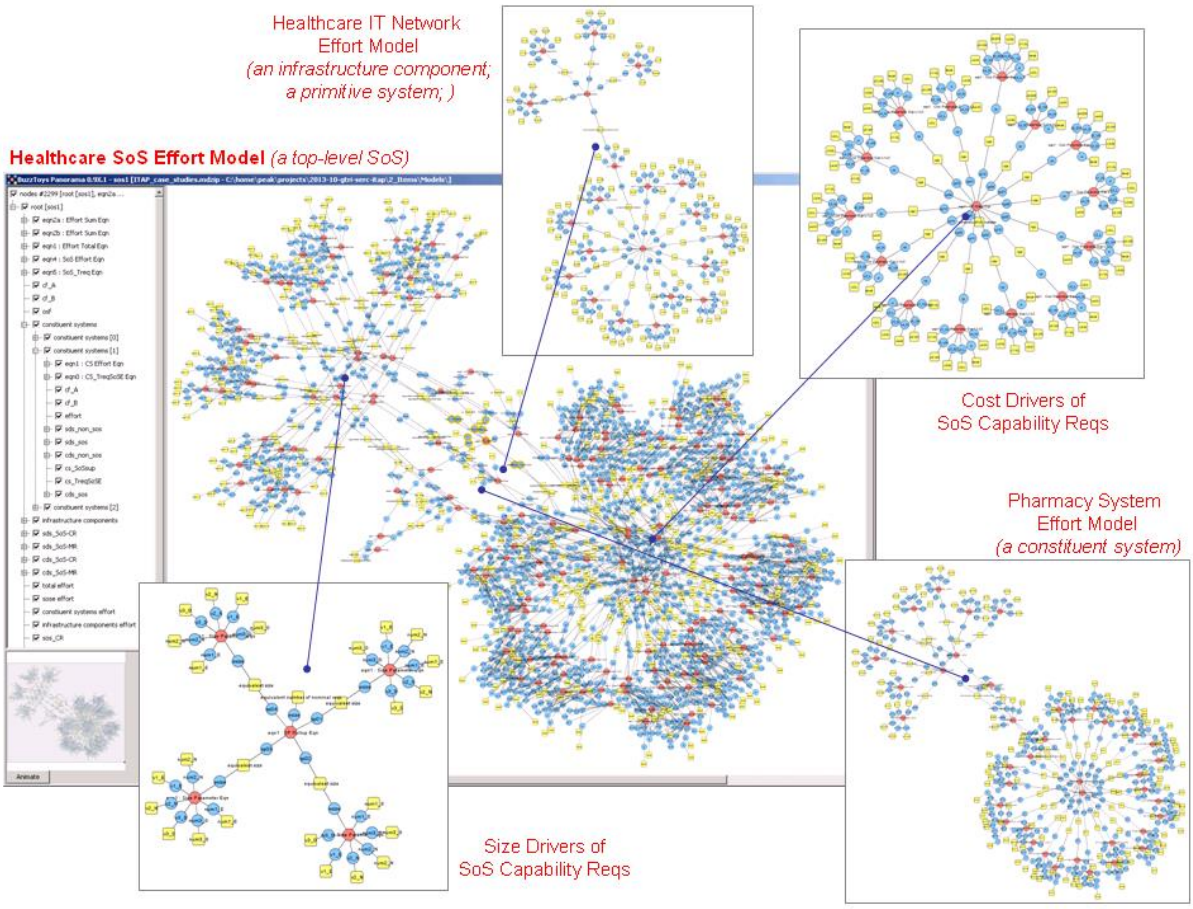


Figure 20: Healthcare SoS Case Study1: selected DNA signatures auto-generated from the SysML model.

5.3.5 CASE STUDY 2 – HEALTHCARE SOS (INCREASED COMPLEXITY)

During Stage 2 the team utilized an additional healthcare SoS example from [Lane, 2010] as Case Study2. This example has increased complexity vs. Case Study1 as seen in **Error! Reference ource not found.** Table 8. Also note that it exercises additional features and variations in COSYSMO-SoS that Case Study1 did not. Figure 21 summarizes the original case study. Figure 22 illustrates the execution of the Case Study2 SysML model and successful verification compared to the original results. In Figure 23 we see the same overall DNA signature patterns that are present in Case Study1, but note that the clusters are denser because Case Study 2 has more systems.

6 main systems; SoS top reqs: 130; CS reqs: 375 SoS, 175 non-SoS (680 reqs grand total)

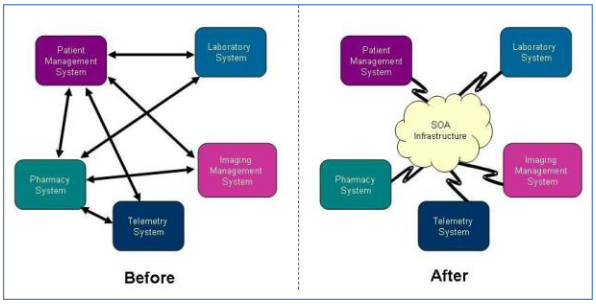
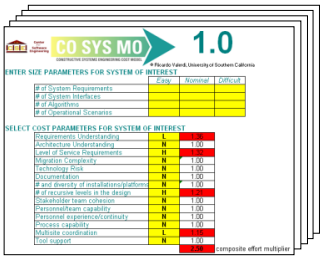


Table 6-2. Health Care SoS SOA Capability Size Overview

SoS Product (CS)	SoS SOA Requirements ($SoS_{CSalloc}$)	Concurrent Product Upgrade Size (CS_{nonSoS})	Total Requirements for CS ($CS_{TreqSoSE}$)
SOA Infrastructure	• Infrastructure characteristics: 20 • Shared data standards: 10	None (new constituent system)	30
Patient Management System	• Infrastructure characteristics: 10 • Shared data standards: 100	30 requirements	140
Pharmacy System	• Infrastructure characteristics: 10 • Shared data standards: 75	60 requirements	145
Telemetry System	• Infrastructure characteristics: 10 • Shared data standards: 40	25 requirements	75
Imaging Management System	• Infrastructure characteristics: 10 • Shared data standards: 35	50 requirements	95
Laboratory System	• Infrastructure characteristics: 10 • Shared data standards: 45	10 requirements	65

Recursively uses COSYSMO, and adds SoS aspects.



Using the values in Table 6-3, the total SE upgrade effort calc
which is the sum of equations 6-6 through 6-12 shown below.

Equation 6-6: *SoSE Effort*

$= 38.55*[(1.130/138.75)*(138.75)^{1.06} * 2.50) + ((8.75/138.75)*(138.7$
$= 112$ person months

Equation 6-7: *SOA Infrastructure Constituent System SE Effort*

$= 38.55*[(1.0+0.30) * ((30_r/30)*(30)^{1.06} * 1.62)] /152$
$= 19.65$ person months

Equation 6-8: *Patient Management Constituent System SE Effort*

$= 39.47*[(1.0+0.10) * ((110/140)*(140)^{1.06} * 1.06) + (30/140) * (140)^{1.06}$
$= 52.44$ person months

Equation 6-9: *Pharmacy Constituent System SE Effort*

$= 39.25*[(1.0+0.10_R) * ((85/145)*(145)^{1.06} * 1.06) + (60/145) * (145)^{1.06}$
$= 46.62$ person months

Equation 6-10: *Telemetry Constituent System SE Effort*

$= 36.82*[(1.0+0.10) * ((50/75)*(75)^{1.06} * 1.06) + (25/75) * (75)^{1.06} * 0.85$
$= 24.57$ person months

Equation 6-11: *Imaging Management Constituent System SE Effort*

$= 38.55*[(1.0+0.10) * ((45/95)*(95)^{1.06} * 1.06) + (50/95) * (95)^{1.06} * 0.72] /152$
$= 29.74$ person months

Equation 6-12: *Laboratory Constituent System SE Effort*

$= 38.95*[(1.0+0.10) * ((55/65)*(65)^{1.06} * 1.06) + (10/65) * (65)^{1.06} * 1.02] /152$
$= 24.48$ person months

Table 6-3. SoSE Effort Equation Parameters[1]

Parameter	SoSE	SOA Infrastructure (New SoS Element)	Patient Management System	Pharmacy System	Telemetry System	Imaging Management System	Laboratory System
CS_{nonSoS}		0	30	60	25	50	10
$CS_{TreqSoSE}$		30	140	145	75	95	65
EM_{SOS-CR}	2.50						
EM_{SOS-MR}	0.47						
$EM_{CS-CRwSOSE}$		1.62	1.06	1.06	1.06	1.06	1.06
$EM_{CSnonSOS}$		n/a	0.72	0.67	0.83	0.72	1.02
OSF (minimal oversight of product vendors)	5%						
SoS_{CR}	130						
$SoS_{CSalloc}$		30	110	85	50	45	55
SoS_{MR} (sum of $CS_{non-SoS}$ * OSF)	8.75						
CS_{SoScap} (can range from 0-30%)		30%	10%	10%	10%	10%	10%
$SoST_{Treq}$	138.75						
A_r (Default Value: 38.55)	38.55	38.55	39.47	39.25	36.82	38.55	38.95
B_r (Default Value: 1.06)	1.06	1.06	1.06	1.06	1.06	1.06	1.06

Figure 21: Healthcare SoS Case Study2: original document and spreadsheet views. [Lane, 2009]

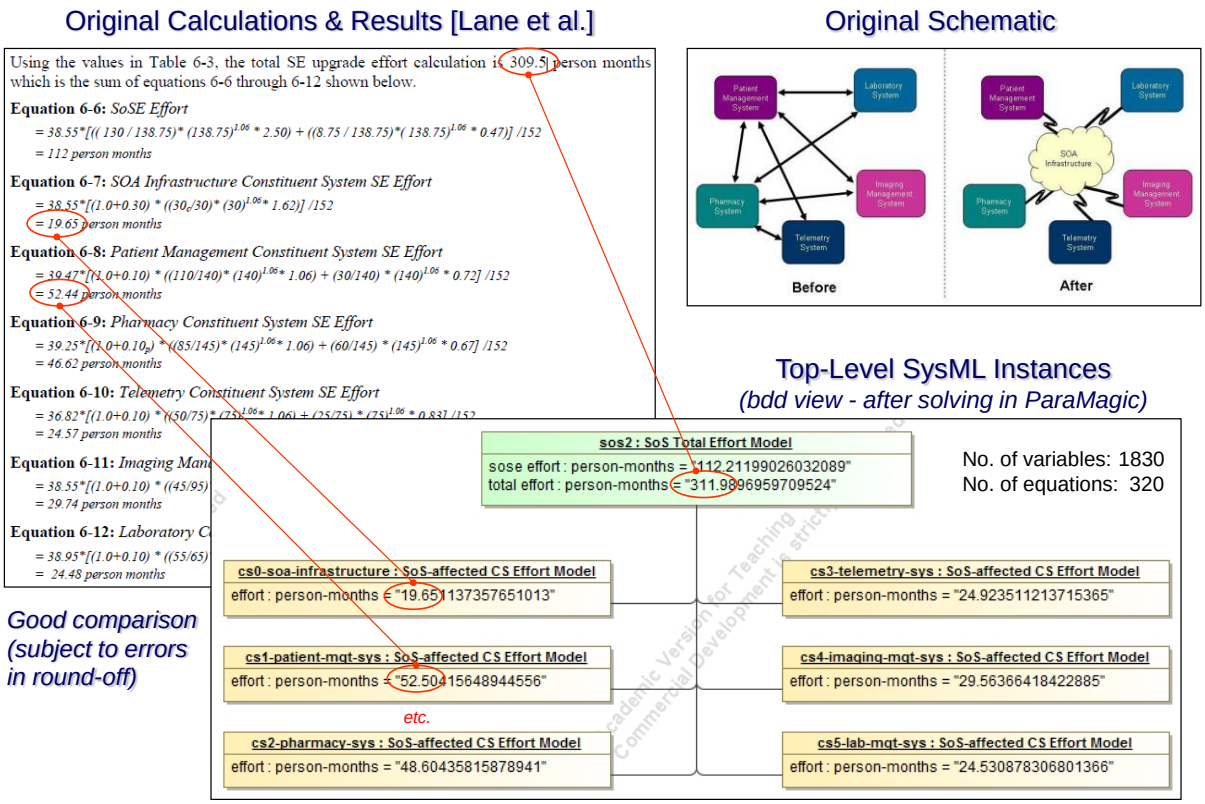


Figure 22: Healthcare SoS Case Study2: verification of the SysML model results compared to original results.

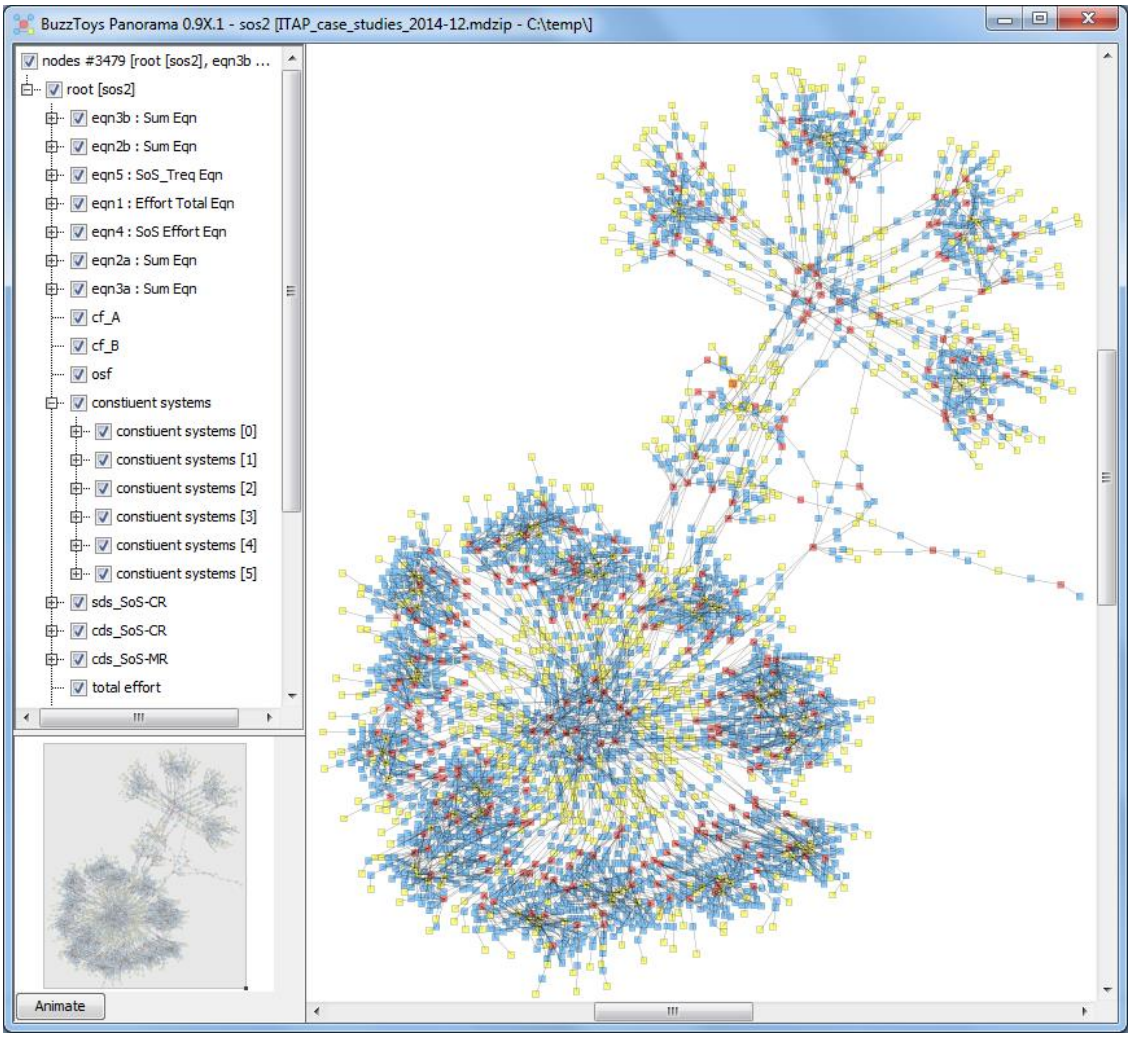


Figure 23: Healthcare SoS Case Study2: SoS-level DNA signature auto-generated from the SysML model.

The composite team has also explored several potential extensions to be considered for Stage 3 (CY2015) and beyond. One key extension deals with how to provide alternative interfaces to interact with these SysML-based cost models. SysML provides nice modeling capabilities with flexible structure and rich semantics. However, people who do not know SysML may find it challenging to interact with all of that power in its native SysML format. Therefore, several organizations have developed technologies over the past few years that connect SysML models to more traditional front-ends including interactive web pages and auto-generated documents. One of the most promising implementations is Open-MBEE[6], which NASA JPL has developed and uses internally on its projects. JPL has recently released this technology to the public. We have installed this environment at Georgia Tech recently and have learned the basics. During Stage 3 we will investigate creating interactive front-ends that are spreadsheet-like and easy-to-use, yet which maintain all the power of SysML building blocks behind them.

[6] https://github.com/Open-MBEE

Another extension area is additional capabilities that leverage the same cost and size driver building blocks we have already created. For example, previous research indicates that we should be able to support risk analysis (Madachy 1997, 2013), and other work has shown that total system cost can be derived from the systems engineering effort calculated by COSYSMO. Other envisioned applications include:

- Analysis of alternatives
  - Subsystem/component upgrades
  - Levels of capability option performance within SoS
  - Interoperability assessments for alternatives
- System/component retirement (or replacement) assessments
- Capabilities vs. costs

Furthermore, we have identified several case studies for potential future work. One pool of candidates involves simply taking existing SysML-based system models and wiring in the above cost modeling building blocks. For example, there is a hybrid SUV model described in the SysML spec itself that most vendors implement. We also have multiple SysML system models covering various domains that are available from MBSE/SysML-oriented academic classes at Georgia Tech. We have presented the Stage 1 work at several venues [Peak and Lane, 2014], and have had resulting discussions with several companies in the DoD supply chain who are interested to try it on their SysML-based projects. Note that DoDAF/UPDM-based models (which leverage SysML) are additional candidates, including the search & rescue model that is part of the UPDM spec. Two other specific case study candidates are illustrated in Figure 24.

Case: Military Operations SoS

Case: Emergency Response SoS

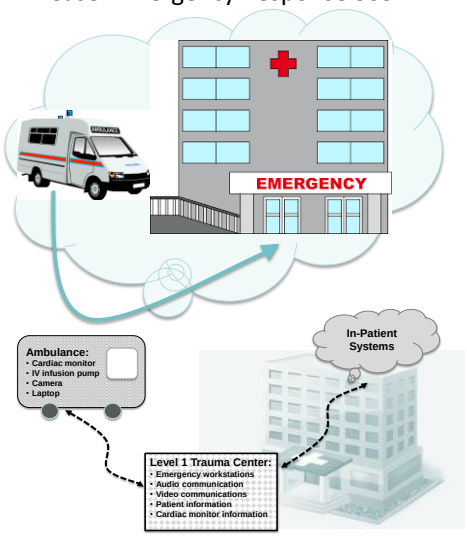


Figure 24: Candidate future case studies.

5.3.6 SUMMARY: SYSML COST MODELING

In summary, we have created cost modeling building blocks in SysML that leverage the COSYSMO-SoS/COCOMO legacy and experiences. We have successfully validated this knowledge capture via two healthcare SoS case studies: base complexity (Case Study1) and increased complexity (Case Study2). Benefits of this approach include the following:

- Enables better knowledge capture:
  - More modular, reusable, precise, maintainable, and complete (e.g., units).
  - Non-causal; better verification & validation vs. spreadsheets.
- Enables swapping in/out alternative subsystem designs.
- Provides patterns that are easy-to-apply with practically any system or SoS.
- Provides a basis to integrate with existing body of system models, including executable system models represented in SysML and/or DoDAF/UPDM. Methods to automate this integration are WIP in RT113/ITAP for Stage 3 (CY2015).

The above building blocks and case study applications effectively demonstrate how the basic MIM approach incorporates COSYSMO-SoS and related concepts in a modular SysML building block fashion. This then puts COSYSMO-SoS capabilities within the broader MIM context that can be built upon in future ITAP phases and tied together with FACT for larger-scale case studies involving multiple "–ility" considerations.

5.4 ACTIVITY 3. FUTURE PLANS: PHASE 4 AND 5

In previous ITAP phases and on the basis of past lessons learned from other, customer-specific toolsets, GTRI has identified and begun maturing the foundations of a design space environment and integrated workflow to aid in the investigation of –ility formalisms. Building on existing formalism definitions from across the ITAP team, GTRI is further investigating methods whereby we may operationalize these formalisms into measureable and executable constructs to support Pre-Milestone A analysis. The work proposed for Fiscal Years 2015 and 2016 seeks to further mature these methods, processes, and tools (MPTs) to specify operationally relevant dimensions for alternative analysis of early-stage DoD system designs.

Throughout the development process and method inclusion, this effort and its future maturation will seek to preserve an open framework and approach that promotes quantitative and qualitative transparency of the tradespace refinement. Our goals are to ensure that the workflow and toolset support easy inclusion of analytical constructs that may be developed on other ITAP efforts to evaluate different –ilities in different ways. A key to synthesizing these constructs will be scalability, flexibility, and modularity of the construct as well as the workflow and processes we are striving to integrate. Our hope is that this philosophy will lead to more effective collaboration and traceability while offering a capability to both refine and synthesize research constructs for complex tradespace evaluation.

Specifically, for CY 2015 and CY2016, GTRI proposes to mature the previous work under ITAP in two primary directions: 1) Maturation of the methods and constructs to analytically execute formalisms, and 2) Maturation of the processes and tools that help operationalize these constructs in a scalable and traceable manner. For the first, we seek to add the modeling and analysis dimension required to evaluate and produce environmentally dependent system attributes. Because the model of a system alone is insufficient to produce all quantitative system attributes important to the decision making process, this step will enable us to evaluate different environment influences relevant to fielded operational performance.

There are many –ilities being investigated across the ITAP teams that could benefit from this capability. Context-dependent notions of risk and identification of feasible sets as being investigated by WSU are good examples. Other –ility metrics include versatility, agility, functional persistence, etc. as investigated by AFIT, USC, MIT, and others. A key to the usefulness and broad applicability of incorporating this dimension is how to achieve the link to environmentally dependent analytical blocks to help complete a tradespace generation. This must also adhere to concepts of modularity, flexibility, and scalability to support tradespace analyses.

Including the environmental dimension (or rather, the capability to include many different environmental dimensions) is also directly in line with DoD Concept of Operations (ConOps) needs. A ConOps is essentially a type of requirements document. It describes the mission of the system, its operational and support environments, and the functions and characteristics of the system within an overall operational environment[7]. A ConOps therefore communicates a story in terms of needs from the users' point of view at a given point in the life cycle. In the Pre-Milestone A stage, evaluating anticipated system performance against various ConOps helps promote early-stage development specification. In turn, this identifies a much smaller set of design alternatives to carry forward for more rigorous and (typically) data-intensive evaluations.

To evaluate the performance of how we incorporate the environmental dimension in the workflow and toolset, GTRI proposes to mature the Needs Context construct from Phases 2 and 3. The Needs Context already adds a dimension to traditional utility analysis. Instead of one utility per design alternative, there are as many utilities as there are Needs Contexts, which are representations of requirements across stakeholders and/or operational needs. However, we have not yet been evaluating conditions under which system attributes may be expressed differently due to distinct operational environment conditions. When we add the environment dimension, we may, for example, have three different values of a system performance attribute for a single system design alternative. And, the different values may be arrived at through different "mapping modules" that calculate environmentally dependent system attributes. The resulting tradespace may be thought of as partitioned:

{ Intrinsic system attributes |

[7] ANSI/AIAA G-043- 1992, "Guide for the Preparation of Operational Concept Documents"

Extrinsic attributes dependent on Environment 1 |
Extrinsic attributes dependent on Environment 2 | etc. }.

This will occur because a system attribute may capture a concept, have the same name, and yet be evaluated differently (by distinct mapping equations) and expressed at different quantitative levels for a single design alternative.

Instead of aggregating the various expressions of a given system attribute that may arise however, we seek to preserve the different performance information. Where the existing Needs Context construct produces an aggregated utility, the environmentally dependent system attributes will not be aggregated.

Key to successful realization of the above goals, are the supporting computational methods, processes, and tools that also require additional research effort. Partitioned tradespaces are high dimensional, nontrivial analysis problems that may require significant computational time to detect the set of ideal alternatives in the tradespace. To reduce the analysis effort required, the successful implementation must be scalable and flexible. Scalability of these methods includes designing algorithms that are tractable for high dimensional problems as well as being able to be executed in a parallel computing environment. The implementation should also be flexible so that it may be reused for different problem domains. This can be partly accomplished by understanding the algorithms needed and implementing the salient portions in a modular fashion.

<u>FY 2015</u>
Specifically, for FY 2015, we propose to investigate the MPTs as described above to add the environmental dependence and ability to produce and evaluate a partitioned tradespace.

<u>FY 2016</u>
For FY 2016, we hope to expand this concept yet again to consider variance within a single environment. For example, a ground vehicle may be designed to operate with certain capabilities in a desert environment. This is usually evaluated at some nominal conditions for that environment. In reality, however, there are variations within that environment that can be extreme such as during monsoon season. Any ground vehicle in such a regions would need to perform under all conditions in that environment that occur with any degree of significance. We therefore propose to develop the framework and MPTs to allow us to evaluate across single environment distributions in addition to the multiple environment study proposed for FY 2015.

5.5 REFERENCES

1. Assistant Secretary of Defense (Networks and Information Integration) and DoD Chief Information Officer (2007). DoD Information Assurance Certification and Accreditation Process (DIACAP), Department of Defense Instruction 8510.01.
2. Assistant Secretary of Defense (Networks and Information Integration) and Chief Information Officer (2009). Clarifying Guidance Regarding Open Source Software (OSS), Department of Defense Memorandum.
3. Beesemyer, J. , A. Ross, and D. Rhodes (2012). "Case Studies of Historical Epoch Shifts: Impacts on Space Systems and their Responses." In *AIAA Space*.
4. Culler, M. (2010) Modeling Product Life Cycle Networks in SysML with a Focus on LCD Computer Monitors. Master's thesis, GW Woodruff School of Mechanical Engineering, Georgia Tech, Atlanta.
5. Hassine-Guetari, Soumaya Ben, Jérôme Darmont, and Jean-Hugues Chauchat (2010). "Aggregation of data quality metrics using the Choquet integral." *Aggregation of data quality metrics using the Choquet integral*.
6. Browne, Daniel, Robert Kempf, Aaron Hansen, Michael O'Neal, and William Yates (2013). "Enabling Systems Modeling Language Authoring in a Collaborative Web-based Decision Support Tool." *Procedia Computer Science* 16: 373-382.
7. Charnetski, Johnnie R., and Richard M. Soland (1976). "Technical Note—Statistical Measures for Linear Functions on Polytopes." *Operations Research* 24, no. 1: 201-204.
8. Charnetski, Johnnie R., and Richard M. Soland (1978). "Multiple-attribute decision making with partial information: The comparative hypervolume criterion." *Naval Research Logistics Quarterly* 25, no. 2: 279-288.
9. Christel, Michael G., and Kyo C. Kang (1992). *Issues in requirements elicitation*. No. CMU/SEI-92-TR-12. Software Engineering Institute, Carnegie Mellon University, Pittsburgh, PA.
10. Ender, Tommer R., Daniel C. Browne, William W. Yates, and Michael O'Neal (2012). "FACT: An M&S Framework for Systems Engineering." In *The Interservice/ Industry Training, Simulation & Education Conference (I/ITSEC)*, vol. 2012, no. 1. National Training Systems Association.
11. Fricke, E., & Schulz, A. P. (2005). Design for changeability (DfC): Principles to enable changes in systems throughout their entire lifecycle. Systems Engineering; 8(4).
12. Fu, Guangtao, Hall, J., Lawry, J. (2005). "Beyond probability: new methods for representing uncertainty in projections of future climate." *Tyndall Centre for Climate Change Research Working Paper* 75.
13. Goerger, S.R., Madni, A.M., and Eslinger, O.J. (2014). "Engineered Resilient Systems: A DoD Perspective," *Conference on Systems Engineering Research (CSER 2014)*, Procedia Computer Science; 28, 865-872.
14. Grabisch, Michel (1996). "The application of fuzzy integrals in multicriteria decision making." *European journal of operational research* 89, no. 3: 445-456.
15. Grabisch, M., and Labreuche, C. (2010). "A decade of application of the Choquet and Sugeno integrals in multi-criteria decision aid." *Annals of Operations Research* 175, no. 1: 247-286.

16. Grabisch, M., and Roubens, M. (2000). "Application of the Choquet integral in multicriteria decision making." *Fuzzy measures and integrals* 40: 348-375.
17. Hazelrigg, George A. (1998) "A framework for decision-based engineering design." *Journal of mechanical design* 120, no. 4: 653-658.
18. Lahdelma, R., Hokkanen, J., Salminen, P. (1998). "SMAA-Stochastic multiobjective acceptability analysis." *European Journal of Operational Research* 106, no. 1: 137-143.
19. Lahdelma, R., Miettinen, K., Salminen, P. (2003). "Ordinal criteria in stochastic multicriteria acceptability analysis (SMAA)." *European Journal of Operational Research* 147, no. 1: 117-127.
20. Lane, J.A. (2009) Cost Model Extensions to Support Systems Engineering Cost Estimation for Complex Systems and Systems of Systems. 7th Annual Conference on Systems Engineering Research (CSER), Loughborough.
21. Lane, J.A. (2010) Evolution of Systems Engineering Cost Estimation. Chapter 6 (unpublished draft) for cost modeling book. http://csse.usc.edu/
22. Madachy, R.J. (1997) Heuristic Risk Assessment Using Cost Factors. *IEEE Software*.
23. Madachy, R.J. (2013) Expert COSYSMO - Systems Engineering Cost Model Risk Advisor, http://csse.usc.edu/tools/ExpertCOSYSMO.php (as accessed Nov 2013)
24. O'Neal, M., Ender, T.R., Browne, D.C., Bollweg, N.B., Pearl, C.J., Brico, J.L. (2011). "Framework for Assessing Cost and Technology: An Enterprise Strategy for Modeling and Simulation Based Analysis." In *MODSIM World 2011 Conference and Expo*, Virginia Beach, VA, October 14.
25. OpenMDAO; 2014. http://openmdao.org/
26. Parnell, G.S., Jackson, J.A., Burk, R.C., Lehmkuhl, L.J., Engelbrecht, J.A. (1999). "R&D concept decision analysis: using alternate futures for sensitivity analysis." *Journal of Multi-Criteria Decision Analysis* 8, no. 3: 119-127.
27. Peak, R.S. and Lane, J.A. (2014) SysML Building Blocks for Cost Modeling: Towards Model-Based Affordability Analysis. INCOSE International Workshop (IW14), Torrance, California.
28. Peak, R.S., Paredis, C.J.J., McGinnis, L.F., Friedenthal, S.A., Burkhart, R.M., et al. (2010) Integrating System Design with Simulation and Analysis Using SysML. INCOSE MBSE Challenge, Modeling & Simulation Interoperability (MSI) Team, Phase 2 Final Report (v2.1). *http://www.pslm.gatech.edu/projects/incose-mbse-msi/*
29. Romaniw, Y., Bras, B., Guldberg, T. (2011) Sustainable Manufacturing Analysis using Activity Based Costing in SysML. ASME IDETC/CIE, Washington DC.
30. Romaniw, Y. and Bras,B. (2010) Sustainable Manufacturing Analysis using an Activity Based Object Oriented Method. SAE Journal of Aerospace 2(1) 214-224.
31. Ross, A., and Rhodes, D. (2008). "Architecting Systems for Value Robustness: Research Motivations and Progress," 2nd Annual IEEE Systems Conference, Montreal, Canada.
32. Ross, A.M., and Rhodes, D.H. (2008). "Using natural value-centric time scales for conceptualizing system timelines through epoch-era analysis." In *INCOSE International symposium*.
33. Ryan, E. T., Jacques, D. R., & Colombi, J. M. (2013) An ontological framework for clarifying flexibility-related terminology via literature survey. Systems Engineering; 16(1), 99-110.

34. Schulz, A.P., and Fricke, E. (1999). "Incorporating flexibility, agility, robustness, and adaptability within the design of integrated systems-key to success?." In *Digital Avionics Systems Conference, 1999. Proceedings. 18th*, vol. 1, pp. 1-A. IEEE.
35. Sitterle, V., Curry, M., Freeman, D., and Ender, T. (2014). "Integrated Toolset and Workflow for Tradespace Analytics in Systems Engineering," 24th Annual International Council on Systems Engineering (INCOSE) Symposium, Las Vegas, NV.
36. Tamburini, D.R., Peak, R.S., Paredis, C.J.J. (2005) Composable Objects (COB) Requirements & Objectives v1.0. Technical Report, Georgia Tech, Atlanta. *http://eislab.gatech.edu/projects/nasa-ngcobs/*
37. Tervonen, T., and Lahdelma, R. (2007). "Implementing stochastic multicriteria acceptability analysis." *European Journal of Operational Research* 178, no. 2: 500-513.

6 PENNSYLVANIA STATE UNIVERSITY

6.1 PAST RESULTS: PHASES 1 AND 2 RESULTS

Under Phase 2, PSU has developed a preliminary formal model of design as a sequential decision process of initially considering broad ranges of potential solutions to the design problem at a low level of detail, and then sequentially reducing the space of design solutions considered while increasing the detail. More specifically, we consider how the computational models used to support decision making evolve from simple models for early conceptual analyses (e.g., Excel spreadsheets running on desktop computers) to extraordinarily detailed models running on supercomputers (e.g., coupled aero and structural analyses). Key considerations in establishing a workflow (and addressed in this work) include the nonrecurring cost to create the models, the recurring cost to exercise them across a trade space, and the reduction in trade space that can be achieved with the models. As each level of modelling requires different levels of detail, the rate of increase of detail is also critical. Finally, how the models couple to form a sequential chain of analyses is considered.

The model of decision making developed in this work has its roots in the domain of marketing, which is fundamentally concerned with how consumers choose. While there are many different approaches to modelling the consumer's choice process, a common thread is that the consumer goes through a process of sequentially reducing the space of considered choices through a number of discrete sets. Shocker et al. [5] define a model that has been widely adopted in the field, directly informing our efforts (see Fig. 1). The initial set is the *universal set* and is the set of all possible choices. The *awareness set* consists of the subset of items for which the DM becomes aware. It is a subset of the universal set and is likely a strict subset. A *consideration set* is a purposefully constructed set of potential solutions – so creating this set took effort by the DM. The idea of a consideration set is a critical one, as it reflects the concept that the DM makes a preliminary choice in forming the consideration set prior to a final choice. The *choice set* is defined as the final consideration set – it is the set of alternatives that are considered prior to a final choice. Although only one consideration set is shown in Figure 25, there can be many intermediate sets, each smaller than the predecessor and subject to more scrutiny. The strategies for forming each set is expected to evolve adaptively based on the numbers of alternatives and attributes as per adaptive strategy selection research [14-16].

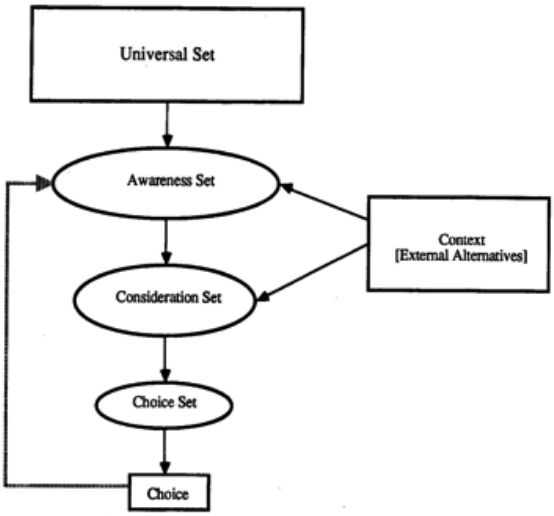


Figure 25. Model of decision making in marketing

This choice model has since been extended by the authors to explicitly include the effort to use computational modeling to guide a DM (Decision Maker) in their search through the trade space [9]. The basic premise is that DMs and their team start with low fidelity designs subjected to low fidelity analyses. This phase identifies regions of interest and culls other regions from further consideration using some heuristic such as compensatory or non-compensatory [1]. The process is repeated while subjecting the reduced size set to further and further analyses. The final choice set is analyzed at the maximum fidelity. Figure 26 shows an intermediate step in the larger convergence process.

In executing a TSE (Trade Space Exploration) exercise and selection process against a particular consideration set, the DM down-selects to a smaller consideration set thus forming a decision epoch — their final decision is the single choice. The entire process can be viewed as a *series of decision epochs*, with each epoch incurring an allocation of time and resources resulting in a smaller consideration set for the next stage. Each epoch retains a set Graphically, the core product of phase 2 is the design flow in the figure below.

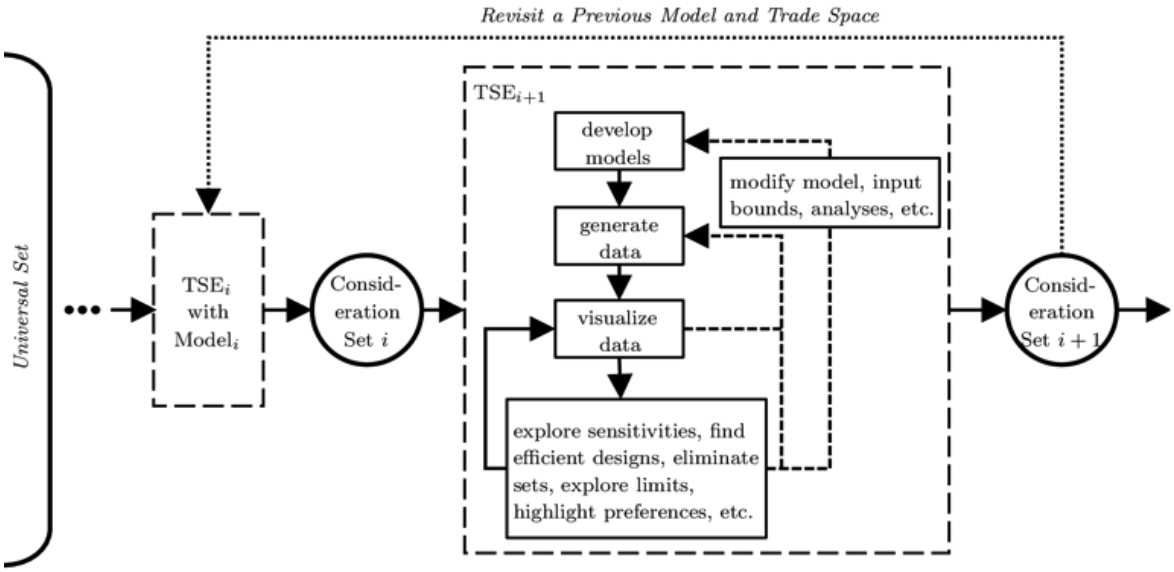


Figure 26: Core Produce of Phase 2 Design Flow

6.2 PHASE 3 RESULTS

PSU has developed a preliminary formal model of design as a sequential decision process of initially considering broad ranges of potential solutions to the design problem at a low level of detail, and then sequentially reducing the space of design solutions considered while increasing the detail. More specifically, we consider how the computational models used to support decision making evolve from simple models for early conceptual analyses (e.g., Excel spreadsheets running on desktop computers) to extraordinarily detailed models running on supercomputers (e.g., coupled aero and structural analyses). Key considerations in establishing a workflow (and addressed in this work) include the nonrecurring cost to create the models, the recurring cost to exercise them across a trade space, and the reduction in trade space that can

be achieved with the models. As each level of modelling requires different levels of detail, the rate of increase of detail is also critical. Finally, how the models couple to form a sequential chain of analyses is considered.

The core of the modelling problem and analysis approach adopted here is that we start with a fixed, finite of discrete points defining the tradespace, the *consideration set*. Each discrete point is defined by a set of *n* parameters, thus the set forms a cloud of points in an *n*-dimensional tradespace. No restriction is put on the form of the parameters other than that we can form a rank ordering over them. So a parameter can be anything from the real number line to discrete numbers, to choices of colors, as long as they can be ordered.

Models/analyses are applied to points in the tradespace, with the results used to decide which tradespace points to carry forward and which ones to eliminate from further consideration. Early models and analyses either do not incorporate all of the physics of the problem or do not consider all of the parameters of the points, i.e., they consider a reduced level of detail of the tradespace points. For this reason, reduced order models are incapable of completely reducing a tradespace to a final solution. Instead, they can only decrease the size of the consideration set. The remaining consideration set is then subject to further analysis by more expensive modelling efforts that consider more parameters and more physics, resulting in further decrease in the size of the set. In the final model/analysis stage, a choice of a single point in the tradespace is made.

In the flow from a conceptual level model to a detailed model, we constrain the conceptual level model to *only remove points that are guaranteed to not be the final choice when run through the final detailed model*. Since the concept level model will be analysing trade points based on fewer numbers of parameters, the requirement that points are guaranteed to be removed in more detailed analyses tightly couples the levels of analyses. How the concept model is used is heavily dependent on what the detailed model is and how it will be used. They have to be considered as a flow.

Within this flow, then, all of the modelling stages prior to the final one serve only to cull the tradespace. A particular level of model, when applied to either the original consideration set or a reduced subset of it, will only remove the points in the set it is capable of removing. Each model has a certain *discriminatory power* which is tied to the total number of points from the original consideration set that it will remove. From a cost-of-modelling perspective, a reasonable expectation is that the greater the discriminatory power of a model, the greater the recurring and non-recurring costs of its use.

As this work is preliminary in nature, assumptions used in order to focus on the key aspects of the modelling approach include

- Discrete pre-determined sets of design points
- All points described by the same set of parameters
- The concept of discriminatory power and the restricted ability of models to remove design points

- No uncertainty in the models (such as could be modelled by a probability distribution)

Nomenclature:

Z_i	the ith consideration set of designs
q_i	size of the ith set $\|Z_i\|$
M_i	The ith modelling effort used to analyse a consideration set and cull it to a smaller set
X	vector input to a concept model
Y	vector of input to a detailed model
$g_c(x)$	concept model
$g_d(x,y)$	detailed model

6.2.1 FORMAL MODEL OF THE SEQUENTIAL PROCESS

Start with a random list that represents finite parameter trade space, Z, and where each element of Z corresponds to a parameter of a design. An example would be spherical tank design where the three parameters are radius, wall thickness, and material choice and so Z = [radius, thickness, material] and each of the parameters can take values from some set of possibilities.

The space of *Z* has an ideal point z^*. The ideal point is the point that would be chosen if there were time and ability to analyse every point in *Z* with the most detailed modelling effort. We apply a "modelling effort" *M* to *Z* and this refines and localizes the position of z^*. The modeling effort can be more or less discriminatory, with differing results for each case. So if we have efforts M_1 and M_2 and M_2 and is more discriminatory, then using M_2 results in a smaller next set as compared to using M_1.

Modelling efforts also have cost. The cost of an effort is decomposed into a fixed cost for initially establishing a model and a variable cost proportional to the size of the set Z that the model is applied to. We will use a linear approximation to cost, so cost c_i of model M_i is $a_i + b_i|Z_i|$, where $|Z|$ is the size of the set Z. Label the size of the set as q_i, then

$$c_i = a_i + b_i q_i \quad .$$

The simple linear cost model neglects factors such as learning curves, where the cost per analysis should decrease with each analysis conducted. It also neglects sustainment costs merely for having a model in hand, such as annual licensing costs for software and hardware. These can be added to increase accuracy.

6.2.1 The Modeling Effort M

The modeling effort *M* is all of the cost and time to define designs and to identify and acquire computational models to the point of being able to use them, the time and effort to train the people in using the tools, along with the cost and time to exercise the models, analyze the results, and down-select the trade space. The discriminatory power of the modeling effort is a function both the tools used to execute it and the people executing the effort. So for example the same toolset used by senior users empowered by the final decision maker would likely have more discriminatory power than a group is less trusted users. In summary, the modeling effort includes a technology element for populating the tradespace and visualizing it, and a human element for analyzing the results and making a decision. The model has a fixed and variable cost. Furthermore, the model has a discriminatory power, which is reflected in its ability to localize the ideal point z^*.

An important restrictive modeling assumption adopted for this preliminary work is that applying a model to a space reduces the space to a subset of the original *q*, not by a percentage of the remaining q_i. Therefore, there are no gains to be realized by repeatedly exercising the same model. Similarly, the size of the set realized by running a model is independent of set it is run over, unless the set's size is smaller than q_i. This is a restrictive assumption that can be relaxed and modified in later models.

6.2.2 SINGLE STAGE MODELING VERSUS TWO STAGE MODELING

Look at the case where we have two efforts, M_1 and M_2, assume that M_2 is of the quality that applying M_2 to all of Z will identify exactly z^*. The effort M_1 will not identify exactly where z^* is, but will cull the space of Z to some smaller region Z_1 where $|Z_1| < |Z|$, or equivalently $q_1 < q$. If M_1 is run against Z first, then when M_2 is run, it need only be run over Z_1 since we know that z^* does not lie in the set $Z - Z_1$. This results in a cost savings in running M_2 since the run cost depends on the size of the space to run it on. This is the fundamental key to executing the multi-step modeling, the reduction in size of the set considered by the next stage of analysis.

So what must hold with regards to the costs of M_1 and M_2 and the reduction in set size by M_1 in order to justify using a two-step process? The cost of a single step process is $c = a_2 + b_2 q$. The cost for a two-step process is

$$\begin{aligned} c &= a_1 + b_1 q + a_2 + b_2 q_1 \\ &= (a_1 + a_2) + b_1 q + b_2 q_1 \end{aligned}.$$

Solving for q_1 results in the following relationship:

$$q_1 \pounds \frac{(b_2 - b_1) q - a_1}{b_2}.$$

So M_1 needs to restrict the space to q_1 or less in order to justify using it in a two-stage process.

6.2.3 MULTI-STAGE PROCESS

The sequential process can have arbitrarily many modeling steps. Consider the case of having three potential models to employ. There are four potential configurations to consider (note that all four sequences end with M_3):

$$M_3$$
$$M_1M_3$$
$$M_2M_3$$
$$M_1M_2M_3$$

In general, if there are *n* models to choose from then there is an upper bound of 2^{n-1} possible configurations to consider. This upper bound is loose, however, and can be tightened considerably. Considering all of the alternatives, the cost of each is the following:

$$c_3 = a_3 + b_3q$$
$$c_{13} = a_1 + a_3 + b_1q + b_3q_1$$
$$c_{23} = a_2 + a_3 + b_2q + b_3q_2$$
$$c_{123} = a_1 + a_2 + a_3 + b_1q + b_2q_1 + b_3q_2$$

The alternative paths can be laid out as a graph through a state space, with q_i's as states and the models used as arcs in the graph.

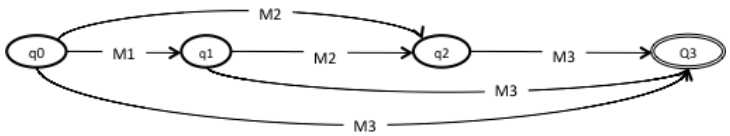


Figure 27. Sequencing of models

Adopting the nomenclature $M_iM_j \rightarrow M_{ij}$, the best modeling approach to get to state q_3 can be calculated as

$$M_{123} = \min\left[M_1M_3, M_{12}M_3, M_3\right]$$

while the best modeling approach to get to state q_2 is

$$M_{12} = \min\left[M_1M_2, M_2\right]$$

and the only approach to get to q_1 is to use M_1. In general, if there are *N* models, then there are potentially 2^{N-1} configurations to choose from, but one only needs to actually consider $N+\cdots+3+2+1=(N^2+N)/2$ possible values, which can be written as

$$\text{number of combinations} = \begin{pmatrix} N+1 \\ 2 \end{pmatrix}.$$

6.2.4 EXAMPLE: WING DESIGN FOR LIGHT CIVIL AIRCRAFT

The performance of a wing is primarily a function of its geometry, which for typical light civil aircraft can be characterized by its airfoil properties, span, and chord. Given a wing that uses the same airfoil shape throughout, and the 2-D airfoil has a known rate of change a of c_l with respect to angle of attack a of $c_l = a_l a$, then a first order approximation to the behavior of the 3-D wing is the relationship

$$a_w = \frac{a_l}{1 + \frac{a_l}{pe\text{AR}}}$$

where e is the Oswald coefficient and AR is the aspect ratio, which is based on the wing span b and the wing area S. This model only has the variables b and S; and the equation to determine a_w is in simple closed form. It can be considered a concept model.

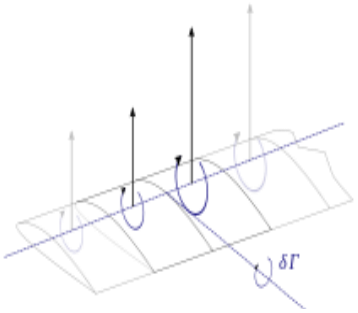


Figure 28: Wing Segment

improved model of wing performance can achieved by using *lifting line theory* [17] to model the wing is a series of finite 1-D wing segments with vortices from each segment influencing the other (**Figure 28**). This improved modeling approach involves bo... ...design variable to be specified (the wing chord for each segment) and the computationally complexity (the algorithm solves a linear system of size equal to the number of segments the wing is divided into). Whereas the concept model was a simple closed form equation, a lifting line model dividing the wing into m segments requires solving a set of m linear equations with a typical algorithmic complexity $O(n^3)$. Additionally, the wing chord at the m locations must be specified. The ability to choose the discretization m offers a relatively smooth continuum of models of increasing detail to consider.

We define the initial consideration set Z to be a set of 5000 different wing designs, each with a unique choice of 2-D airfoil (setting a_l), a fixed span b and a set of 200 chord values along the wing. We can arbitrarily choose three levels of models to consider, the concept model, a lifting line model with m=20 unique wing segments, and a lifting line model with m = 200 segments, i.e., maximum detail.

Table 9. Models of wing to form a sequence from

Model	Description	Variables	Complexity	Accuracy
M_1	Closed form equation	a_l, b, S	*O(1)*	Low
M_2	Lifting line with 20 chord segments	a_l, b, $chord_1,\ldots,chord_{20}$	$O(20^3)$	Medium
M_3	Lifting line with 200 chord segments	a_l, b, $chord_1,\ldots,chord_{200}$	$O(200^3)$	Final

Assuming that the design problem consists of finding some ideal combination of wing aerodynamic performance with mass, cost, aircraft space requirements, etc., we would want to use the cheaper models as much as possible to remove as many of the designs in *Z* as possible prior to running the more expensive models. All model sequences must end with M_3, however, as only it can discriminate between the wing designs.

6.2.5 Connection between Concept & Detailed

Consider the simple wing model above, and the relationship between M_1 and M_2. The concept model only has the three input variables, while the (relatively) detailed model has 21. There are potentially many wing designs in Z that are unique with respect to M_2 but have the same three values of input with respect to M_1.

A similar property holds between M_2 and M_3. We say that to fully describe a wing requires setting 201 attributes. When a wing has fixed only the 20 chord values needed for M_2, we in essence have fixed 21 of the 201 attributes that fully describe a wing, and *have 180 degrees of freedom remaining*. The 180 degrees of freedom reflect on deferred decisions. When we run M_2, we are leaving the other 180 variables free to take a value to be determined later. This is in a way uncertainty, but it is fundamentally different from the uncertainty associated with unmodeled physics or random events, and cannot appropriately be modeled by a probability distribution. Instead, it should be modeled by an interval. The formal model of the relationship follows.

6.2.7 Formal Model of Relationship between Concept & Detailed

Assume there is a detailed model and a conceptual model. The detailed model has as input a set of vector-valued variables *x* and *y* from discrete sets of parameters $X = \{x_1,\cdots,x_N\}$ and $Y = \{y_1,\cdots,y_M\}$ and returns a scalar value v that is to be minimized,

$$v = g_d(x,y) \ .$$

The complementary conceptual model only takes as input *x* and that has the following form:

$$v = g_c(x) \ .$$

The decision problem is to find some optimal choice of *x* and *y* satisfying

$$v^* = \min_{x,y} g_d(x,y)$$
$$x^*, y^* = \arg\min_{x,y} g_d(x,y)$$

Further assume that the detailed model is expensive to exercise, and that finding the optimum requires an exhaustive search of all possible values for *X* and *Y*. In order to reduce the search time, we'd like to use the lower order (conceptual) model to cull as much of the search space as possible prior to running the detailed model. The complementary conceptual model that has the following form

$$v_c = g_c(x)$$

The concept model takes as input a subset of the inputs to the detailed model. The goal is to *use the concept model to eliminate values of X that are guaranteed not to contain* x^*. An example ideal concept model would be implemented as (or be equivalent to)

$$g_c(x) = \min_y g_d(x,y) \ .$$

Such a model would identify x^* directly, leaving only a search over *Y* to complete the optimization. However by our problem definition this would require actually running the detailed model exhaustively over the space of possible *Y* values for each *X*, which would result in exactly the same number of evaluations as running the detailed model and optimizing over *X* and *Y* in the first place.

Typically concept models are lower order models and almost always taking as input a subset (*X*) of the total number of inputs that would run in a detailed model, meaning they do not contain all of the details of relationships that are in the detailed model. Much work has been done on "model uncertainty" but in this instance uncertainty is not the correct term, since what really remains when we pick a value from *X* is interval, i.e., there are remaining degrees of freedom as we still need to ultimately pick a value from *Y*.

Instead of a single concept model, we propose to have a concept interval returned by the conceptual model.

$$v_l = g_l(x)$$
$$v_u = g_u(x) \ .$$

For a particular value of *X*, the upper and lower bounds have the property $g_l(x) \le \min_y g_d(x,y) \le g_u(x)$. So the concept model forms an upper and lower bound on the best possible value achievable with g_d given *X* is fixed. The bounds define an interval in which the ideal point x^* must lie. The concept model(s) can then be used to exclude regions of *X* that x^* is guaranteed not to lie.

$$\forall\, x_i \in X:\ \text{if } \exists\, x_j \in X \text{ s.t. } g_l(x_i) > g_u(x_j) : x_i \notin \text{consideration set}$$

In plain English, for a point x_i if there is another point x_j such that the best possible result of $g_d(x_i,y)$ is worse than the worst possible result with $g_d(x_j,y)$ then we know that $x_i \neq x^*$, i.e., we can remove it from the consideration set of candidates for x^*.

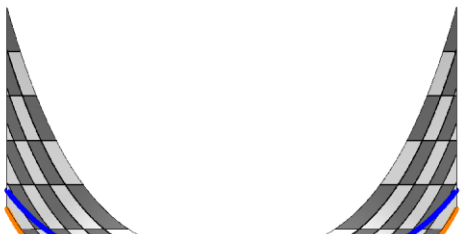


Figure 29. Model value space

Figure 29 shows a graphic example of the principle. The grey region is the projection of all values of $v = g_d(x,y)$, while orange curve and the blue curve are the lower and upper bounding curves respectively on the best possible value of $g_d(x,y)$ for a given fixed *x*.

Reconsidering the wing design problem, let each of 201 parameters of $\{a_l, \text{chord}_1 \cdots, \text{chord}_{200}\}$ take one of 10 possible values. So there are potentially 10^{200} points in the initial consideration set Z if we were to explicitly enumerate all possible combinations. Treating the models M_2 and M_3 as concept and detailed respectively, then

$$X = \{a_l, \text{chord}_1 \cdots, \text{chord}_{20}\}$$
$$Y = \{\text{chord}_{21} \cdots, \text{chord}_{200}\} \quad .$$

The model M_3 takes in all of M_2's variables plus additional ones. For M_2 to be a proper conceptual model with respect to M_3 it should return two values,

$$M_{2_l}(x) \leq \min_y M_3(x,y) \leq M_{2_u}(x) \quad .$$

6.3 Future Plans: Phases 4 and 5 plans

This effort contains two key contributions to the design community; a model of design as a sequential decision process of refinement using progressively more accurate and expensive models, and a connection approach for how conceptual models couple with detailed models. A number of assumptions were used to simplify developing and understanding the modeling approach. One was that a discrete pre-determined set of design points formed the consideration set *Z*. However, even in the simple wing problem the size of *Z* would be 10^{200}, which could never be exhaustively enumerated. There are no obvious hindrances to extending the modelling approach to continuous spaces. Another assumption is that all points are described by the same set of parameters. An obvious counter-example is where a concept model for a ship is created in an Excel spreadsheet with a fixed set of parameters (*X*) but then a solid geometry model is created for the detailed design, which differ from other detailed design by gigabytes of data. The key is the fact that after $x \in X$ has been specified, many degrees of freedom remain, and do the concept model should return ranges, not single values.

The assumption that models have a discriminatory power and a restricted ability of models to remove design points aligns well with the idea that trade spaces are culled at least initially using non-compensatory strategies such as applying requirements/constraints to candidate designs and removing them if they violate. The final assumption is that there is no uncertainty in the models. There is no inherent reason that uncertainty cannot be included - the purpose in this assumption is just to clearly highlight that the models need to provide bounds and intervals as output rather than points due to the remaining degrees of freedom of the decision process. Including uncertainty would have the impact of making the upper and lower bounds probabilistic rather than deterministic, otherwise the fundamental approach remains the same.

The next step in research is to identify candidate models across many domains and attempt to categorize their recurring and nonrecurring costs and discriminatory power, forming them into

sequences and exploring methods to create optimal model flows. Methods to easily create the upper and lower concept models are also a near-term target. Last is the introduction of multiple objectives into the mix, which will greatly complicate the definition of an upper and lower bound, replacing a simple greater/less than relationship with a dominance one.

6.3.1 PHASE 4 PLANS

In Phase 4, PSU will focus their efforts on validating the Sequential Decision Process model against a series of case studies of increasing complexity. The case studies will in turn drive the next steps in exploration. We will implement the simple wing design model with varying number of wing segments from 10s to thousands, and will treat it as both a value optimization problem (linear-weighted sum or other scalar function) and as a multi-objective problem with Pareto optimality as the defining metric. Intent is to publish at the ASME International Design Engineering Technical Conferences, August 2015, and submit to the journal our results.

Under separate funding, PSU will have opportunity to deploy the techniques developed here to both the US Army *Capital Planning Model* effort and the *DOD Rapid Composition of Cost Models* effort. These complementary efforts will offer the first opportunity to prove out the techniques on practical problems.

6.3.2 PHASE 5 PLANS

With phase 4 focused on validation and extension, Phase 5 will be focused on deployment. PSU will develop questionnaire templates to help organizations understand their sequential modeling process, along with computational tools to help them understand the principles of fidelity increase versus trade space decrease. Finally, we will deploy decision tools to help program managers and system engineers plan the Model-Based System Engineering processes

6.4 REFERENCES

1. Payne, J.W., J.R. Bettman, and E.J. Johnson, The adaptive decision maker. 1993: Cambridge University Press.
2. Busemeyer, J.R. and A. Diederich, Survey of decision field theory. Mathematical Social Sciences, 2002. 43(3): p. 345-370.
3. Busemeyer, J.R. and J.T. Townsend, DECISION FIELD-THEORY - A DYNAMIC COGNITIVE APPROACH TO DECISION-MAKING IN AN UNCERTAIN ENVIRONMENT. Psychological Review, 1993. 100(3): p. 432-459.
4. Hotaling, J.M., J.R. Busemeyer, and J. Li, Theoretical Developments in Decision Field Theory: Comment on Tsetsos, Usher, and Chater (2010). Psychological Review, 2010. 117(4): p. 1294-1298.

5. Shocker, A.D., et al., Consideration set influences on consumer decision-making and choice: Issues, models, and suggestions. Marketing letters, 1991. 2(3): p. 181-197.
6. Frey, D.D., et al., The Pugh Controlled Convergence method: model-based evaluation and implications for design theory. Research in Engineering Design, 2009. 20(1): p. 41-58.
7. Singer, D., N. Doerry, and M. Buckley, What is Set-Based Design? Naval Engineers Journal, 2009. 121(4): p. 31-43.
8. Stump, G., et al., Visual Steering Commands for Trade Space Exploration : User-Guided Sampling With Example. Journal of Computing and Information Science in Engineering, 2009. 9(4): p. 044501:1-10.
9. Miller, S., et al., Preference Construction, Sequential Decision Making, and Trade Space Exploration, in ASME 2013 IDETC/CIE 2013. 2013: Portland, OR.
10. Bettman, J.R., M.F. Luce, and J.W. Payne, Preference construction and preference stability: Putting the pillow to rest. Journal of Consumer Psychology, 2008. 18(3): p. 170-174.
11. Dean, M. and A. Caplin, Search and satisficing. The American Economic Review, 2011. 101(7): p. 2899-2922.
12. Kahn, B.E. and J. Baron, An Exploratory Study of Choice Rules Favored for High-Stakes Decisions. Journal of Consumer Psychology, 1995. 4(4): p. 305-328.
13. Abdul-Muhmin, A.G., Contingent Decision Behavior: Effect of Number of Alternatives To Be Selected on Consumers' Decision Processes. Journal of Consumer Psychology, 1999. 8(1): p. 91-111.
14. Kahneman, D. and A. Tversky, The rational choice, values and frames. Psikhologicheskii Zhurnal, 2003. 24(4): p. 31-42.
15. Tversky, A. and D. Kahneman, RATIONAL CHOICE AND THE FRAMING OF DECISIONS. Journal of Business, 1986. 59(4): p. S251-S278.
16. Tversky, A. and D. Kahneman, ADVANCES IN PROSPECT-THEORY - CUMULATIVE REPRESENTATION OF UNCERTAINTY. Journal of Risk and Uncertainty, 1992. 5(4): p. 297-323.
17. Phillips, W.F. and D.O. Snyder, Modern adaptation of Prandtl's classic lifting-line theory. Journal of Aircraft, 2000. 37(4): p. 662-670.

7 AIR FORCE INSTITUTE OF TECHNOLOGY

7.1 PAST RESULTS: PHASES 1 AND 2

Phase 1 focused on the development and application of the CEVLCC model referenced below. Phase 2 focused on the initial work on the application to the Air Force Advanced Trainer (T-X) Concept described below.

7.2 PHASE 3 RESULTS

7.2.1 APPLICATION PROJECT – A METHOD FOR EVALUATING DESIGN FLEXIBILITY EARLY IN DESIGN APPLIED TO AIR FORCE ADVANCED TRAINER (T-X) CONCEPT

This research concluded that a methodology could be developed that quantitatively measures design flexibility and the expected impact to Life Cycle Cost (LCC) based upon era realization. The methodology builds on the stochastic life cycle cost work done by Ryan et al, [Ryan, E., "Cost-Based Decision Model for Valuing System Design Options," PhD Dissertation, Air Force Institute of Technology, Wright-Patterson AFB, OH, September 2012] and applies it to the Air Force Advanced Trainer (T-X) Concept, with subject matter support from the AF Life Cycle Management Center. Variations of an Air-Force-only trainer development program were considered based on potential requirements for Navy, Heavy (multi-engine) and Special Operations trainers. Varying probabilities for the realization of these requirements were considered as well as the cost impact from incorporating "hooks" for these requirements in the baseline AF-only design. These "hooks," such as a stronger and heavier structure to accommodate Navy carrier operations or a multi-engine configuration to support a heavy aircraft requirement, had adverse impacts on both acquisition and sustainment costs, which was not always offset by economies of scale in the acquisition and sustainment phases.

The quantitative measurement was accomplished by using LCC as a proxy metric for design flexibility. Probability of occurrence for the requirements realizations, the cost impact of design flexibility, and cost modifiers associated with economies of scale (for both production and sustainment) were large drivers of expected LCC. Sensitivity analysis was performed to identify how robust the preferred solution was to variations in the event probabilities and/or cost impact parameters. Given the assumptions of the study, this method and application demonstrated that when expected LCC is used to infer value of design flexibility, a quantifiable return on investment can be measured. The results of this research were briefed to AF Materiel Command Headquarters (AFMC/EN) and the Development Planning office of the AF Life Cycle Management Center and were documented in the AFIT MS Thesis (AFIT-ENV-14-M-05) entitled "Exploring a Method to Quantitatively Measure Design Flexibility Early in the Defense

Acquisition Life Cycle" by Captain Joseph Kim. Extensions to this work are looking to create and demonstrate stronger linkages between the system/concept architecture and the design impacts, as well as to provide sufficient input data for higher fidelity cost estimating models.

7.2.2. METHODOLOGY DEVELOPMENT WITH APPLICATION – EVALUATING THE IMPACT OF REQUIREMENTS CHANGES ON DESIGN AND ACQUISITION EFFORT

Military systems have traditionally been designed to satisfy a small set of initial requirements. This approach often causes an inability for the system to easily meet future requirements. This research introduces a modeling method which uses architecture analysis to assess the impact of change on a system module or component due to a specific requirements change. It attempts to inform the designer/decision-maker as to strategies that are more likely to mitigate the impact of potential requirements changes. The method modifies the Generational Variety Index [Martin, M. V., & Ishii, K. (2002). Design for variety: developing standardized and modularized product platform architectures. *Research in Engineering Design, 13*, 213-235] methodology by including uncertainty in (1) the likelihood of a specific requirements change, and (2) the impact of that requirements change on a system module. In addition to requirements changes, per se, we consider two types of design changes that can result from a requirements change—scalable changes, which are incremental in nature, and modifiable changes, which are more radical. Our probabilistic model examines (1) how a specific requirements change impacts a module, (2) how a given module is impacted by a combination of requirements changes, (3) how a specific requirements change impacts the entire system, and (4) how the system is impacted by a combination of requirements changes. Examples of architectural information used by this approach include requirements traceability to system functionality, allocation decisions tying functions to components, and identification of interfaces and/or design dependencies between components or modules. Results obtained from our method can further inform decisions regarding resource allocation and system design in the face of uncertainty regarding future system requirements.

The following investigative questions are considered by this research:

1. What are various system-level architecture options that allow a system architecture/design to meet current and possible future requirements? The system architecture provides traceability from the requirements to system functionality to system modules or components. The terms "component" and "module" are used interchangeably for purposes of this research. A module is defined as an "an independent building block of a larger system with a specific function and well-defined interfaces" [Hölttä-Otto, K. (2005). *Modular Product Platform Design.* Espoo, Finland: Helsinki University of Technology].
2. How does a type of requirements change impact a module/component?
3. How is the module/component impacted by a combination of requirement changes?

The two previous questions provide the necessary information for a decision maker to understand the impacts on modules as a result of requirements changes. This information allows system designers to determine where change mitigation should occur and how it should occur.

4. How does a type of requirements change impact the system as a whole?
5. How is the system impacted by a combination of requirement changes?

These two questions use the same information available to the component impact questions but instead focus the results on how the requirements impact the system rather than components. This information provides the necessary information for a decision maker to prioritize the management of requirements change and then determine strategies on how to reduce the impact of changes due to requirements.

The proposed methodology as described by this paper only considers the initiating point of functional change and the direct/indirect impact of that functional change on linking functions in the system. The research demonstrates the model use through the analysis of a flexible munition system architecture case study.

This effort will continue under Phase IV of RT-113. A Master's Thesis by Captain Jason Altenhofen will be provided in the March 2015 timeframe, and a paper/presentation based on this work has been accepted for the Industrial and Systems Engineering Conference (ISERC) in May 2015. Additional work associated with a flexible weapons cost modeling approach is ongoing as part of this effort, and will be completed under Phase IV.

7.2.3 Theoretical Development – Energy and Information Impacts on System Functionality and Effectiveness

While system functions and functionality are widely used concepts in systems engineering, there is significant diversity in their definitions and no unified approach to measurement. This research establishes a systems engineering method for measuring impacts to functionality in dynamic engineered systems based on changes in kinetic energy. This metric is applied at particular levels of abstraction and system scales, consistent with the established multi-scale nature of systems. By measuring system behavior in context with expected scenarios, it is possible to estimate expected functionality or set bounds on a system's maximum functionality. This measurement can be used to evaluate system robustness and resiliency, compare systems, examine impacts to functionality based on energy or information availability, and guide development of system architecture.

An initial investigation in this new research area explores the relationship between energy and system functionality to assess the robustness of an operational system in the face of variations associated with energy available. Results from this initial investigation have been submitted to the INCOSE Systems Engineering Journal and for presentation at the Industrial and Systems

Engineering Conference (ISERC) in May 2015. Additional work in this area will continue under Phase IV, and will address connections to information content, information availability, and system complexity. The principal author for this work is Jason Clark, an AFIT doctoral student supported in part by RT-113.

7.3 Planned Phase 4 Work - Combined AFIT and NPS RT-113 Task for 2015

AFIT and NPS are developing a common application and integrated effort for validating several of the approaches they have been developing and refining under RT-46 and RT-113. The application will involve heterogeneous teams of autonomous and cooperative agents for ISR missions. A System-of-System level architecture will be developed encompassing a variety of tactical ISR missions that must be conducted in contested and denied environments. Environments that affect the functionality and/or performance of communications, surveillance, and navigation environments present unique challenges for ISR UAV CONOPs with respect to UAV system capabilities (communications, sensing, data processing, command and control, navigation). Actors associated with this architecture will include not only variants of autonomous/cooperative agents, but command and control and adversarial elements as well. The architecture will be developed to support analysis of effectiveness and robustness given uncertain future scenarios and possible design approaches for the autonomous/cooperative agents and their associated command and control system. The architecture will also be developed to support software and system cost estimating approaches in order to assess cost effectiveness as well as mission effectiveness. This effort will build upon several prior efforts at AFIT and NPS:

- AFIT has modeled System-of-System level architecture behavior and effectiveness in both the Arena and MATLAB environments. These efforts have shown the value of architecture-based modeling to aid in requirements definition and design decisions. While these efforts have been successful, they have required a mapping from the architecture definition to the modeling and simulation environment, and they have typically not integrated cost estimation into the analysis. The current proposed effort will utilize new Model Based Systems Engineering tools to eliminate the need for a transition from the architecture to an executable model. Tools such as Magic Draw and/or the Monterey Phoenix approach developed by NPS will be considered for the proposed task.

- NPS has pioneered the Monterey Phoenix (MP) approach to developing executable architectures that can be used to evaluate performance, expose design problems, and uncover problems associated with emergent behavior. The approach can be used to auto-generate a range of scenarios based on the behaviors of the various actors, allowing the architect/system designer to modify the systems to either accommodate favorable scenarios or eliminate the possibility of unfavorable scenarios. The intent with MP is to support a complete cycle of automated system verification/validation that is currently not possible using existing UML/SysML approaches.

- AFIT has been applying MIT's Epoch Era Analysis for capturing uncertain future requirements and environments. This has been combined with a stochastic life cycle cost modeling approach to estimate an expected value of the life cycle cost given probabilities for the future epochs and cost estimates to adapt or modify the systems to accommodate the epochs. An initial application looked at the Air Force advanced trainer concept, and a current application is applying the methodology to an Air Force concept for Flexible Weapons (GBU-X). The current GBU-X effort is identifying the architectural features essential for estimation of both acquisition and operations/support costs of the systems, and seeks to create a business case analysis that compares the legacy munitions approach to a more flexible architecture.

- NPS has been developing approaches for improving integrated systems and software cost models and connecting them directly to tradespace analysis tools. During RT46, they prototyped a web service for the COQUALMO cost/quality model and in this phase are developing SysML constructs for computing COSYSMO and COCOMO II cost/risk (with USC and Georgia Tech). The ISR UAV architectural system requirements can be automatically fed into COSYSMO for systems engineering cost/risk. Software size also needs to be captured. MP can compute function point measures for software size, and this capability will be evaluated. It will also be assessed for deriving systems engineering inputs.

The proposed AFIT-NPS tasks are as follows:

Develop a baseline Operational and System Architecture to capture a set of military scenarios, using new Model-based Systems Engineering (MBSE) tool(s).

For a representative multi-agent UAS system within the environments of interest, this effort will produce a baseline set of SysML views. In particular, a Parametric View will be produced to support the cost and analysis models that will become part of the overall effort.

Transition the baseline architecture to the Monterey Phoenix environment. The MP process will be used to identify alternate events for each actor in each scenario. With this "superset" of information, we can automatically generate all possible use case variants within the scope to provide a scenario coverage that far exceeds what could be done with only a manual effort. This provides a more substantial reference data set the system architecture must satisfy (containing many more possible behaviors, including off nominal cases) and a more complete set of input data to ensuing cost and performance analyses.

Utilize the executable architecture modeling framework of MP to perform automated assertion checking and find counterexamples of behavior that violate the expected

system's correctness. This supports a complete cycle of automated system verification/validation impossible with existing SysML/UML technologies. Exhaustive/representative sets of scenarios generated by MP will be transformed by automated tools into implementation testing suites, another huge benefit providing the continuity of design from the early architecture models to the implementation.

Design and Demonstrate ISR UAV tradespace. The power of new tools such as MagicDraw and ModelCenter is the integration with analysis tools, across a network or on the same machine. SysML has traditionally only captured and documented the operational concept, system requirements, activities/tasks, organizations and information flows, including possible physical instantiations. New MBSE tools facilitate analysis such as optimization, simulation, design of experiments, assessment, sensitivity analysis and statistical hypothesis testing and regression. For this project, such trades and characterizations could examine collaborative and vehicle swarming algorithms, increasing autonomy on multi-vehicle, and single operator operations. Likewise, the effects on environmental variables within the architecture, such as communications and/or GPS jamming, air defenses, evasion, camouflage, and other factors could define the scenarios. These types of trades demonstrate how a business case for varying technologies, capabilities or designs could be accomplished, if cost information is included.

Develop life cycle cost models for the various components of the architecture, and embed them within a larger stochastic life cost estimating approach to evaluate cost effectiveness in an uncertain future environment. Cost data will be attached to every actor and event in every possible scenario, computing the cost of each scenario, were it to occur. A probability of occurrence for each possible scenario will be generated, providing for highly refined overall cost estimates (for operations, for maintenance, and perhaps earlier lifecycle phases) within a specified confidence interval.

8 NAVAL POSTGRADUATE SCHOOL

8.1 PAST RESULTS: PHASES 1 AND 2

The NPS Phase 2 activities improved and piloted several existing ITA analysis toolsets based on the results of Phase 1. The focus for iTAP MPT extensions and applications was in the Ships and Aircraft domains, and making provisions for Space Systems in Phase 3.

We met the following goals for research as detailed in the Phase 2 report 2:

- Experimented with tailoring existing or new tradespace and affordability MPTs for use by an early adopter organization
- Trained early adopters in its use, monitor their pilot usage, and determined areas of strengths and needed improvements, especially in the MPTs' ilities
- Extended the MPTs to address the top-priority needed improvements
- Worked with early adopters to help transition the improved MPTs into their use
- Identified and pursued further improvements for the early adopters or for more general usage.

The tools were tailored for software product line cost modeling, and total ownership cost for integrated engineering activities. The early adopters represented NAVAIR and NAVSEA. An array of improvements for our models and tools were identified for going forward in Phase 3 for ility tradeoffs.

We supported outreach meetings to summarize and demonstrate iTAP capabilities to potential early-adopter organizations. These included visits to the Army Engineer Research and Development Center (ERDC) in Vicksburg, MS, and NAVSEA CREATE-Ships personnel in Carderock associated with DoD Engineered Resilient Systems (ERS).

We also engaged in new community-building activities with NAVAIR stakeholders. NPS and USC began collaboration with the NAVAIR avionics software product line FACE program. We are supporting their surveys with recommendations, data collection, interpreting software lifecycle cost models and calibrations of the COPLIMO product line cost model 8.

This MPT transitioning is an outgrowth from RT-46 Phase 1 and RT-18 product line cost modeling. This application is a highly relevant example of modeling product line benefits for the DoD. See the *Appendix - Product Line Modeling Background* for more detail as this is core model is applied and extended in later phases.

A previous shortfall of our TOC toolset was lacking the capability to estimate operations and maintenance. We added parametric maintenance models into our system cost model suite for

systems engineering, software engineering, hardware development and production. The initial maintenance models are for systems and software.

Cost uncertainty modeling was also extended via improvements in Monte Carlo analysis. Additional size inputs were made available for probabilistic distributions, as well as a wider array of distribution types. This feature works in tandem with the new lifecycle extensions for maintenance.

We began a ship case study for design and cost tradeoffs with military students at NAVSEA. The group is designing a new carrier and integrating RT-46 cost models into a Model-Based Systems Engineering (MBSE) dashboard for Total Ship Systems Engineering (TSSE). Part of the applied research is a comparison and refinement of potential ship cost models for affordability tradeoffs in the MBSE framework.

Initial comparisons of MIL-STD 881 Work Breakdown Structures (WBS) were performed to find commonalities and variabilities across DoD domains, and identify suggested improvements. This analysis informs us how to best structure canonical TOC tools to address multiple DoD domains efficiently. Additionally, a detailed review and critique of the recent MIL-STD 881 UAV WBS was done and deficiencies noted for *autonomy* trends which are of increasing importance.

NPS further extended Phase 1 cost models for breadth of engineering disciplines to include systems engineering, software engineering and hardware. We also added Monte Carlo risk analysis for a subset of cost parameters in the integrated SE/SW/HW cost model.

To better the address full lifecycle costs we improved TOC capabilities by adding lifecycle maintenance models. We started on extensions of general cost models for DoD system types starting with ships and space systems.

NPS supported startup efforts with USC, SMC and the Aerospace Corp. for researching and incrementally developing the next-generation full-coverage space systems cost estimation model COSATMO. A prototype mockup for an existing satellite cost model was developed to support tool usage scenario discussions.

The recently updated MIL-STD 881C 13 standard WBS for UAVs was critiqued by NPS and assessed for cost modeling. Recommendations were identified to better address *autonomy* trends for the DoD, as these are increasingly important and crossing into the Ship domain for mixed Navy systems.

See the Phase 2 report for underlying details of the WBS critique 2. A UAV oriented pilot application is planned for Phase 4. This critique will help inform the cost modeling aspect for UAV systems.

8.2 - PHASE 3 RESULTS

On Task 2 we reached out to DoD organizations and sought MPT piloting opportunities. Task 3 activities added incremental capability enhancements to our suite of system cost models and tools. Phase 3 included the successful adoption of the Constructive Product Line Model (COPLIMO) at NAVAIR resulting in follow-on research funding to extend the software cost model. This example application of a model and tool is an outgrowth of both Task 2 for piloting, and Task 3 to develop an ensemble of DoD-relevant cost models.

8.2.1 TASK 2: MPTS AND PILOTING

This task is a continuation of RT46 Method, Processes and Tools (MPT) development and pilot applications. It continued some Phase 2 work seeking to further apply concurrent cyber-physical-human system ilities tradespace analysis and design to NAVSEA and Army TACOM systems. We supported the NAVSEA collaboration activities, helping prepare for the NAVSEA Technical Interchange meeting on cost modeling capabilities but couldn't attend.

After investigations to collaborate with the CREATE-SHIPS program in Phase 2, we sought to pilot MPTs in the Navy Ship domain for affordability tradeoffs with NAVSEA ship design. This continued in Phase 3 along with further targets of opportunity at NAVSEA. We also continued integration and community-building activities with Engineering Resilient Systems (ERS) and other DoD programs.

We also continued collaboration with the NAVAIR avionics software product line cost modeling, defining goals of the overall FACE business case analysis. For this we studied NAVAIR's air platform scenarios and COPLIMO variation for model validation purposes. A multi-module and multi-mission extension to support the avionics software product line cost analysis was defined. See the *Appendix - Product Line Modeling Background* for further details of the model foundation.

We helped validate the preliminary parametric approach adopted by NAVAIR and their assumptions. We identified and discussed survey improvements for product line organizational maturity assessment for the FACE consortium for NAVAIR.

For MPTs we also explored application of SysML architecture models to support costing in ship and space system domains, and canvas for existing SysML models for deconstruction. Monterey Phoenix (MP) 56 was identified as an alternative modeling approach for tradespace analysis.

We continued assessment of the MP architectural modeling approach in conjunction with SySML. We also evaluated the feasibility of using web service-based cost models in the MP environment. Potential domain applications were evaluated and we chose a UAV swarm pilot

supporting an ISR mission. We defined a corresponding tradespace demonstration project using MBSE tools with AFIT for Phase 4.

Supporting details of the NAVSEA and NAVAIR pilot applications are next.

NAVSEA Ship Design Pilot Application

We continued NAVSEA collaboration activities by demonstrating the NPS Dashboard for Total Ship Systems Engineering (TSSE), in support of NAVSEA and the Army Materiel Systems Analysis Activity (AMSAA) application to watercraft cost modeling and tradespace analysis.

NPS has a TSSE MBSE Dashboard platform suitable for MPT piloting, extending and transitioning research. We can extend the TSSE MBSE infrastructure for affordability trades to complement and interoperate physical tradespace with TOC models. This is feasible since the MBSE models encapsulate ship or air vehicle requirements and factors linkable to parametric cost models for two-way integration.

The TSSE dashboard implements a methodology for effectiveness-based engineering design including:

- Integration of systems architecture, combat systems, and combat operations, as well as related life cycle design and cost considerations
- Impact of system trade offs
- Impact of decision options for ship design, cost, and effectiveness in multiple criteria trade space analysis.

The system design output is based on needed combat capabilities (mission effectiveness), engineering feasibility (ship synthesis), and cost. The dashboard focus is early in life cycle, and to provide assistance to a decision maker. It is conceptually overviewed in Figure 30.

Currently two Excel-based ship cost models exist for 1) a very simple cost model within an existing ship synthesis model, and 2) an NPS ship cost model derived from more detailed outside sources. The team can study and implement aspects of each and the web-based cost modeling tool.

The MBSE dashboard project was web demonstrated by NPS researchers in Phase 3 to potential adopters at AMSAA for a new Maneuver Support Vessel. The dashboard was well received, but it wasn't good adoption timing due to imminent watercraft program milestones.

There was extensive NAVSEA and ERS coordination effort with WSU, also enlisting support from Cliff Whitcomb at NPS as we sought the pilot application with AMSAA. Discussions with Jeff Hough at NAVSEA indicated they already have a large staff supporting analysis of the new watercraft with no immediate need for assistance. There is currently no viable opportunity for piloting but we will continue discussions to reach better understandings. These activities are co-reported with supporting details in the WSU section.

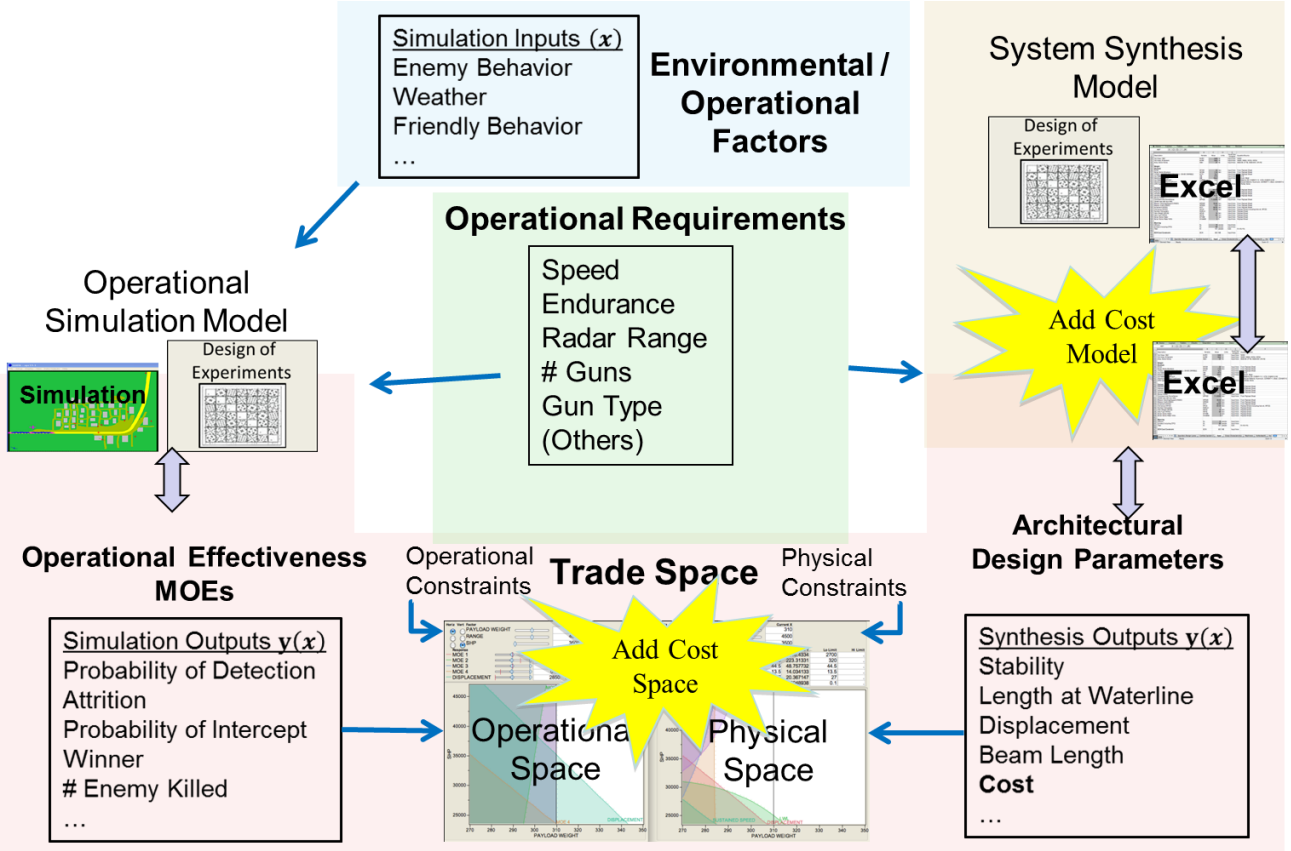


Figure 30: MBSE Design Process

NAVAIR Software Product Line Modeling Pilot Application

NPS and USC have been collaborating with NAVAIR stakeholders involved in avionics software product line architectures on the Future Airborne Capability Environment (FACE™) initiative. The FACE approach, via common standards, standardization of software interfaces and software re-use, offers a number of benefits such as increased competition, reduced software development times, greater innovation, and lower cost of doing business 12.

FACE is a technical standard that defines a common operating environment supporting portability and reuse of software components across DoD aviation systems. The FACE Ecosystem is intended to provide the following:

- An open technical standard that defines/specifies a reference architecture which is in alignment with DoD Open Architecture guidance (modular, open, partitionable)

- Thoroughly defined, standardized, verifiable, open APIs at key interfaces
- A process for conformance verification and certification
- A registry of certified FACE conformant software.

FACE describes the standard framework upon which capabilities can be developed as Software Product Lines (SPLs) to enhance portability, speed to field, reuse, and tech refresh, while reducing duplicative development. The FACE initiative ties SPLs, architectures and business principals together into a coherent process for use across DoD. The FACE standard also describes a Reference Architecture that supports several technical "ilities" to include flexibility, scalability, reusability, portability, extensibility, conformance testability, modifiability, usability, interoperability, and integrateability.

By using the FACE technical standard, decoupling the software from its interfaces, and adding the required layers of abstraction, the software can be reused across multiple platforms for very little cost beyond the initial development costs for both new development and life cycle updates.

Benefits to the government from a SPL approach include:

- Reduced development and life cycle costs
- Reduced time to field
- Reduce vendor lock
- Reduced redundant development
- Increased competition and competitive avionics software marketplace
- Increased opportunities for reuse
- Testable OSA requirements

Industry benefits from a SPL include:

- Companies can avoid “locking in loss” in a time of decreasing budgets
- Opens previously closed markets to all vendors
- Innovative companies can preserve market share due to reduced vendor lock
- Allows small businesses more opportunity to provide capabilities
- Allows air frame vendors to focus on what they do best
- Facilitates interoperability between industry partners in support of teaming arrangements

Delphi Survey

The goal of gathering industry responses is to determine current software development costs, development processes and reuse practices in the defense avionics software industry. This can be used to forecast potential cost savings and process improvements brought about by the FACE common operating environment. The Delphi surveys are to obtain a consensus view identifying:

- The current software effort drivers in this sector
- Their level of influence on software development effort
- The impact of FACE on software effort drivers.

This information be used to calibrate Government software cost estimation models by

- Adding or changing effort drivers for FACE

- Calibrating the influence of particular effort drivers for estimates of programs using FACE.

The final results will be used to develop and refine a Business Case Analysis (BCA) that will estimate cost avoidance over the lifecycle of a FACE conformant platform. An earlier, preliminary Delphi showed the representative impacts of the FACE product line approach in **Figure 31**. Note this CER applies only to software engineering effort and is an adjustment factor applied to estimates of effort to develop “new” or "modified" interface software code. The FACE CER is not to be applied to reused code (business logic), hardware, testing or other types of costs.

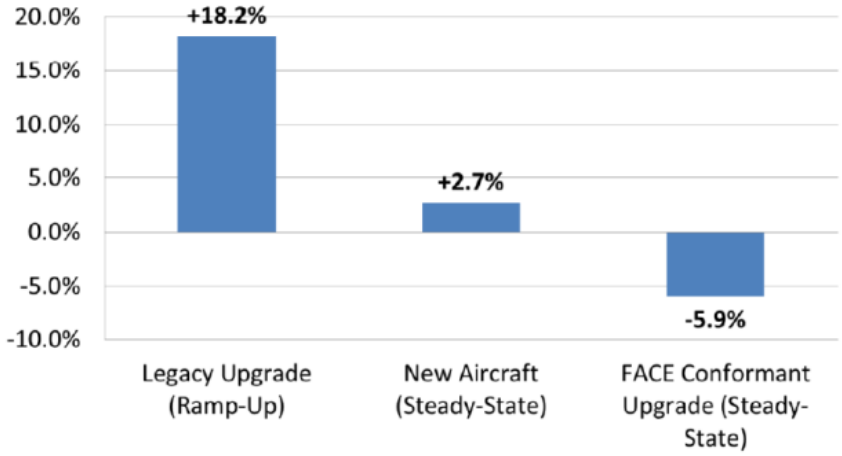


Figure 31: Representative Impact of FACE Architecture on Effort

We are supporting the fuller Delphi effort to better define the cost parameters and usage scenarios using the more detailed COPLIMO baseline parameters.

At the end of Phase 3 the next FACE industry survey was still under refinement. It is currently composed of the following sections:

- Section 1 - Company (Division) & Software Engineering Unit Background
- Section 2 - Software Engineering Unit Operations (Practices)
- Sections 3 - Software Effort Drivers prior to the FACE Initiative
- Section 4 - Introduction to the FACE Initiative
- Sections 5 - Software Effort Drivers in the FACE Environment
- Section 6 - Study Close & Opportunity for Feedback

Parametric Modeling

By interpeting COPLIMO for this unique environment, this collaboration has also identified the following extensions to better model the avionics software product line approach:

- Treat only a portion of the overall software system as product-line software. The original COPLIMO assumes sizes are 100% inherited from product commonality.
- Account for additional equivalent size for the integration layer requirements and associated effort, which must be included with system-specific requirements/effort.

These new model aspects were pursued in Phase 3 along with supporting a revised Delphi survey. The specific NAVAIR extensions and instantiations of COPLIMO are summarized below:

- Five aircraft platforms need updates to their current Flight Management System (FMS):
    1. Jet
    2. Cargo Plane
    3. Multi-mission Helicopter
    4. Support Helicopter
    5. New Jet Fighter (does not yet exist, i.e. a Gen X Fighter)

- For each platform updates are needed on one or more of the following possible FMS capabilities:
    1. Guidance
    2. Hover
    3. Landing System
    4. Auto Rerouter
    5. Military Flight Management (MIL FM)
    6. Civilian Flight Management (CIV FM)
    7. Navigation Database (NAV DB)

- Task Order (Used for Process Maturity/ Learning)
    1. Updates performed one platform at a time in the order listed above.
    2. Applicable FMS capabilities for each platform will be updated one at a time in the order the capabilities are listed above.

Considerations of the scenarios were used to derive the model inputs in Table 10. Not shown are the software size values of the components. From these inputs a portion of the outputs are shown in Table 11. In this platform case the saving are substantial.

Table 10. FACE Product Line Model Scenario Inputs

Parameter	**Scenario 1**	**Scenario 2**	**Scenario 3**	**Scenario 4**
REUSE	High	Extra High	Low	Nominal
DOCU	High	Low	High	Low
RELY	High	Nominal	Nominal	Nominal
AA	Basic	Basic	Basic	Basic
SU	Very High	Very High	Very High	Very High
UNFM	Completely Familiar	Completely Familiar	Completely Familiar	Completely Familiar
DM	20%	20%	20%	20%
CM	20%	20%	20%	20%

IM	50%	50%	50%	50%
PFRAC	40%	40%	40%	40%
RFRAC	30%	30%	30%	30%
AFRAC	30%	30%	30%	30%
FACE Dev. State	Legacy Upgrade	Legacy Upgrade	Legacy Upgrade	Legacy Upgrade
Labor Rate	$150	$150	$150	$150
Hours Per Month	160%	160%	160%	160%
Level of Integration Relative to Effort Size	10%	10%	10%	10%
LCU 1	20%	20%	20%	20%
LCU 2	65%	65%	65%	65%
LCU 3	30%	30%	30%	30%
LCU 4	15%	15%	15%	15%
LCU 5	25%	25%	25%	25%
FACE LCU State	FACE Upgrade	FACE Upgrade	FACE Upgrade	FACE Upgrade

Table 11. FACE Produce Line Model Ample Output (Portion)

	LCU 1 PMRs	**LCU 1 Cost**	**LCU 2 PMRs**	**LCU 2 Cost**
Non_PL Dev	886.22	$21,269,268	2880.21	$69,125,120
Integration	133.50	$3,203,988	433.87	$10,412,963
Non-FACE Total	1019.72	$24,473,256	3314.09	$79,538,083
FACE/PL Dev	219.59	$5,270,086	688.67	$16,528,175
FACE/Integration	125.62	$3,014,953	408.27	$9,798,598
FACE Total	345.21	$8,285,039	1096.95	$26,326,773
	LCU 3 PMRs	**LCU 3 Cost**	**LCU 4 PMRs**	**LCU 4 Cost**
Non_PL Dev	1329.33	$31,903,902	664.66	$15,951,951
Integration	200.25	$4,805,983	100.12	$2,402,991
Non-FACE Total	1529.58	$36,709,884	764.79	$18,354,942
FACE/PL Dev	323.83	$7,771,884	167.47	$4,019,187
FACE/Integration	188.43	$4,522,430	94.22	$2,261,215
FACE Total	512.26	$12,294,313	261.68	$6,280,402
	LCU 5 PMRs	**LCU 5 Cost**		
Non_PL Dev	1107.77	$26,586,585	**Total Non FACE**	
Integration	166.87	$4,004,986	$189,667,735	
Non-FACE Total	1274.65	$30,591,570	**Total FACE**	
FACE/PL Dev	271.71	$6,520,985	$63,476,204	
FACE/Integration	157.03	$3,768,691	**SPL Savings**	$126,191,531

FACE Total	428.74	$10,289,676		

The parametric cost modeling and survey enhancement activities continue. The above are representative outputs of one analysis iteration. Assumptions are still being evaluated and parameter values refined. The ongoing study results will be superseded.

8.2.2 TASK 3. NEXT-GENERATION, FULL-COVERAGE COST ESTIMATION MODEL ENSEMBLES

Beginning with work in the space domain with USAF/SMC and the Aerospace Corp., this task is to research and develop an ensemble of cost estimation models covering the systems engineering, development, production, operations, sustainment, and retirement. NPS continued extending the scope and tradespace interoperability of cost models and tools from previous phases.

This also comprises the related Task 2 work supporting NAVAIR avionics software product line cost analysis (see Task 2 above). We also elaborated a ship cost model and developed a representative TOC analysis. This is shown later. We also made minor improvements in *System Cost Model Suite* for additional air vehicle types.
We continued exploring cost model extensions to meet initial COSATMO needs, and as a basis for additional domain cost models. The research team also reached out to BAE Systems for empirical systems engineering cost data.

We supported the development and data analysis for an ensemble of cost estimation models/tools beyond COSATMO. This involves defining COSATMO-oriented next-generation versions of software (COCOMO III), systems engineering (COSYSMO 3.0), and extensions of COSYSMO to estimate system development costs.
We started periodic meeting for COCOMO III. We drafted some model requirements, use cases, data definitions and cost driver rating scales. NPS began defining the formal COCOMO III use cases for the research team.
Workshops, presentations, meetings and other events supporting this task included:

- Participated in the January INCOSE Affordability Working Group meeting.

- Participated and presented *Issues in Total Ownership Cost Modeling for DoD Systems* at the Ground Systems Architecture Workshop working group meeting on future ground systems cost estimation needs and models.

- Presented *NPS Total Ownership Cost Models* at SERC Technical Review in March.

- Participated in COSATMO, COSYSMO 3.0 and COCOMO III cost modeling workshops at the USC-CSSE Annual Research Review (ARR) in April.

- Participated in a workshop with the Carnegie Mellon University System Assurance project in Pittsburgh in July. Presented on *Total Ownership Cost and Product Line Models*. Established several potential new collaboration thrusts with both CMU and RT113 researchers at the workshop.

- Prepared for and held USC-NPS-industry workshop at October 21-23 USC COCOMO/Systems and Software Cost Modeling Forum to define COSATMO-oriented next-generation versions of software (COCOMO III), systems engineering (COSYSMO 3.0), and extensions of COSYSMO to estimate system development costs.

The following example illustrates a relevant application of the *System Cost Model Suite* for a ship Total Ownership Cost (TOC) probabilistic estimate using Monte Carlo analysis. The previous Phase 2 report 2 demonstrated a point estimate with the integrated tool. This update shows a representative TOC analysis with further supporting details and the enhanced Monte Carlo capabilities.

Ship Total Ownership Cost Example

In this scenario the Navy requires a new Amphibious Assault Ship with a planned IOC of 2019. The program office wants to assess the Total Ownership Cost for a 15 year operational lifetime and set a conservative budget for it. This example demonstrates uncertainty analysis to determine a high confidence level budget.

The ship will be new design with a most likely weight of 5000 long tons. The combined subsystems for the ship total to the systems engineering sizes and degrees of difficulty in Table 12.

Table 12. Systems Engineering Size

	Easy	Nominal	Difficult
# of System Requirements	120	185	48
# of System Interfaces	12	67	45
# of Algorithms	19	125	58
# of Operational Scenarios	3	14	8

Other system-level cost factors are:

- The requirements understanding is strong with only a few undefined areas.
- There is a similar strong understanding of the system architecture and COTS.
- The level of service requirements involve complex, coupled Key Performance Parameters and are critical due to a risk to human life.

The software will be comprised of the following:

- The total new software to develop is estimated to be 850 KSLOC.
- There will be reused combat system software that is 225 KSLOC. This reused software will require substantial integration and testing that is 50% of the effort compared to new software.
- Other subsystems will include 400 KSLOC of modified software. It is estimated that 10% of the modified software design needs to change, about 15% of its code will change and will require 60% integration and testing compared to new. The software baseline is described as:
    - Moderate level of code commentary, headers, documentation.
    - High cohesion, low coupling.
    - Clear match between program and application world-views.

Additional software cost factors include:

- There will be some development flexibility allowed to the contractors, and occasional relaxation of the need to conform to specified requirements.
- The overall team of organizations is fairly cohesive and largely cooperative with each other.
- The required software reliability of nearly all systems must be very high, since failure could be a risk to human life.
- The software complexity is very high with many subsystems interfacing with each other.
- The software development starts in FY14 and must be completed in five years to meet the IOC of 2019.

For RDT&E we include the costs of systems engineering, software development, hardware development and production. For maintenance and upgrades over 15 years of operations we assume:

- Systems engineering modifications to requirements, interfaces, algorithms, and scenarios are estimated to be 10% per year.
- For software maintenance 150 Equivalent KSLOC is expected to change per year. The software understanding will be improved from RDT&E due to the new software, with an estimated 10% less penalty.
- The likely operational costs of the ship hardware, associated maintenance and spare parts are $30M annually for this ship class.

Further ground rules and assumptions for the lifecycle costs and phasing are:

- All systems, software and hardware activities commence in parallel.
- Hardware monthly costs are constant before IOC, and constant within each year after IOC.
- The software Transition phase occurs once after IOC.

All labor rates are $10,000 per Person-Month.

For systems engineering, the total size is 2650 Equivalent Nominal Requirements. The off-nominal cost factors are described as:

- *Requirements Understanding* is High
- *Architecture Understanding* is High
- *Level of Service Requirements* is Very High.

The software size requires inputs for new, modified and reused categories. The total aggregate size is 1031710 Equivalent SLOC (the detailed inputs are shown later in this example). The off-nominal cost factors are described as:

- *Development Flexibility* is Low
- *Team Cohesion* is High
- *Required Software Reliability* is Very High
- *Product Complexity* is Very High
- *Required Development Schedule* is Very Low.

The ship weight of 5000 long tons at 2240 pounds/long ton is the equivalent of 11,200,000 pounds for the AMCM model input. The new design indicates Block #1.

We will use a symmetrical triangular probability distribution for the ship weight. The ship architects say it might be reduced to 2500 long tons, 5000 is still most likely, but it might need to be 7500 long tons to fit all mission systems on-board.

System requirements volatility is modeled with a uniform probability distribution for systems engineering size. The point estimate for total equivalent size will be the minimum of a uniform distribution with a maximum that is 20% additional size for possible growth.

For software size, we choose a normal probability distribution using the point estimate equivalent size as the mean with a standard deviation of 15%. For simplicity we assume that all other parametric cost factors stay fixed.

The complete systems engineering inputs are shown in **Figure 32**2. The software inputs are in **Figure 33**. Hardware development and production input factors are shown in **Figure 34**.

Constructive Systems Engineering Cost Model (COSYSMO)

System Size

Equivalent Nominal Requirements Distribution: Uniform | Min: 2650 | Max: 3180

System Cost Drivers

Requirements Understanding	High	Documentation	Nominal	Personnel Experience/Continuity	Nominal
Architecture Understanding	High	# and Diversity of Installations/Platforms	Nominal	Process Capability	Nominal
Level of Service Requirements	Very High	# of Recursive Levels in the Design	Nominal	Multisite Coordination	Nominal
Migration Complexity	Nominal	Stakeholder Team Cohesion	Nominal	Tool Support	Nominal
Technology Risk	Nominal	Personnel/Team Capability	Nominal		

Maintenance On | Annual Change %: 10 | Maintenance Duration (Years): 15

System Labor Rates

Cost per Person-Month (Dollars): 10000

Calculate

Results

Systems Engineering

Effort =1294.9 Person-months
Schedule = 16.0 Months
Cost = $12948595

Total Size =2915 Equivalent Nominal Requirements

Acquisition Effort Distribution (Person-Months)

Phase / Activity	Conceptualize	Develop	Operational Test and Evaluation	Transition to Operation
Acquisition and Supply	25.4	46.2	11.8	7.3
Technical Management	48.4	83.6	55.0	33.0
System Design	132.1	155.4	66.0	35.0
Product Realization	25.2	58.3	62.2	48.6
Product Evaluation	72.3	108.4	160.6	60.2

Maintenance

Annual Maintenance Effort = 112.8 Person-Months
Annual Maintenance Cost = $1127775
Total Maintenance Cost = $16916632

Figure 32: Systems Engineering Inputs

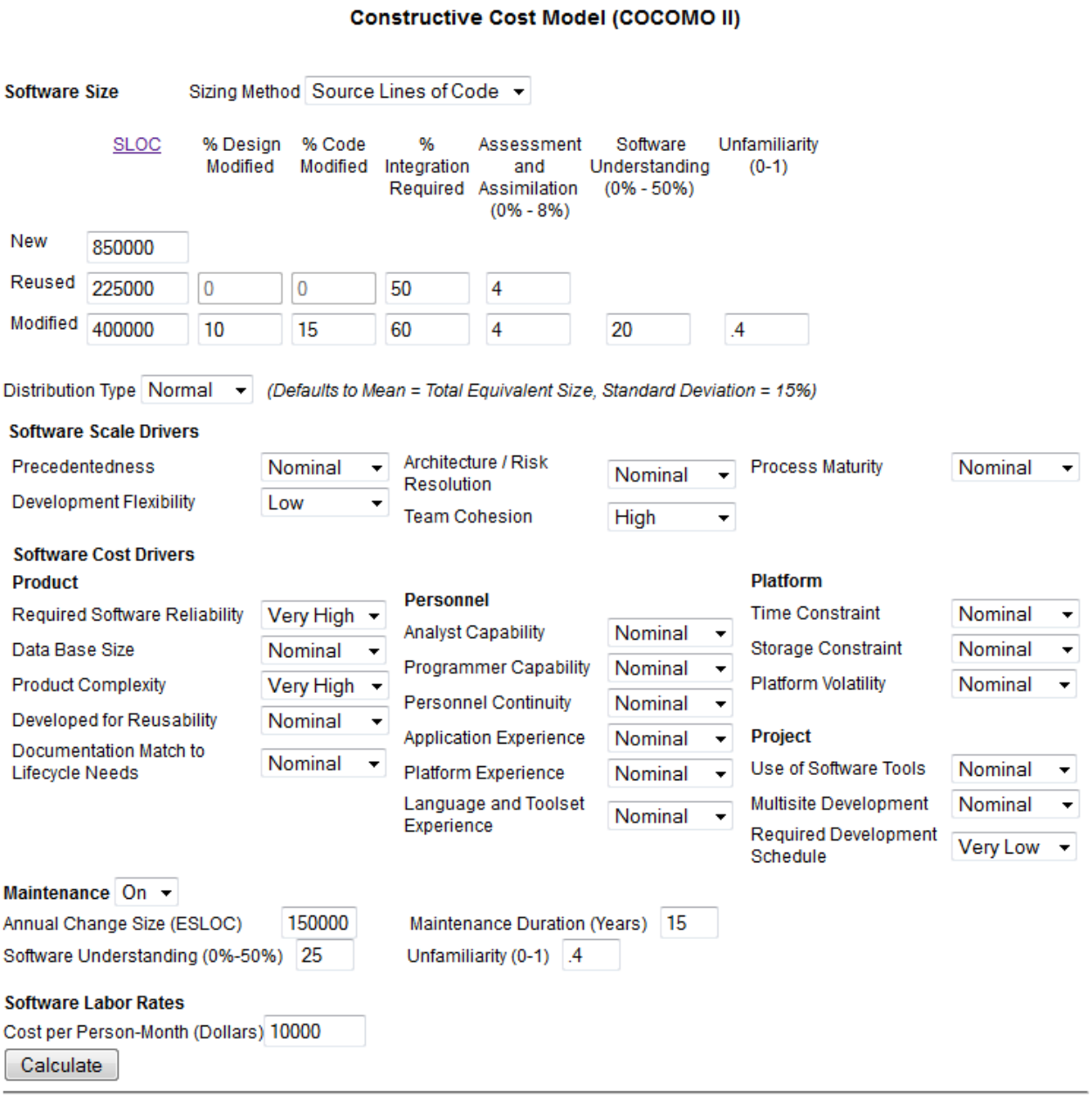


Figure 33: Software Engineering Inputs

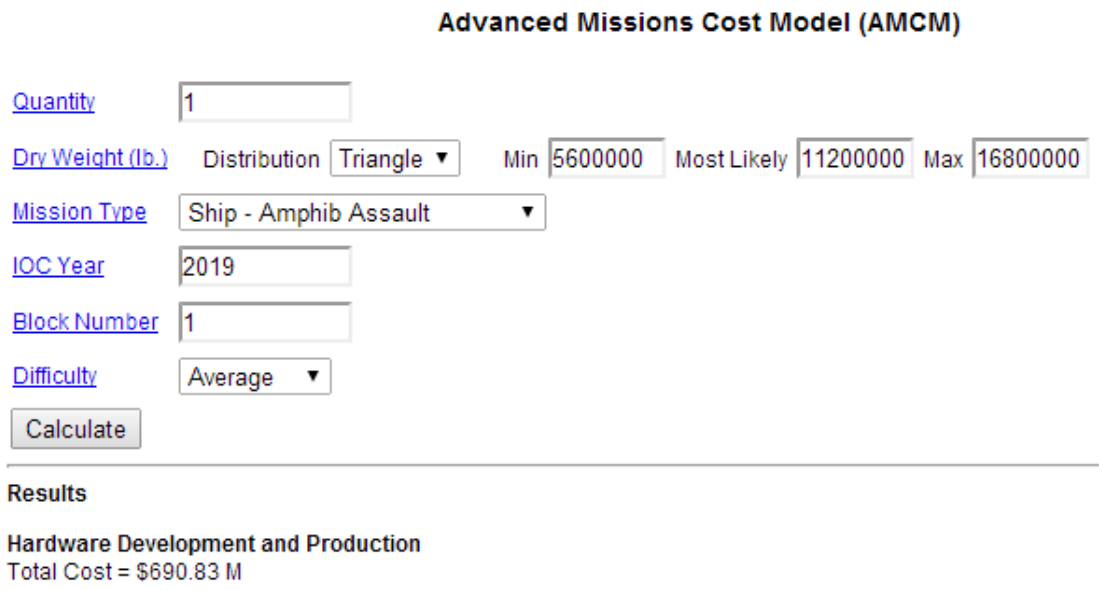


Figure 34: Hardware Inputs

The results in **Figure 35** and **Figure 36** show the summaries and separate CDFs for RDT&E, maintenance and the combined TOC.

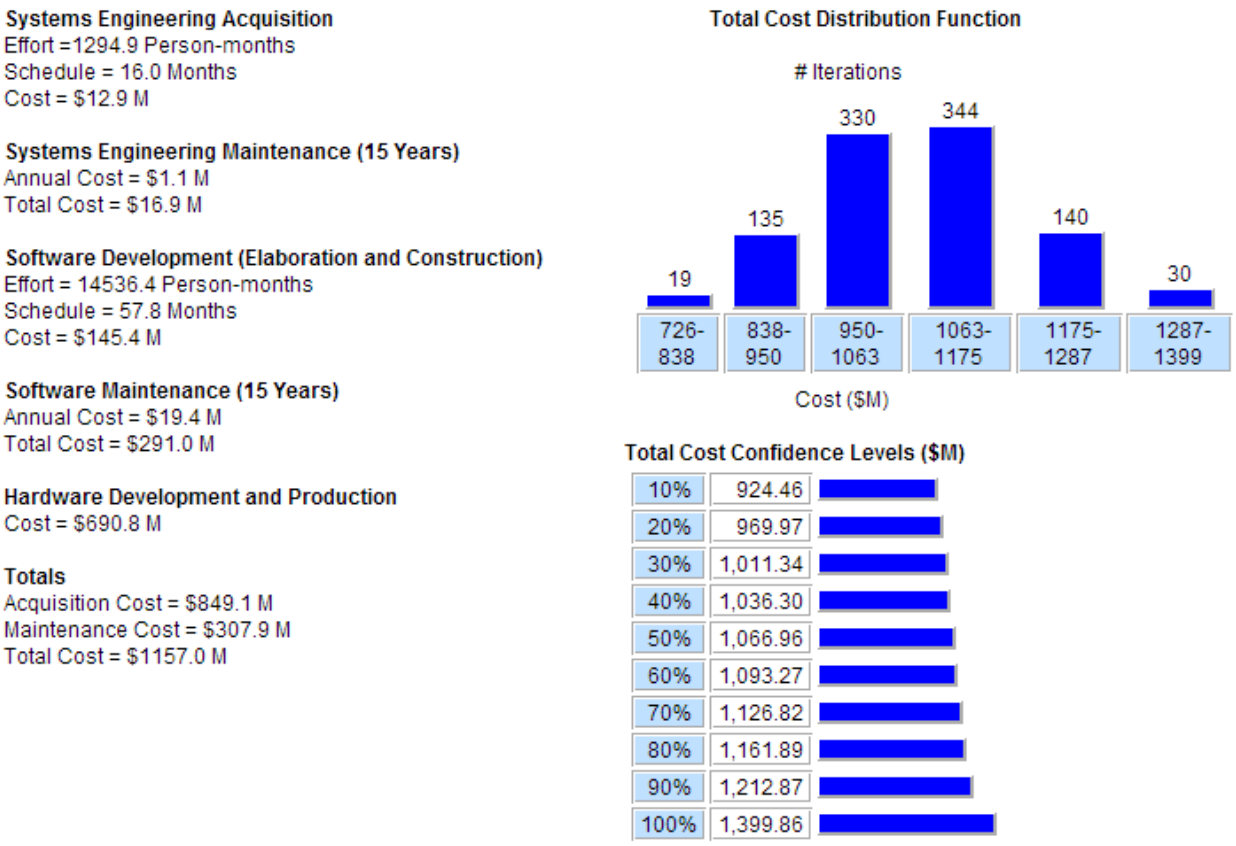


Figure 35: Project Summary with Monte Carlo Analysis

Acquisition Monte Carlo Results

Systems Engineering Cost Confidence Levels ($M)

10%	11.96
20%	12.22
30%	12.47
40%	12.74
50%	12.96
60%	13.18
70%	13.39
80%	13.67
90%	13.92
100%	14.20

Software Engineering Cost Confidence Levels ($M)

10%	91.94
20%	100.82
30%	107.20
40%	112.34
50%	116.20
60%	122.66
70%	126.56
80%	133.07
90%	142.24
100%	184.78

Hardware Cost Confidence Levels ($M)

10%	559.10
20%	604.92
30%	638.87
40%	664.17
50%	686.26
60%	712.00
70%	732.54
80%	768.08
90%	815.12
100%	900.59

Maintenance Monte Carlo Results

Systems Engineering Maintenance Cost Confidence Levels ($M)

10%	15.62
20%	15.96
30%	16.29
40%	16.64
50%	16.93
60%	17.22
70%	17.49
80%	17.86
90%	18.19
100%	18.55

Software Engineering Maintenance Cost Confidence Levels ($M)

10%	184.03
20%	201.80
30%	214.58
40%	224.85
50%	232.58
60%	245.52
70%	253.31
80%	266.35
90%	284.70
100%	369.85

Figure 36: Detailed Monte Carlo Results

8.3 FUTURE PLANS: PHASES 4 AND 5

8.3.1 TASK 2: MPTS AND PILOTING

Task 2 will be a collaboration with AFIT for a joint Intelligence, Surveillance and Reconnaissance (ISR) mission application involving heterogenous teams of autonomous and cooperative agents. NPS will provide cost modeling expertise, tools and Monterey Phoenix (MP) modeling support. A focus will be on translations between models/tools in MBSE, specifically mapping architectural elements into cost model inputs.

AFIT and NPS will develop a common application and integrated effort for validating several of the approaches they have been developing and refining under RT-46 and RT-113. A System-of-System level architecture will be developed encompassing a variety of tactical ISR missions that must be conducted in contested and denied environments. Environments that affect the functionality and/or performance of communications, surveillance, and navigation environments present unique challenges for ISR UAV CONOPs with respect to UAV system capabilities (communications, sensing, data processing, command and control, navigation).

Actors associated with this architecture will include not only variants of autonomous/cooperative agents, but command and control and adversarial elements as well. The architecture will be developed to support analysis of effectiveness and robustness given uncertain future scenarios and possible design approaches for the autonomous/cooperative agents and their associated command and control system. The architecture will also be developed to support software and system cost estimating approaches in order to assess cost effectiveness as well as mission effectiveness.

This effort will build upon several prior efforts at AFIT and NPS:

- AFIT has modeled System-of-System level architecture behavior and effectiveness in both the Arena and MATLAB environments. These efforts have shown the value of architecture based modeling to aid in requirements definition and design decisions. While these efforts have been successful, they have required a mapping from the architecture definition to the modeling and simulation environment, and they have typically not integrated cost estimation into the analysis. The current proposed effort will utilize new Model Based Systems Engineering tools to eliminate the need for a transition from the architecture to an executable model. Tools such as Magic Draw and/or the Monterey Phoenix approach developed by NPS will be considered for the proposed task.

- NPS has pioneered the Monterey Phoenix (MP) approach to developing executable architectures that can be used to evaluate performance, expose design problems, and uncover problems associated with emergent behavior. The approach can be used to auto-generate a range of scenarios based on the behaviors of the various actors, allowing the

architect/system designer to modify the systems to either accommodate favorable scenarios or eliminate the possibility of unfavorable scenarios. The intent with MP is to support a complete cycle of automated system verification/validation that is currently not possible using existing UML/SysML approaches.

- AFIT has been applying MIT's Epoch Era Analysis for capturing uncertain future requirements and environments. This has been combined with a stochastic life cycle cost modeling approach to estimate an expected value of the life cycle cost given probabilities for the future epochs and cost estimates to adapt or modify the systems to accommodate the epochs. An initial application looked at the Air Force advanced trainer concept, and a current application is applying the methodology to an Air Force concept for Flexible Weapons (GBU-X). The current GBU-X effort is identifying the architectural features essential for estimation of both acquisition and operations/support costs of the systems, and seeks to create a business case analysis that compares the legacy munitions approach to a more flexible architecture.

- NPS has been developing approaches for improving integrated systems and software cost models and connecting them directly to tradespace analysis tools. Earlier on RT46 they prototyped a web service for the COQUALMO cost/quality model and in this phase are developing SySML constructs for computing COSYSMO and COCOMO II cost/risk (with USC and Georgia Tech). The ISR UAV architectural system requirements can be automatically fed into COSYSMO for systems engineering cost/risk. Software size also needs to be captured. MP can compute function point measures for software size, and this capability will be evaluated. It will also be assessed for deriving systems engineering inputs.

The planned tasks are:

- **Develop a baseline Operational and System Architecture to capture a set of military scenarios, using new Model-based Systems Engineering (MBSE) tool(s).** For a representative multi-agent UAS system within the environments of interest, capture a baseline set of SySML views. In particular, begin developing the Parametric View.

- **Transition the baseline architecture to the Monterey Phoenix environment.** The MP process will be used to identify alternate events for each actor in each scenario. With this "superset" of information, we can automatically generate all possible use case variants within a scope limit to provide a scenario coverage that far exceeds what could be done with only a manual effort. This provides a more substantial reference data set the system architecture must satisfy (containing many more possible behaviors including off nominal cases), and a more complete set of input data to ensuing cost and performance analyses. MP can also provide cost attributes.

- **Utilize the executable architecture modeling framework of MP to perform automated assertion checking and find counterexamples of behavior that violate the expected system's correctness**. This supports a complete cycle of automated system

verification/validation impossible with existing SysML/UML technologies. Exhaustive/representative sets of scenarios generated by MP will be transformed by automated tools into implementation testing suites, another huge benefit providing the continuity of design from the early architecture models to the implementation.

- **Design and Demonstrate ISR UAV tradespace**. The power of new tools such as MagicDraw and ModelCenter is the integration with analysis tools, across a network or on the same machine. SySML has traditionally only captured and documented the operational concept, system requirements, activities/tasks, organizations and information flows, including possible physical instantiations. New MBSE tools facilitate analysis such as optimization, simulation, design of experiments, assessment, sensitivity analysis and statistical hypothesis testing and regression. For this project, such trades and characterizations could examine collaborative and vehicle swarming algorithms, increasing autonomy on multi-vehicle, and single operator operations. Likewise, the effects on environmental variables within the architecture, such as communications and/or GPS jamming, air defenses, evasion, camouflage, and other factors could define the scenarios. These types of trades demonstrate how a business case for varying technologies, capabilities or designs could be accomplished, if cost information is included.

- **Develop life cycle cost model interfaces for the various components of the architecture, and embed them within a larger stochastic life cost estimating approach to evaluate cost effectiveness in an uncertain future environment**. Cost data will be attached to every actor and event in every possible scenario, computing the cost of each scenario, were it to occur. A probability of occurrence for each possible scenario will be generated, providing for highly refined overall cost estimates (for operations, for maintenance, and perhaps earlier lifecycle phases) within a specified confidence interval.

With the above steps, this application will use MP with cost modeling to enhance tradespace analysis of UAV systems. The executable integrated architecture will provide for evaluation of UAV technologies and/or design alternatives across a range of operational scenarios utilizing MBSE tools and notations.

Current commercial MBSE tools support creation of a very small number of operational scenario instances compared to what is possible. The validity of these tradespace analyses, however, stands to be substantially improved by expanding the number of scenario variants considered, to include a wider range of possible nominal and off nominal behaviors in both the system under design and the environment. In particular, resulting cost estimates (e.g., for UAV software development, as well as UAV missions during operations & maintenance) are impacted by underrepresentation of possible scenarios and the lack of probability data on those scenarios. One of our objectives is to increase the resolution of source data used for cost model computations. We will compare, contrast, and possibly integrate methods in Phase 5 for cost modeling in MP and SysML depending on the Phase 4 results.

The technical approach involves cycling AFIT-developed operational scenarios through the MP modeling process, whereby alternate events are captured for each actor in each scenario. This will produce a superset of scenario variants from the behavior models, suitable for input to tradespace analysis models and cost model hooks developed in SysML.

With this we can capture lifecycle cost attributes for each function point in the architecture, based on internal and external interactions in the MP models. We can also capture cost attributes for each actor and each event in each generated scenario for use in mission cost effectiveness analyses. Phase 5 will develop further improvements to MP. Based on Phase 4 results these may include an improved event trace generator and a user-friendly GUI.

8.3.2 Task 3. Next-Generation, Full-Coverage Cost Estimation Model Ensembles

NPS will continue extending the scope and tradespace interoperability of cost models and tools in Phases 4 and 5. This is based on stakeholder feedback in earlier phases for parametric model enhancements and tool automation improvements.

Leveraging Phase 3 cost driver research, Phase 4 will develop prototype systems engineering and software cost models and tools for piloting and refinement, and extend the estimation capabilities toward full-coverage in conjunction with USC.

The cost modeling activities will engage domain experts for Delphi estimates, evolve baseline detailed definitions of the cost driver parameters and rating scales for use in data collection, and gather initial data and determine areas needing further research to account for wide differences between estimated and actual costs. Phase 5 will continue the extension in scope of the models and tools and their piloting and refinement.

For tool interoperability we will integrate cost models in different ways with MBSE architectural modeling approaches and as web services (also part of Task 2 piloting). We will also automate systems and software risk advisors that operate in conjunction with the cost models.

We will expand on earlier phase results for cost modeling web services. We previously developed a working prototype web service for Orthogonal Defect Classification Constructive Quality Model (ODC COQUALMO) supporting tool interoperability (costing in the cloud). COQUALMO was demonstrated in Phase 1 with only a subset of cost factors. Per interested stakeholders we will develop full implementations of selected parametric cost models in Phases 4 and 5.

NPS will provide domain expertise to USC and Georgia Tech for the SysML cost model integration effort. We will add the COCOMO software cost model formulas, and risk assessment capabilities for Expert COSYSMO 3 and Expert COCOMO 4. In Phase 5, and we'll continue those and evaluate Monte-Carlo approaches within SySML.

This task is also tied to Task 2 piloting with MBSE cost model interfaces. In Phases 4 and 5 we will assess MP for automatically providing cost information from the architectural models. This is analogous to the SySML method for extracting attributes and they will be compared. MP will be used to extract software sizing information. It will generate function point measures which will be input into COCOMO. We will also assess how MP architectural elements can be mapped into systems engineering cost model inputs.

8.4 REFERENCES

1. Tradespace and Affordability – Phase 1 A013 - Final Technical Report SERC-2013-TR-039-1, Systems Engineering Research Center, July 2013
2. Tradespace and Affordability – Phase 2, A013 - Final Technical Report SERC-2013-TR-039-2, Systems Engineering Research Center, December 2013
3. R. Madachy and R. Valerdi, Automating Systems Engineering Risk Assessment, Proceedings of the 8th Annual Conference on Systems Engineering Research, 2010
4. R. Madachy, Heuristic Risk Assessment Using Cost Factors, IEEE Software, May 1997
5. K. Giammarco and M. Auguston, “Well, You didn’t Say not to! A Formal Systems Engineering Approach to Teaching an Unruly Architecture Good Behavior”, Complex Adaptive Systems Conference, 2013
6. M. Auguston and C. Whitcomb, "Behavior Models and Composition for Software and Systems Architecture", ICSSEA 2012, 24th International Conference on Software & Systems Engineering and their Applications, 2012
7. Boehm, B., C. Abts, A.W. Brown, S. Chulani, B. Clark, E. Horowitz, R. Madachy, D. Reifer, & B. Steece (2000). *Software Cost Estimation with COCOMO II*, Prentice Hall.
8. Boehm, B., A. W. Brown, R. Madachy, & Y. Yang, A Software Product Line Life Cycle Cost Estimation Model. *Proceedings of the 2004 International Symposium on Empirical Software Engineering*, ISESE'04, August 19-20 2004, pp. 156-164.
9. Q. Redman, A. T. Crepea, G. Stratton, (2008). “Weapon Design Trade Off Using Life Cycle Costs”, Raytheon Corporation, URL: http://www.galorath.com/images/uploads/3_-_Quentin_Redman_-_Design_trades_using_Life_Cycle_Costs_short_version.pdf
10. Koskinen, Jussi , (2010). “Software Maintenance Costs”, Jyväskylä: University of Jyväskylä. URL: http://users.jyu.fi/~koskinen/smcosts.htm
11. Boehm, B., J. Lane, and R. Madachy. 2010. "Valuing System Flexibility via Total Ownership Cost Analysis." Proceedings of the NDIA SE Conference, October 2010, San Diego, CA, USA.
12. Robert Matthews, Robert Sweeney, Ken Stanka, “FACE Reference Architecture Business Model White Paper”, Naval Air Systems Command, June 2013
13. Department of Defense, MIL-STD-881C, Work Breakdown Structures for Defense Materiel Items (WBS), October 3, 2011

8.5 APPENDIX - PRODUCT LINE MODELING BACKGROUND

A product line approach provides multiple benefits with respect to ilities across all DoD domains. Affordability gains accrue from reusing common pieces in different systems/products that share features. Furthermore, systems can be fielded faster leading to increased overall mission effectiveness. Flexibility is enhanced increasing the option space. These benefits occur because previously built components reduce the effort and enable more rapid development.

Relevant MPT frameworks for assessing product line aspects are described next. These parametric approaches determine the TOC for various levels of investment in product line architecting. The investment effort is the analysis of the commonalities and variabilities across a product line of similar systems, and building in flexibility to enable reuse or easy adaptation of common components, and plug-compatible interfaces for the variable components.

Product Line Modeling for Affordability and Ility Trades

The Constructive Product Line Investment Model (COPLIMO) is used to assess the costs, savings, and return on investment (ROI) associated with developing and reusing software product line assets across families of similar applications 8 COPLIMO is based on the well-calibrated COCOMO II model 7 with 161 data points.

It includes parameters which are relatively easy to estimate early and be refined as further information becomes available. One can perform sensitivity analyses with the model to see how the ROI changes with different parameters.

Most product line cost models focus on development savings, and underestimate the savings in Total Ownership Costs (TOC). COPLIMO consists of a product line development cost model and an annualized post-development life cycle extension to cover full lifecycle costs. It models the portions of software that involve product-specific newly-built software, fully reused black-box product line components, and product line components that are reused with adaptation.

More elaborate versions of COPLIMO include additional reuse parameters while covering software maintenance as well as development. Additional features such as present-value discounting of future savings and Monte Carlo probability distributions have been added.

The COPLIMO framework has been instantiated and extended at the systems level, used to assess flexibility and ROI tradeoffs. Some of these extensions and applications are described next.

<u>TOC Models for Valuing Product Line Flexibility</u>

The following approaches extend COPLIMO for a TOC analysis for a family of systems. The value of investing in product-line flexibility using Return On Investment (ROI) and TOC is

assessed with parametric models adapted from the basic COPLIMO model. The models are implemented in separate tools available to all SERC collaborators:

- System-level product line flexibility investment model.
- Software product line flexibility investment model. The detailed software model includes schedule time with NPV calculations.

Figure 37 shows the inputs and outputs for the system-level product line model.

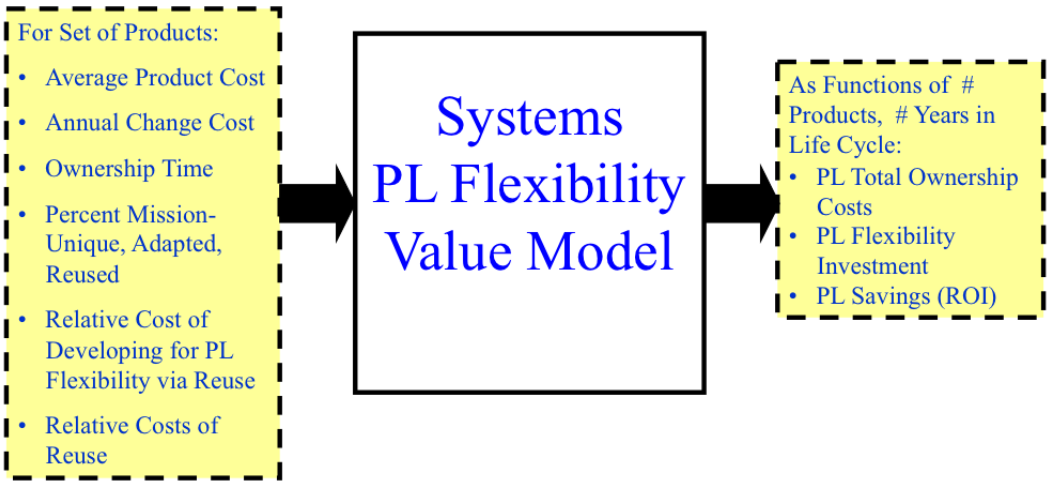


Figure 37: Systems Product Line Flexibility Value Model

The example shown below represents a family of seven related systems with three-year ownership durations. It is assumed annual changes are 10% of the development cost. Within the family of systems, each is comprised of 40% unique functionality, 30% adapted from the product line and 30% reused as-is without changes. Their relative costs are 40% for adapted functionality and 5% for reused. The up-front investment cost in flexibility of 1.7 represents 70% additional effort compared to not developing for flexibility across multiple systems. **Figure 38** shows the consolidated TOC and ROI outputs.

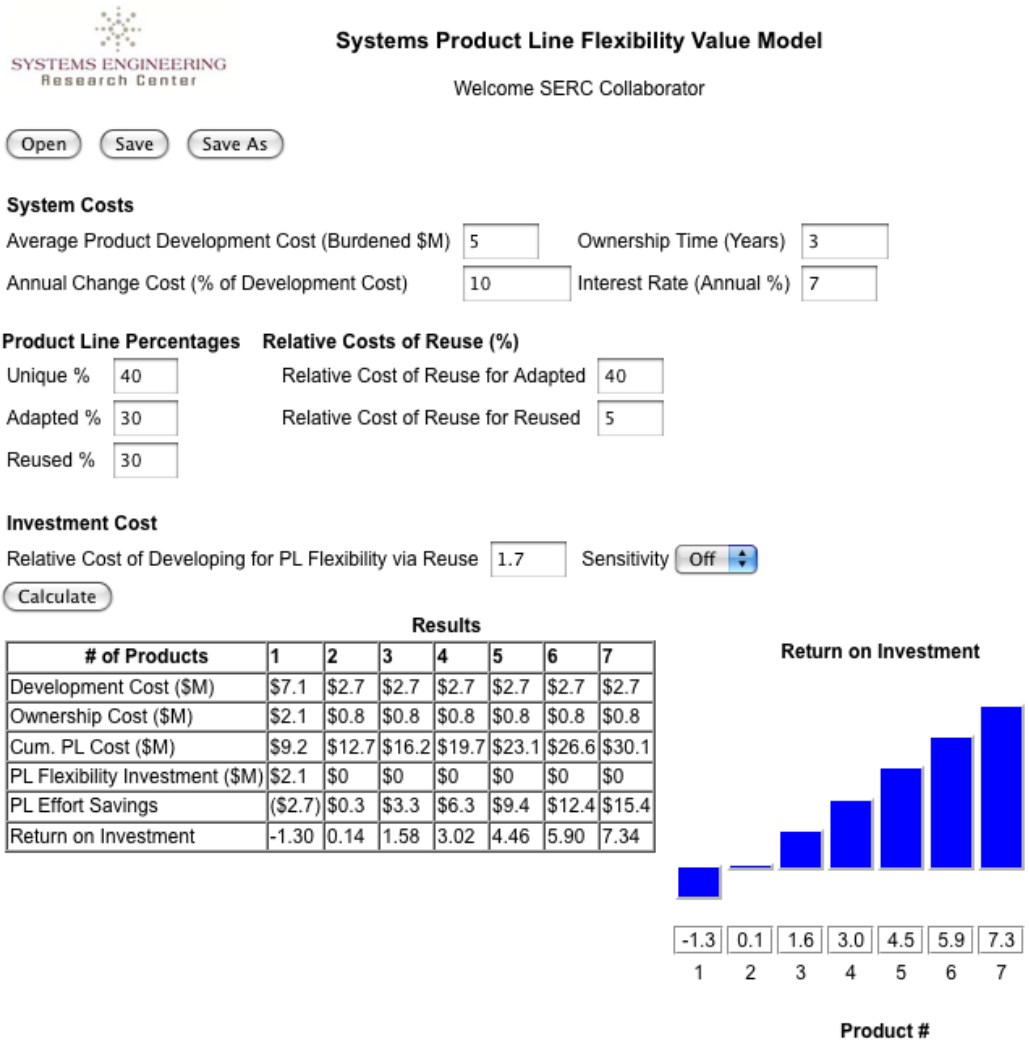


Figure 38: Product Line Flexibility TOC and ROI Results

However, it is desired to evaluate ranges of options and assess the sensitivity of TOC. The tools allow for a range of relative costs as shown in **Figure 39** for sensitivity runs. The results show that the model can help projects determine "how much product line investment is enough" for their particular situation. In the **Figure 39** situation, the best level of investment in developing for reuse is an added 60%.

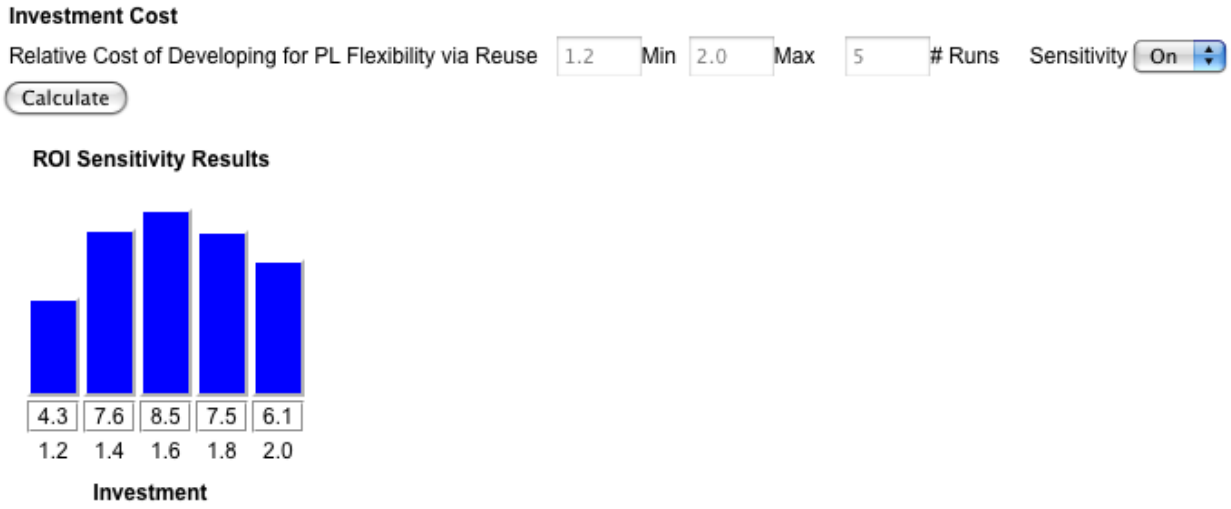


Figure 39: Example Sensitivity Analysis (ROI Only)

Other types of sensitivity analyses can be conducted. **Figure 40** shows example results of assessing the sensitivity of TOC across a range of product ownership durations.

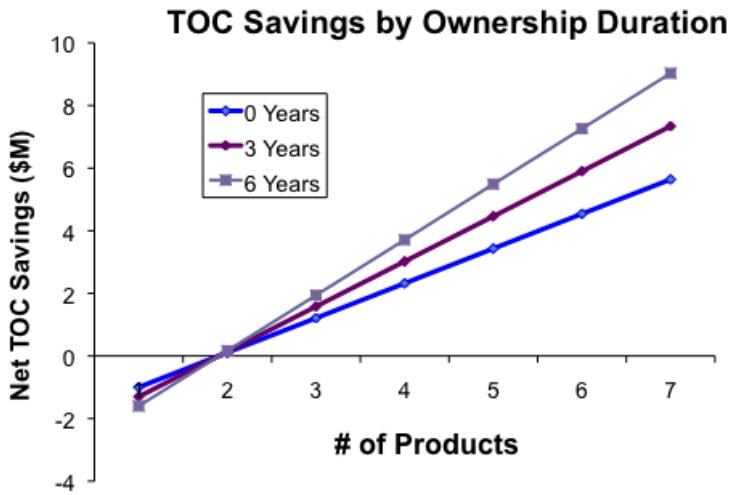


Figure 40: TOC Sensitivity by Ownership Duration Results

The TOC model can also be used in an acquisition decision situation to show that if a project proposes a stovepipe single-product point solution in an area having numerous similar products, and has not done an analysis of the alternative of investing in a product line approach, the project's TOC will represent a significantly higher cost to DoD and the taxpayers.

The general model was enhanced to handle specific DoD application domains, and added initial Monte Carlo simulation capabilities. It incorporates the life cycle cost ratios for Operations and Support (O&S) for hardware O&S cost distributions were derived from 9 and software from 10.

Setting the life cycle cost ratios as a function of system type in the tables impacts the general TOC Product Line model inputs for Ownership Time and Annual Change Cost. The user chooses a system type and ownership time, which invokes a calculated annual change costs for the relevant domain.

The next example illustrates a domain-specific analysis for a missile system with a demonstration of Monte Carlo simulation. The initial case study was for a general system, but in this scenario the user specifies a missile system for O&S life cycle cost defaults.

A missile product line development with three year ownership time is being evaluated. The user chooses the Missile System Type, and sets Ownership Time to 3 years. With these inputs, the pre-calculated Annual Change Cost = 12%/3 years = 4%. The results are in **Figure 41**.

Shown also are the optional Monte Carlo results from varying the relative cost of developing for flexibility. The means are listed with the ROI distribution graph. All input parameters are open to variation for more sophisticated Monte Carlo analysis in follow-on work, per the next section on proposed next steps.

Open Sa

Aircraft
Ground Vehicle
✓ Missile
Ship

System Type

System Costs

Average Product Development Cost (Burdened $M) 5 Ownership Time (Years) 3

Annual Change Cost (% of Development Cost) 4 Interest Rate (Annual %) 7

Product Line Percentages **Relative Costs of Reuse (%)**

Unique % 40 Relative Cost of Reuse for Adapted 40

Adapted % 30 Relative Cost of Reuse for Reused 5

Reused % 30

Investment Cost

Relative Cost of Developing for PL Flexibility via Reuse 1.7

Calculate Monte Carlo On

Means

# of Products	1	2	3	4	5	6	7
Development Cost ($M)	$7.1	$2.7	$2.7	$2.7	$2.7	$2.7	$2.7
Ownership Cost ($M)	$0.9	$0.3	$0.3	$0.3	$0.3	$0.3	$0.3
Cum. PL Cost ($M)	$8.0	$10.9	$13.9	$16.9	$19.9	$22.9	$25.9
PL Flexibility Investment ($M)	$2.1	$0	$0	$0	$0	$0	$0
PL Effort Savings	($2.4)	$0.3	$2.9	$5.5	$8.1	$10.7	$13.3
Return on Investment	-1.12	0.12	1.36	2.60	3.84	5.08	6.32

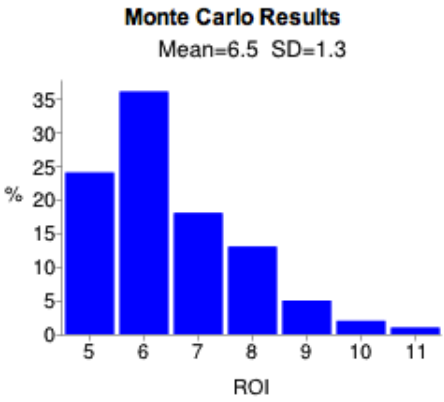


Figure 41: DoD Application Domain and Monte Carlo TOC-PL Results

Summary

The TOC system product line models provide strong capabilities for analyzing alternative approaches to system acquisition and the effects on TOC. They show that if total life cycle costs are considered for development and maintenance, product lines can have a considerably larger payoff, as there is a smaller base to undergo corrective, adaptive, and perfective maintenance.

There are other significant product line benefits besides life cycle cost savings, such as rapid development time and adaptability to mission changes. The models provide an easy-to-use framework for performing these broader ility and affordability analyses.

The models also demonstrate that not all attempts at product line reuse will generate large savings. A good deal of domain engineering needs to be done well to identify product line portions of the most likely to be product-specific, fully reusable, or reusable with adaptation. Much product line architecting needs to be done well to effectively encapsulate the sources of product line variation.

Extensions can be added including the effects of varying product sizes, change rates, product line investment costs, and degrees of reuse across the products in the product line. The models could be combined with other complementary models involving real options, risk assessments, or tradeoffs among flexibility aspects such as evolvability, interoperability, portability, or reconfigurability; or between flexibility aspects and other –ilities such as security, safety, performance, reliability, and availability.