2016-02-10

System Qualities Ontology, Tradespace and Affordability (SQOTA) Project Phase 4

Jacques, David; Columbi, John; Ryan, Erin; Ender, Tommer; Peak, Russell; Sitterle, Valerie; Curry, Michael; Rhodes, Donna; Ross, Adam; Madachy, Ray...

Systems Engineering Research Center (SERC)

System Qualities Ontology, Tradespace and Affordability (SQOTA) Project – Phase 4

Technical Report SERC-2016-TR-101

February 10, 2016

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

Contract Number: H98230-08-D-0171 TO 0031, RT 127

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

This material is based upon work supported, in whole or in part, by the U.S. Department of Defense through the Office of the Assistant of Defense for Research and Engineering (ASD(R&E)) under Contract HQ0034-13-D-0004.

Any views, opinions, findings and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the United States Department of Defense nor ASD(R&E).

INTRODUCTION

This task was proposed and established as a result of a pair of 2012 workshops sponsored by the DoD Engineered Resilient Systems technology priority area and by the SERC. The workshops focused on how best to strengthen DoD's capabilities in dealing with its systems' non-functional requirements, often also called system qualities, properties, levels of service, and –ilities. The term –ilities was often used during the workshops, and became the title of the resulting SERC research task: "ilities Tradespace and Affordability Project (iTAP)." As the project progressed, the term "ilities" often became a source of confusion, as in "Do your results include considerations of safety, security, resilience, etc., which don't have "ility" in their names?" Also, as our ontology, methods, processes, and tools became of interest across the DoD and across international and standards communities, we found that the term "System Qualities" was most often used. As a result, we are changing the name of the project to "System Qualities Ontology, Tradespace, and Affordability (SQOTA)." Some of this year's university reports still refer to the project as "iTAP."

EXECUTIVE SUMMARY

Motivation and Context

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 systems of systems that are full or parts of net-centric systems of systems and enterprises. "Effectiveness" covers the full range of needed System Qualities (SQs) such as reliability, availability, maintainability, safety, security, performance, usability, scalability, interoperability, speed, versatility, flexibility, and adaptability, along with composite attributes such as resilience, affordability, 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-137, System Qualities Ontology, Tradespace and Affordability (SQOTA) project is on strategy 3, although its capabilities also support strategies 1, 2, and 4. It particularly focuses on the tradespace among a system's qualities, also called non-functional requirements or system Ilities. The SQs 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 SQ requirement to a system's specification tends to have a systemwide, multiplicative effect on the system's cost and schedule. Also, SQs 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 in a functional traceability matrix, as the verification traces 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 SQ requirements are the source of many 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-SQ 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).

SQOTA 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), SQOTA 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. SQOTA Phase 4 has included upgrading the models to reflect emerging trends such as changeability to accommodate increasingly rapid change, maintainability to address total cost of ownership, changeability, and availability, 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, Phases 3 and 4 have been developing and evolving cost-schedule-performance tradespace models to provide stronger formulas on which to define formulaic incentives. Further, the Wayne State and Penn State contributions to set-based design and MIT epoch-era analysis approach provide frameworks for evolving the models to accommodate predicted and unpredicted change.

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 commanders 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 developed an ontology of such attributes to provide a stronger basis for defining logistics performance. It has been refined and extended in Phase 4, particularly to extend the capabilities to address software logistics.

Use Modular Open Systems Architecture to stimulate innovation. SERC has been collaborating with TARDEC and NAVSEA in Phases 3 and 4 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, Phases 3 and 4 have

been developing cost-schedule-performance tradespace models to provide stronger formulas on which to define the value of higher levels of performance, including life cycle cost effects.

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. Phases 3 and 4 research have 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 SQOTA Phase 2, including knowledge of Department of Defense (DoD) SQ priorities; foundations and frameworks for SQ tradespace analysis; extension and tailoring of existing SQOTA 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. SQ 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. SQ Foundations and Frameworks. This effort helped to build SQOTA foundations by elaborating key frameworks (process-based, architecture-based, means-ends based, value-based), anticipating further subsequent elaboration via community efforts.

3. SQ-Oriented tool demos and extension plans. This effort created initial demonstration capabilities from strong existing SERC SQ 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 SQ tradespace and affordability analysis that provided an initial common framework for reasoning about SQs, 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' SQ 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 SQ 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 SQ 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: SQOTA Foundations and Frameworks. Phase 2 activities expanded the set of SQs 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 SQs, and developed prototype tools for representing and applying the results.

Task 2. SQOTA Methods and Tools Piloting and Refinement. The SQ-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 pilot program involved advanced pilot training aircraft, simulators and course instructional elements. Its pilot organizations were the Air Force Life Cycle Management Center and the Air Education and Training Command. GTRI's Framework for Assessing Cost and Technology (FACT) was extended beyond its initial support of

USMC, and attracted several Army and Navy programs interested in piloting, extending, and tailoring its capabilities to other domains.

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, along with explorations of sources of data for calibrating the models.

Phase 3 Objectives, Approach, and Results

Task 1: SQ Foundations and Frameworks. MIT's Phase 2 research refined an SQs semantic basis for change-related SQs. and developed prototype tools for formal analysis of the results. Phase 3 extended the SQs semantic basis for change-related SQs, resulting from continuing literature review of SQs, 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 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 SQ definitions, reviewed existing SQ definition standards, developed an initial SQs ontology reflecting 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.

Task 2: SQ-Oriented tool demos and extension plans. Phase 2 effort created initial demonstration capabilities from strong existing SQ 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 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, WSU and PSU advanced the SQOTA 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. TARDEC was actively engaged as a partner for co-development, piloting and transition into use. The GTRI FACT-related capabilities were strongly co-funded and enhanced by and for the Army Engineer R&D Center (ERDC), other Army, Navy, and further USMC programs, including strengthening and extension of the infrastructure for supporting and extending the initial FACT capabilities.

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 identified further sources of data and some shortfalls in current models in these areas, and developed requirements and draft frameworks for the next-generation models. These have been 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 model to better estimate system development costs.

Phase 4 Objectives, Tasks, and Results

Task 1. SQ Foundations and Frameworks

Rather than attempt a breadth-first elaboration of the 176 SQ Synergies and Conflicts strategies in the 7x7 matrix, including its ontology elements of Referents, States, Processes, and Relations for each strategy, the USC ontology-based research did a depth-first research effort on a particular SQ that touches all of the four major SQ categories. This SQ is Maintainability. It clearly drives Life Cycle Efficiency, as typically at least 75% of a system's Total Cost of Ownership is spent on operations and maintenance. It is one of two means for achieving Changeability, involving external change vs. the internal change accomplished by Adaptability. It is clearly key to Dependability, as Maintainability in terms of Mean Time to Repair (MTTR) is the key relation between Reliability in terms of Mean Time Between Failures (MTBF) and Availability in the relation Availability = MTBF / (MTBF + MTTR). And the key systems aspects being depended-upon are primarily the components of Mission Effectiveness.

This depth-first approach thus provided insights on the overall Product Quality ontology structure without having to consider all of the 176 strategies in depth. The insights resulted in changes to the SQ terminology, as shown in the main description of the Phase 4 Results. Examples are changing Resource Utilization to Life Cycle Efficiency, to be more compatible with the Better Buying Power terminology, and changing Flexibility to Changeability, to be better aligned with the MIT Quality In Use ontology structure.

The MIT Quality In Use ontology structure was refined to address further semantic aspects, and requirements for a translation layer to facilitate its use were developed, as elaborated in the main Phase 4 results section. Similarly, the U. Virginia Phase 4 research on formalizing both the MIT and USC ontologies is elaborated in the main Phase 4 results section. An initial semantic diagram relating the USC and MIT terms and relationships is also presented in the main Phase 4 results section.

Task 2. SQ-Oriented tool demos and extension plans.

The USC depth-first exploration of Maintainability identified the need for a better balance of attention during the system acquisition phase between optimizing on system acquisition cost-effectiveness and optimizing on system life-cycle cost-effectiveness, particularly for software, due to the major differences between software and software logistical aspects. This led to the development of a proposed framework of Maintainability Readiness Levels for Software-Intensive Systems. Again, details are provided in in the main Phase 4 results section.

Other Task 2 Phase 4 Objectives, Tasks and Results summaries will be provided later.

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

The Task 3 Phase 4 Objectives, Tasks and Results summaries will be provided later.

References

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

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

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

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

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

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

(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 Mr. Michael Curry
Massachusetts Institute of Technology	Dr. Donna Rhodes Dr. Adam Ross, Co-PI
Naval Postgraduate School	Dr. Ray Madachy, Co-PI Dr. Kristin Giammarco
Pennsylvania State University	Dr. Michael Yukish, Co-PI
University of Southern California	Dr. Barry Boehm, PI Dr. Jo Ann Lane, Co-PI Dr. Bradford Clark Ms. Celia Chen Mr. Jim Alstad
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

1. University of Southern California (USC)

.

1.1 Phase 4 Results

1.1.1 Foundations: SQ Ontology Extensions and Refinements

Rather than attempt a breadth-first elaboration of the 176 SQ Synergies and Conflicts strategies in the 7x7 matrix, including its ontology elements of Referents, States, Processes, and Relations for each strategy, the USC ontology-based research did a depth-first research effort on a particular SQ that touches all of the four major SQ categories. This SQ is Maintainability. It clearly drives Life Cycle Efficiency, as typically at least 75% of a system's Total Cost of Ownership is spent on operations and maintenance. It is one of two means for achieving Changeability, involving external change vs. the internal change accomplished by Adaptability. It is clearly key to Dependability, as Maintainability in terms of Mean Time to Repair (MTTR) is the key relation between Reliability in terms of Mean Time Between Failures (MTBF) and Availability in the relation Availability = MTBF / (MTBF + MTTR). And the key sysems aspects being depended-upon are primarily the components of Mission Effectiveness.

This depth-first approach thus provided insights on the overall Product Quality ontology structure without having to consider all of the 176 strategies in depth. The insights resulted in changes to the SQ terminology, as shown in the main description of the Phase 4 Results. Examples are changing Resource Utilization to Life Cycle Efficiency, to be more compatible with the Better Buying Power terminology, and changing Flexibility to Changeability, to be better aligned with the MIT Quality In Use ontology structure.

1.1.2 The Key Roles of Maintainability in an Ontology for System Qualities

Abstract. In our INCOSE IS 2015 paper, "An Initial Ontology for System Qualities," (SQs), we provided an IDEF5 class hierarchy of upper-level SQs, where the top level reflected classes of stakeholder value propositions (Mission Effectiveness, Resource Utilization, Dependability, Flexibility), and the next level identified means-ends enablers of the higher-level SQs. In experimenting with, refining and formalizing the ontology, we focused on a depth-first approach on a chosen SQ: Maintainability. It is key to reducing 75% of most systems' life cycle costs. Also Maintainability plays key roles in three of the four top-level SQs: Resource Utilization, Dependability, and Flexibility. Dependability also needs Maintainability to relate Reliability to Availability; and Flexibility also needs Maintainability to address new system challenges and opportunities. This paper summarizes resulting changes in the SQ ontology, and also provides examples of Maintainability need and use, quantitative relations where available, and summaries and references on improved practices.

1. IMPROVEMENTS IN THE SQ ONTOLOGY

The nature of the SQs 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. However, functional requirements are much more emphasized in system acquisitions, via such practices as traceability matrices (the SQs generally trace to the entire system), initial system functional review milestones, function-oriented system definition diagrams, functional work breakdown structures and associated earned value milestones. Emerging trends such as systems of systems and rapidly-changing competitive needs and opportunities present major project challenges for SQs such as Maintainability.

1.1 Improvements in the Top-level Class Hierarchy. Tables 1 and 2 show the revisions (in italics) to the upper levels of the initial SQ stakeholder value-based means-ends class hierarchy in our INCOSE IS 2015 paper (Boehm & Kukreja 2015) brought about by our recent Maintainability research and formalization of the SQ relationships.

Table 1. Upper Levels of Initial Stakeholder Value-Based SQ Means-Ends Hierarchy

Stakeholder Value-Based SQ Ends	Contributing SQ 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, Robustness
Flexibility	Modifiability, Tailorability, Adaptability
Composite QAs	
Affordability	Mission Effectiveness, Resource Utilization
Resilience	Dependability, Flexibility

Table 2. Upper Levels of Revised Stakeholder Value-Based SQ Means-Ends Hierarchy

Stakeholder Value-Based	Contributing SQ Means

SQ Ends	
Mission Effectiveness	Stakeholders-satisfactory balance of Physical Capability, Cyber Capability, Human Usability, Speed, Endurability, Maneuverability, Accuracy, Impact, Scalability, Versatility, Interoperability, *Domain-Specific Objectives*
Life Cycle Parsimony	*Development and Maintenance* Cost, Duration, Key Personnel, Other Scarce Resources; Manufacturability, Sustainability
Dependability	Reliability, Maintainability, Availability, Survivability, Robustness, *Graceful Degradation,* Security, Safety
Changeability	*Maintainability*, Modifiability, *Repairability*, Adaptability
Composite QAs	
Affordability	Mission Effectiveness, *Life Cycle Parsimony*
Resilience	Dependability, *Changeability*

The main class of success-critical stakeholders are the system's mission operators, end-users, usage managers, interoperators, and support personnel with respect to the system's operational mission. They will be most concerned with the components of Mission Effectiveness above, but also with domain-specific SQs such as Auditability for finance, In-Process Visibility for supply chains, and Health Maintainability for medical domains.

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 SQ is often called Cost-Effectiveness and the resource expenditures often called Affordability, but the ontology follows INCOSE, NDIA, and MORS in interpreting Affordability as a cost-effectiveness composite SQ, and categorizes the expenditures as Life Cycle Parsimony, which is explicitly elaborated into both Development and Maintenance Costs.

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, boarding an airplane, using a chain saw, providing credit card and social security numbers, etc.). 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). Graceful Degradation has been added to indicate that Survivability can be full or partial.

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. We have generalized Flexibility to Changeability to be more consistent with the MIT Changeability ontology (Ross & Rhodes,

2015). Its mutually-exclusive means are internal change (Adaptability) and external change (Maintainability), which includes Modifiability to support Changeability and Repairability to support Dependability.

1.2 Elaboration of Maintainability Relationships. In pursuing the various contributions and types of Maintainability as a critical but complex SQ, we have developed Figures 1 and 2 as elaborations of Maintainability relationships. Figure 1 focuses on the role of Maintainability in support of Changeability. It shows that Maintainability and Adaptability are two alternative means of achieving Changeability, external and internal to the system. It also shows that Modifiability and Repairability are two alternative means of achieving Maintainability, one for changes and one for defects. It also shows some enabling (and), but not alternative (or) subclasses of Modifiability, Repairability, and support of verification and validation, called "V&V-ity." These latter are Automated Analysis, Peer Reviews, and Execution Testing; their rating scales are provided in (Boehm 2015), where they were taken from the Constructive Quality Model for estimating delivered software defect density (Boehm *et al*. 2000).

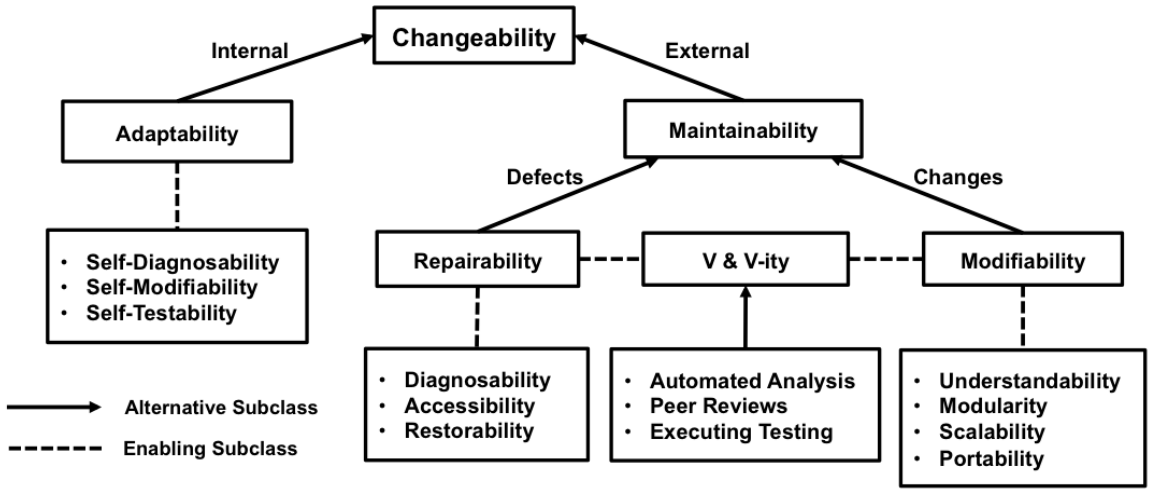


Figure 1. Role of Maintainability for Changeability

Figure 2 further elaborates on how Maintainability supports both Changeability and Dependability, and elaborates on how these two combine to support Resilience. It also elaborates the means-ends subclasses of Reliability, and of Testing in support of V&V-ity.

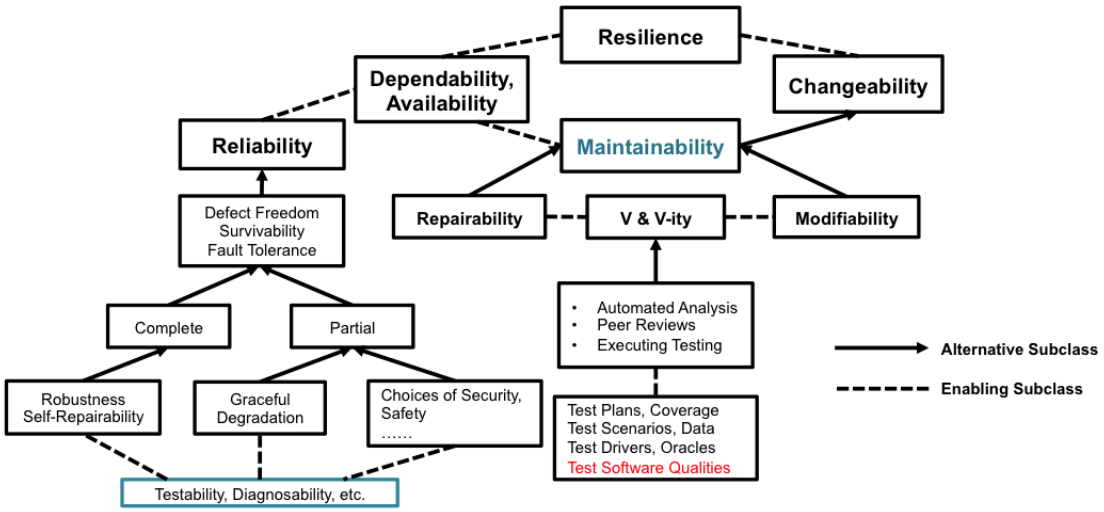


Figure 2. Role of Maintainability for Changeability, Dependability, and Resilience

Figure 2 also emphasizes the many-to-many relationships of Testability, as compared to the one-to-many relationship it has in ISO/IEC 25010 (ISO/IEC, 2011), where Testability only supports Maintainability. It is clear that Testability is also important to the achievement of numerous other SQs, such as the many aspects of Dependability in Figure 2. It also highlights the recursive complexities of the SQ hierarchy, as Testability also involves the test software, such as its classes (performance, stress, regression, etc.) and capabilities (test drivers, monitors, diagnosis aids, data managers, etc.) and the need to ensure its qualities as well, as shown in red in Figure 2.

In summary, Table 1 and Figures 1 and 2 summarize the class hierarchy of the primary stakeholder SQ end value classes of Mission Effectiveness, Resource Utilization, Dependability, and Flexibility, along with their primary means-ends subclasses, and the primary composite SQs of Affordability and Resilience. Each of the SQ classes and subclasses must also be characterized by the variation of their values with respect to their variation by Referents, States, Processes, and Relations, as will be described next in the context of Maintainability.

2. Maintainability Referents, States, Processes and Relations

All too often, SQs are specified as single numbers, when in general their values vary by Referent (primarily by stakeholder viewpoint), by State (the system's internal state and its external environment state), by Process (internal procedures, external operational scenarios), and by their Relations with other SQs. These are discussed below in the context of Maintainability.

2.1. Referents. For Maintainability, all stakeholders will prefer shorter above longer mean times to repair or modify, but their duration values will vary by stakeholder. Security-critical stakeholders will include the time to recertify the systems security, while less-critical stakeholders may be satisfied with the agile proposition that most defects in the current 30-day Scrum can be fixed in the next 30-day Scrum. And their value propositions may affect Maintainability values if there are constraints on the preservation of other SQs such as the

ability to meet real-time performance constraints that require more time and effort to satisfy when making modifications.

2.2. States. The time and effort to effect changes or repair defects will vary by the system's internal state, such as the brittleness of its point-solution architecture, its weak or strong modularity, or for software, the level of its Maintainability Index, to be discussed further below. The system's degree of Maintainability would also depend on aspects of its external state, such as its need to support multiple systems of systems with different stakeholders, each with different priorities and independently-evolving interfaces.

2.3. Processes. Maintainability values will also vary by the nature of the external processes or operational scenarios that the system needs to be involved in, such as the systems of systems processes just discussed. For variation by internal processes, a good example is the quality of the problem report closure means-ends process, as compared to the process summarized in the book Maintainability (Blanchard 1995). 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 led to the definition of enabling sub-SQs such as Diagnosability, Accessibility, Restorability, and V&V-ity.

2.4. Relations. The 7x7 matrix of SQ Synergies and Conflicts in the INCOSE-IS 2015 paper (Boehm&Kukreja, 2015) identifies a number of relations between Maintainability and other desired SQs. For example, tightly-coupled High Performance Computing architectures and software improve computational Speed but reduce Maintainability. Easiest-first agile methods often improve initial Usability, but make architectural commitments that often subsequently degrade Scalability, Safety, and/or Security. On the other hand, investments in nanosensor-based monitoring systems improve both Dependability but also improve Maintainability. For example, aircraft autonomic logistics systems can communicate maintenance needs during flight and communicate them to the landing field, where the maintainers can prepare to turn the aircraft around in hours vs. days.

We try to quantify such relationships where possible. For example the GAO report (GAO 2015) cited in the Abstract identified annual US Government information technology (IT) expenditures of $79 billion, of which $58 billion were in operations and maintenance (O&M). Table 3 provides more detail on fractions of hardware and software costs from the (Redman, 2008) and (Koskinen, 2009) studies.

Table 3. Percentage of Post-Deployment Life Cycle Cost

Hardware (Redman, 2008)	Software (Koskinen, 2009)
• 12% -- Missiles (average)	• 75-90% -- Business, Command-Control
• 60% -- Ships (average)	• 50-80% -- Complex platforms as above
• 78% -- Aircraft (F-16)	• 10-30% -- Simple embedded software
• 84% -- Ground vehicles (Bradley)	

More specifically, our SERC Valuing Flexibility study (Boehm-Lane-Madachy, 2011) developed

models based on TRW technical debt data for single-system life cycle costs and multiple sources on product line investments and life cycle savings. Figure 3 shows how the life cycle costs of individual systems decrease when projects invest in a system's Maintainability in project C, as compared to when they do not, as with projects A and B. Figure 4 shows how investments in product line architectures and reusable components have life cycle Total Cost of Ownership (TCO) savings that increase with product lifetime as well as with number of products in the product line, since only the non-reused components need to be separately modified for each product.

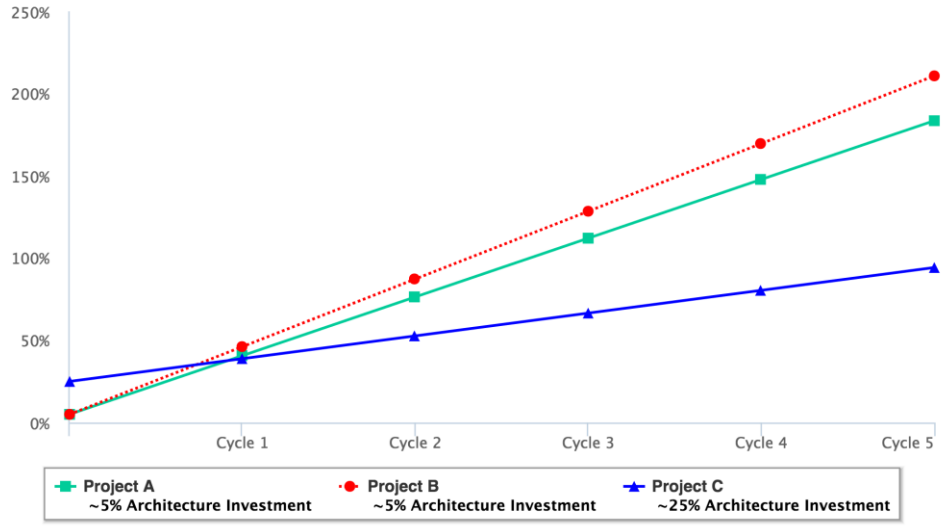


Figure 3. TCO's for Projects A, B, and C (CCPDS-R) Relative to Baseline Costs

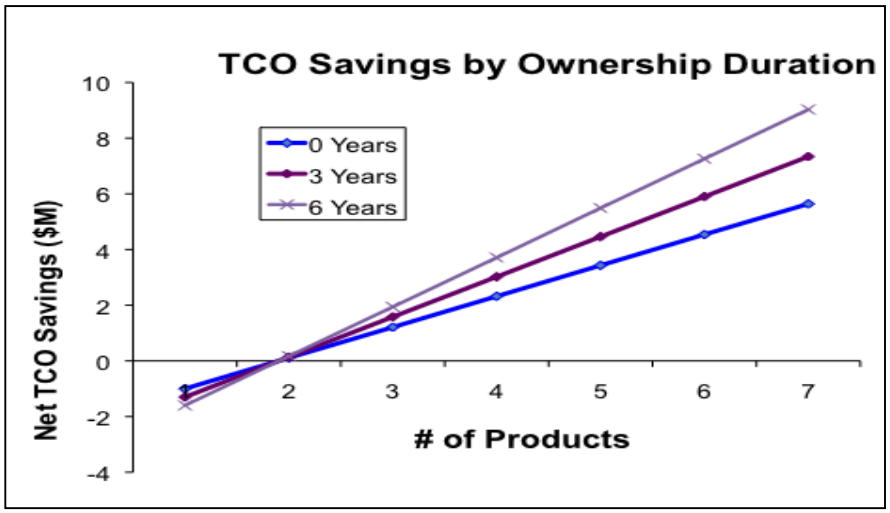


Figure 4. TCO-Product Line sensitivity by ownership duration results.

3. MAINTAINABILITY OPPORTUNITY TREES

Here we present and discuss a pair of Opportunity Trees for improving the Modifiability and Repairability aspects of Maintainability. We have developed similar Opportunity Trees for improving system Cost, Schedule, and Dependability, and find them useful for organizations looking for ways to improve their system qualities. The Opportunity Trees also elaborate the means-ends relationships between SQs and their enablers.

3.1. Modifiability Opportunity Tree. Figure 5 provides an opportunity tree for Modifiability. As with our other opportunity trees for reducing Cost, reducing Schedule, and improving Dependability aspects, it provides a way for systems engineers to improve the system's Modifiability.

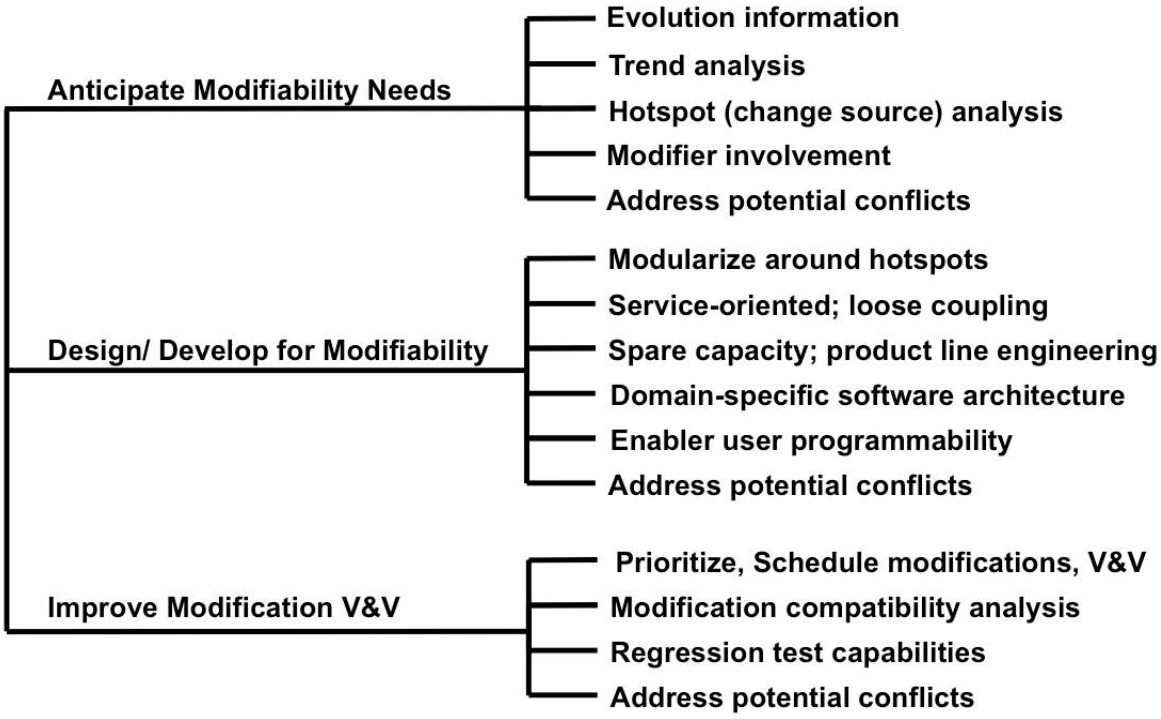


Figure 5. Modifiability Opportunity Tree

Anticipate Modifiability Needs: Evolution Information and Trend Analysis. Often time the requirements for a system acquisition are determined by prioritizing the capabilities, and using a Cost As Independent Variable (CAIV) analysis to determine which capability fits with in the available budget and are to be included in the Request for Proposal (RFP). This unfortunately throws away below-the-line information on the most likely directions of system evolution, often leading to a brittle point-solution architecture as the chosen solution. Including the below-the-line capabilities in the RFP as candidate Evolution Information, and indicating that it should be considered in proposing life cycle architecture is more likely to result in reducing operations and maintenance costs. Trend Analysis is another way to identify likely directions of system evolution.

Hotspot (change source) Analysis. Common sources of change include user interface changes, device driver changes, external interface changes, and changes in non-developmental items

(NDIs). Many commercial-off-the-shelf (COTS) products have new releases every 8-12 months, and continue support for only the latest three releases. Systems employing numerous COTS products will therefore have numerous changes.

Modifier Involvement. Lack of modifier involvement often results in key Modifiability enablers being either neglected or implemented in incompatible ways. It may take some effort to get the modifiers involved, as often they are too busy overcoming the Maintainability shortfalls caused by previous projects lacking modifier involvement.

Addressing Potential Conflicts of Anticipate Modifiability Needs. A previous version of Evolution Information was called Evolution Requirements. Considering these below-the-line desired capabilities as requirements is too strong a term, as there are likely to be many emergent requirements that should have higher priority. Trying too hard to get Modifier Involvement may lead to the modifiers sending someone they can easily do without, which is often worse than having no representative.

Design/Develop for Modifiability: Modularization Around Hotspots. The (Parnas, 1979) paper, "Designing Software for Ease of Extension and Contraction," and earlier Parnas papers identify this Maintainability strategy. Common sources of change that have been encapsulated in modules will contain the change effects within the module rather than having them ripple across the other parts of the system. A quantitative model of this strategy's effect on Affordability is shown in Figure 4 above (Boehm-Lane-Madachy, 2011), based on analysis of change effort data across several TRW projects.

Service Orientation; Loose Coupling. Service loose coupling is a design principle that is applied to the services in order to ensure that the service contract is not tightly coupled to the service consumers and to the underlying service logic and implementation (Sundbo & Gallouj, 2000). Basically, the concept of loose coupling provided by a Service–Oriented Architecture (SOA) component is that it publishes a contract indicating that if it is furnished with a specified set of inputs, it will produce a specified set of outputs, without otherwise interacting with the user's environment (Boehm & Bhuta, 2008; Borges *et.al*., 2004).

Apart from this, SOA strongly advocates development of physically independent service contracts from the service logic (decoupled contract design pattern (Erl, 2008, 401-407)) in favor of interoperability and technology independence. As the contracts are physically independent, there is a need to not only look into the coupling between service consumers and service contracts but also between service contracts and their underlying logic and implementation. This is where the application of this design principle helps in identifying the various types of couplings that exist (inter service as well as intra service) and how to design the contracts in order to minimize negative coupling types and maximize positive coupling types. A service-oriented solution that consists of services having loosely coupled contracts directly supports the increased interoperability goals of service-orientation. By having loose coupling, it increases Resilience, which is a combination of Changeability and Dependability.

Spare Capacity. One technique, which uses spare capacity to create a more reliable and maintainable product, is called "passive redundancy". Passive redundancy uses excess capacity to reduce the impact of component failures. One usage of this technique is to incorporate it with a fault tolerance strategy. Early cost models such as PRICE S (Freiman & Park, 1979) and COCOMO (Boehm, 1981) drew on data from (Williman & O'Donnell, 1970) that the cost per line of code becomes asymptotically high as software fills up the available execution and storage capacity, leading to common requirements for spare capacity and/or easy migration to higher-performance computing resources. Too small spare capacity will result in unreliable products where the software cannot allocate sufficient resource when needed, while too large spare capacity will result in unaffordable software. Early analysis of the data in (Williman & O'Donnell, 1970) indicated that the cost of system overall will be minimized if the software has at least 50 percent to 100 percent more capacity than it absolutely necessary (Boehm, 1973). As a result, having a sufficient spare capacity will provide improvements not only Maintainability, but also Resilience, which is the combination of Changeability and Dependability, and Life Cycle Efficiency.

Product Line Engineering. The Total Cost of Ownership benefits from Product Line Engineering are shown in Figure 4. Product Line Engineering involves developing a Domain-Specific Architecture for the product line by identifying the commonalities and variabilities across a range of products; making reusable components for the commonalities, and providing standard interfaces for the variabilities. It also involves coordinated evolution of the reusable components, providing Spare Capacity for growth and variabilities, and providing asset library functions for project users. Good references are (Clements-Northrop, 2002; Jacobson-Griss-Jonsson, 1997; Poulin, 1998; and Reifer, 1997).

Domain-Specific Architecture. A domain-specific architecture (DSA) comprises "(a) a reference architecture, which describes a general computational framework for a significant domain of applications, (b) a component library, which contains reusable chunks of domain expertise, and (c) an application configuration method for selecting and configuring components within the architecture to meet particular application requirements" (Hayes-Roth *et al*., 1995). The core concept is to generalize knowledge of domain componentry within the same domain so that developers can reuse assets and apply common solutions to common problems (Taylor *et al.,* 2012). This common knowledge or solutions make both development and maintenance easier to understand and perform. Developers can focus more on implementing new features and spend less time develop more related products. Generalizing DSA across product-line or family of applications can be expensive, but one can expect a reduction in application development costs (approximately a quarter) and in maintenance cost (approximately a third) (Tracz, 1995). DSA enables improvement for not only Maintainability, but also Dependability and Life Cycle Parsimony.

Enable User Programmability. In many circumstances, users have found it easier to develop spreadsheet applications for specific needs rather than trying to master and tailor complex general applications to fit their needs. Many special devices with numerous options (e.g., medical infusion pumps; educational robots) have simple special-purpose languages for specifying the options.

Addressing Potential Conflicts of Design/Develop for Modifiability. Multi-Domain Architecting. Committing to several domain ontologies within a system of systems will complicate both Changeability and Dependability, as the special domain assumptions may have incompatibilities causing both ease-of change and dependability-of-change problems. For example, a system of systems involving ground, air, and space operations may have compatible inputs and outputs, but have problems with incompatible internal assumptions in which the ground operations assume a flat earth, the air operations a spherical earth, and the space operations an oblate earth. Thus, ensuring compatible internal assumptions, or coping with incompatible assumptions, will be important for multi-domain architecting. Also, User Programmability may have problems: an IBM study found that 44% of a large sample of spreadsheet programs had defects that would if exercised have had major negative financial outcomes for the organization.

Improve Modification V&V: Prioritizing and Scheduling Modifications and their V&V). For complex systems, incremental development is at least as important for the V&V support system as it is for the system under development. Waiting until near the end of development to create the V&V support system is likely to cause V&V delays, and it loses the opportunity to perform continuous V&V as the system is being developed.

Modification Compatibility Analysis. As above, the three primary V&V strategies are automated analysis, peer reviews, and execution testing. For automated analysis, there are various types of tools that can contribute by V&V-ing the proposed changes' internal consistency, external-interface compatibility, race or deadlock avoidance, common-defect-freedom, or reductions in the software's Maintainability Index (Oman & Hagemeister, 1992). For peer reviews, modification compatibility analysis by experts in the system's quality-objective areas would be effective.

Regression Test Capabilities. Particular capabilities to address are the Testability enablers of having cost-effective test plans, test coverage, test drivers, test oracles (e.g., agreement with previous regression test results), test data management capabilities, and diagnostic capabilities.

Addressing Potential Conflicts of Improve Modification V&V. As identified above, waiting until near the end of development to create the V&V support system is likely to cause V&V delays, and delays in the delivery schedule for the overall system. Another potential conflict with Affordability is to treat each test case as equally important. Prioritizing tests by business value has been shown to be significantly more cost-effective in testing new releases (Li and Boehm, 2013).

b. Repairability Opportunity Tree

Figure 6 provides an opportunity tree for Repairability. It has significant overlaps with the opportunity tree for Modifiability, and its discussion will be more concise. The major source of differences comes from differences in hardware and software Repairability, which will be discussed next.

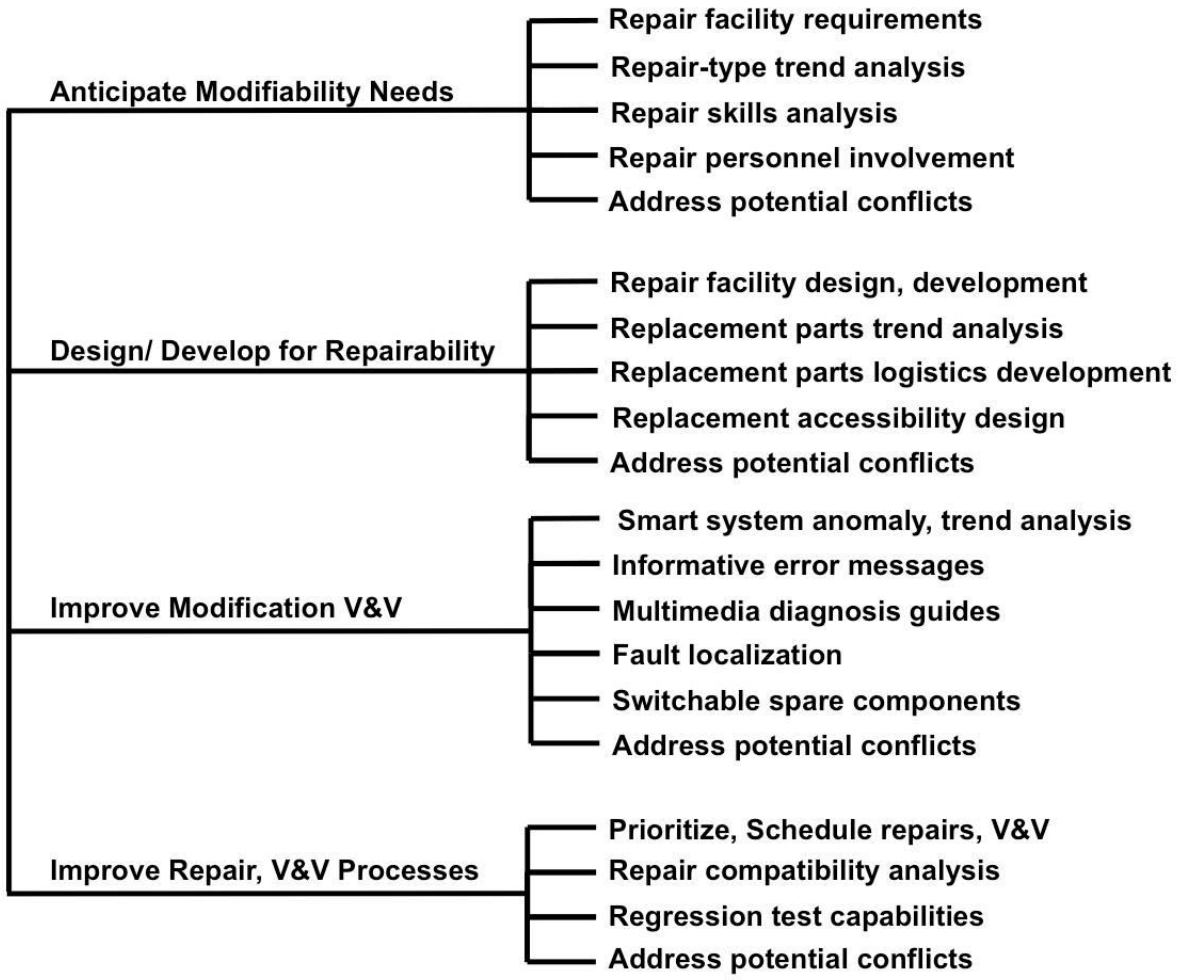


Figure 6. Repairability Opportunity Tree

The list below summarizes the major differences between hardware and software Repairability. Further differences and their discussions are in (Boehm, 1994).

- Software components to not degrade due to wear and fatigue;
- No imperfections or variations are introduced in making copies of software components;
- Software interfaces are conceptual rather than physical; there is no easy-to-visualize three-prong plug and its mate;
- There are many more distinct logic paths to check in software than in hardware;
- The failure modes are generally different. Software failures generally come with no advance warning, provide no period of graceful degredation, and more often provide no announcement of their occurrence;

- Repair of a hardware fault generally restores the system to its previous condition; repair of a software fault does not.

Some implications of these differences are that hardware repair requires a significant logistics activity for replacement of worn or degraded components. In terms of the Anticipate Repairability Needs and Design/Develop for Repairability branches of the Repairability opportunity tree, hardware repair involves a much larger logistics effort than does software repair. On the other hand, software repair generally requires more complex skill levels than does hardware repair. Facilities for hardware repair differ significantly from software repair facilities, although there are benefits from having a single facility addressing both hardware and software and both repairs and modifications. Recent trends in additive manufacturing or 3-D printing are simplifying hardware repair logistics (e.g., initiatives underway to put additive manufacturing capabilities on Navy ships) and moving hardware and software repair closer together.

Similar initiatives are underway in the Improve Repair Diagnosis branch of the Repairability opportunity tree. A good example is the development of smart aircraft-maintainability systems, which populate the aircraft with thousands of nanosensors, which can monitor a component's degree of wear, structural integrity, or battery level, and provide the information in advance so that necessary repair capabilities are pre-positioned to effect rapid repair when the aircraft lands. This provides a much lower mean time to repair and much higher level of Availability, showing a strong synergy in increasing both Changeability and Dependability.

An excellent source on Repairability and Maintainability for hardware and to some extent software is the (Blanchard-Verma-Peterson, 1995) book Maintainability. It identifies and provides guidance for the many sources of post-acquisition cost lying below the acquisition tip of the iceberg. These include operations cost, product distribution cost, maintenance cost, training cost, test and support equipment cost, software cost, technical data cost, supply support cost, and retirement and disposal cost. For individual cost components such as Repair, it identifies and provides guidance on the sequence of key activities involved in repair: detection, preparation for maintenance, localization and isolation, disassembly (access), faulty item repair or replacement, reassembly, alignment and adjustment, and condition verification (checkout).

Improving Repair V&V is similar to its Modification V&V counterpart above.

Addressing Potential Conflicts: Repairability. In many cases, organizations will optimize on Manufacturability or Endurability at the expense of degrading Repairability. A good example of the tradespace among these SQs is the Smartphone Repairability website at https://www.ifixit.com/smartphone-repairability. It shows that the best ratings are 9 out of 10, that most smartphones at least make it easy to replace batteries, but some 1 and 2-rated smartphones make even battery replacement impossible.

Another source of Repairability conflicts involves casually-written maintenance contracts that pay for time consumed during repair, where the contractor's incentive is to take longer and increase income while increasing mean time to repair and Availability. Preferred contracting

approaches pay for outcomes rather than labor hours or number of transactions, called Performance-Based Logistics in the defense community (Insinna, 2015) and terms such as Vested Outsourcing (Vitasek-Ledyard-Manrodt, 2013) in the commercial community.

4. QUANTIFYING MAINTAINABILITY AND ITS ENABLERS

Where possible, we are trying to provide quantitative information to help projects determine how maintenance cost and duration vary as a function of which Maintainability methods, processes, and tools (MPTs) to use in which situations. In the past, we have developed various partial solutions such as lists of Maintainability-enhancing practices (Boehm et al., 1973, 1978); cost model drivers that increase cost to develop but decrease cost to maintain such as required reliability and architecture and risk resolution (Boehm et al., 2000); MPTs that reduce life cycle technical debt (Boehm-Valerdi-Honour, 2008); and quantification of early-increment maintenance costs as sources of later-increment productivity decline in incremental and evolutionary development (Moazeni-Link-Boehm, 2013). Some other good sources on SQ quantification are (Barbacci et al., 1995; Clements-Kazman-Klein, 2002; Dou et al., 2015; Gehlert &Metzger, 2008; Gilb, 1988; Oman & Hagemeister, 1991; Oriol-Marco-Franch, 2014; Ross & Rhodes, 2015; and Sitterle et al., 2014).

In reviewing the literature, we fond that the most widely-used metric was the (Oman & Hagemeister, 1992) Maintainability Index (MI). We experimented on evaluating its use in evaluating the relative Maintainability of a sample of 97 open-source-software (OSS) projects by programming language and application domain (Chen-Shi-Srisopha, 2015). As shown in Figures 7 and 8, we found that the MI of the OSS projects varied by programming language with moderate statistical significance (p=0.079), and that the MI varied by domain with strong statistical significance (p=0.002).

ANOVA

		Sum of Squares	df	Mean Square	F	Sig.
MIwoc	Between Groups	844.599	2	422.299	2.544	0.084
	Within Groups	15602.788	94	165.987		
	Total	16447.386	96			
MIwc	Between Groups	589.095	2	294.548	3.069	0.051
	Within Groups	9022.420	94	95.983		
	Total	9611.516	96			
MI	Between Groups	1044.871	2	522.435	2.614	0.079
	Within Groups	18783.525	94	199.825		
	Total	19828.395	96			

Fig. 7. MI Variation by Programming Language

ANOVA

		Sum of Squares	df	Mean Square	F	Sig.
MIwoc	Between Groups	1541.295	4	385.324	2.378	0.057
	Within Groups	14906.092	92	162.023		
	Total	16447.386	96			
MIwc	Between Groups	741.498	4	185.374	1.923	0.113
	Within Groups	8870.018	92	96.413		
	Total	9611.516	96			
MI	Between Groups	3221.732	4	805.433	4.462	0.002
	Within Groups	16606.663	92	180.507		
	Total	19828.395	96			

Fig. 8. MI Variation by Domain

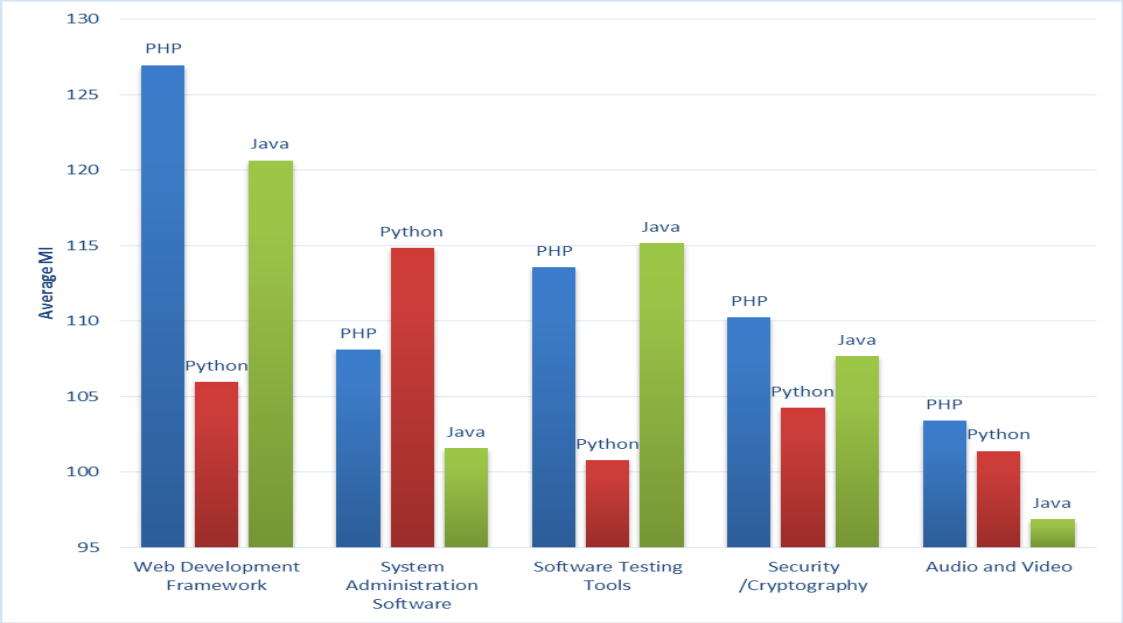


Figure 9. MI Variation by Programming Language and Domain

Figure 9 summarizes the MI variation by programming language and domain. It indicates that PHP may be a good option for projects that desire higher maintainability within the Web Development Framework, Security/Cryptography and Audio and Video domains, that Python may be a good option for System Administrative Software, and that Java may be a good option for Software Testing Tools**.**

In performing the evaluations, we also evaluated the completeness of MI as a determinant of Maintainability. We found that it did well as a purely automated, easy-to-use metric, but would likely be improved with cohesion and coupling metrics such as (Chidamber-Kemerer, 1994). Also, some recent technical debt assessment tools such as Cast and Sonar provide more complete and specific identification of sources of technical debt or reduced Maintainability. However, a mix of automated and human assessments such as the COCOMO II Software Understanding components of structure, application clarity, and self-descriptiveness (Boehm et al., 2000) would generally be stronger. We are preparing to perform further research in these directions.

5. Conclusions and Further Research

System qualities (SQs) 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.

An initial SQ ontology (Boehm&Kukreja, 2015) helps address these problems by clarifying the nature of the SQs. A particularly important SQ is Maintainability. It is key to reducing the roughly 75% of most systems' life cycle costs spent on maintenance. Maintainability plays key

roles in three of the four top-level SQs: Besides Life Cycle Parsimony, Dependability also needs Maintainability to relate Reliability to Availability; and Changeability also needs Maintainability to address new system challenges and opportunities. For Life Cycle Parsimony, models are presented that show the significant returns on investments in single-system and product line Maintainability.

Means-ends diagrams are provided to clarify how aspects of Maintainability relate to other SQs. Complementary means-ends diagrams called Opportunity Trees are provided and elaborated to enable systems engineers to develop strategies for improving the various aspects of Maintainability. They and other Opportunity Trees for cost reduction, schedule acceleration, and Dependability improvement provide a good candidate framework for realizing the vision of providing a Qualipedia as a body of knowledge for SQs.

Some initial experiments in using the (Oman & Hagemeister, 1992) Maintainability Index (MI) to evaluate a software system's maintainability showed promise in evaluating the relative maintainability of 97 open-source software systems as functions of their programming language and application domain. They also provided insights in identifying complementary methods, processes, and tools such as cohesion-coupling analyzers, technical debt analyzers, and human-assisted system understandability analyzers that could strengthen future abilities to evaluate a system's Maintainability.

6. REFERENCES

Barbacci, M., Klein, M., Longstaff, T., & Weinstock, C.1995. "Quality Attributes." *Technical Report CMU/SEI-95-TR-021*.

Blanchard, B., Verma, D., & Peterson, E. 1995. *Maintainability: a key to effective serviceability and maintenance management*. John Wiley and Sons.

Boehm, B, Lane, J. & Madachy, R. 2011. "Total Ownership Cost Models for Valuing System Flexibility." In *Proceedings, CSER.*

Boehm, B. 1973. "Software and Its Impact: A Quantitative Assessment." *Datamation*.

Boehm, B. 1981. *Software Engineering Economics*. Prentice Hall.

Boehm, B., & Bhuta, J. 2008. "Balancing opportunities and risks in component-based software development." *Software, IEEE*. 25(6): 56-63.

Boehm, B., Abts, C., Brown, AW., Chulani, S., Clark, B., Horowitz, E., Madachy, R., Reifer, D., & Steece, B. 2000. *Software Cost Estimation with COCOMO II.* Prentice Hall.

Boehm, B., Brown, J., Kaspar, H., Lipow, M., MacLeod, G., & Merritt, M. 1973. "Characteristics of Software Quality." *NBS Technical Report*; Amsterdam: North Holland, 1978.

Boehm, B., & Kukreja, N. 2015. "An Initial Ontology for System Quality Attributes." In *Proceedings, INCOSE International Symposium,* July.

Borges, B., Holley, K., & Arsanjani, A. 2004. "Delving into Service-Oriented Architecture." http://www.developer.com/java/data/article.php/3409221/Delving-into-Service-Oriented-Architecture.htm

Chen, C, Shi, L., & Srisopha, K. 2015. "How does software maintainability vary by domain and programming language?" In *Proceedings, IEEE SW Tech Conference.*

Chidamber, S. & Kemerer, C. 1994. “A Metrics Suite for Object-Oriented Design” In *Proceedings, IEEE Trans SW Engr.* June. 476-493.

Clements, P., Kazman, R., & Klein, M. 2002. *Evaluating Software Architectures.* Reading MA: Addison Wesley.

Dou, K., Wang, X., Tang, C., Ross, A., & Sullivan, K. 2015. “An Evolutionary Theory-Systems Approach to a Science of the ilities.” *In Proceedings, CSER.*

Erl, T. 2008. *SOA design patterns*. Pearson Education.

Freiman, F., & Park, R. 1979. “PRICE Software Model-Version 3.” *IEEE Catalog No. TH0067-9.*

GAO (U.S. General Accountability Office). 2015. “Additional Opportunities to Reduce Fragmentation, Overlap, and Duplication and Achieve Other Financial Benefits,” *Annual Report GAO-15-404SP,* April 14.

Gehlert, A., Metzger, A. 2008. “Quality Reference Model for SBA.” *S-Cube Report CD-JRA-1.3.2*

Gilb, T. 1988. *Principles of software engineering management*. Reading MA: Addison-Wesley.

Hayes-Roth, B., Pfleger, K., Lalanda, P., Morignot, P., & Balabanovic, M. 1995. “A domain-specific software architecture for adaptive intelligent systems.” *Software Engineering, IEEE Transactions on*, 21(4): 288-301.

Insinna, V. 2015. “Market for Performance-Based Logistics Grows.” *National Defense*, March.

ISO/IEC, Systems and software engineering – SQuaRE system and software quality models standard 25010, ISO/IEC; 2011.

Jacobson, I., Griss, M., & Jonsson, P. 1997. *Software Reuse.* Wiley

Koskinen, J. 2009. *Software maintenance fundamentals*. In Encyclopedia of Software Engineering, Laplante, P., Ed., Taylor & Francis Group.

Li, Q., & Boehm, B. 2013. “Improving scenario testing process by adding value-based prioritization: an industrial case study.” I*n Proceedings of the 2013 International Conference on Software and System Process (pp. 78-87). ACM.*

Oman, P. & Hagemeister, J. 1992. “Metrics for Assessing a Software System’s Maintainability.” In *Proceedings, IEEE Conf. SW Maint.,* March, 337-344.

Oriol, M., Marco, J., & Franch, X. 2014. “Quality models for web services: A systematic mapping.” *Information and Software Technology* (56) 1167-1182.

Parnas, D. 1979. “Designing Software for Ease of Extension and Contraction.” In *Proceedings, IEEE Trans SW Engr.,* March, 128-137.

Poulin, J. 1996. *Measuring Software Reuse.* Addison Wesley

Rainey, L. B., & Tolk, A., eds. 2015. *Modeling and Simulation Support for System of Systems Engineering Applications*. John Wiley & Sons.

Redman, Q. 2008. “Weapon System Design Using Life Cycle Costs.” Raytheon Presentation.

Reifer, D. 1997. *Practical Software Reuse.* Wiley

Ross, A. & Rhodes, D. 2015. “Towards a Prescriptive Semantic Basis for Change-Type Ilities.” *In Proceedings. CSER.*

Sundbo, J., & Gallouj, F. 2000. “Innovation as a loosely coupled system in services.” *International Journal of Services Technology and Management.* 1(1): 15-36.

Taylor, R. N., Medvidovic, N., & Dashofy, E. M. 2009. *Software architecture: foundations, theory, and practice*. Wiley Publishing.

Tracz, W. 1995. “DSSA (domain-specific software architecture): pedagogical example.” *ACM SIGSOFT Software Engineering Notes.* 20(3): 49-62.

Vitasek, K., Ledyard, M., & Manrodt, K. 2013. *Vested outsourcing: five rules that will transform outsourcing*. Palgrave Macmillan.

Williman, A., & O'Donnell, C. 1970. "Through the Central 'Multi-Processor' Avionics Enters the Computer Era." *Aerona*

1.2.2 Task 2: Methods and Tools Piloting and Refinement

The USC depth-first exploration of Maintainability identified the need for a better balance of attention during the system acquisition phase between optimizing on system acquisition cost-effectiveness and optimizing on system life-cycle cost-effectiveness, particularly for software, due to the major differences between software and software logistical aspects. This led to the development of a proposed framework of Maintainability Readiness Levels for Software-Intensive Systems.

1.2.2.1 Maintainability Readiness Levels for Software-Intensive Systems

Abstract

The US Department of Defense (DoD) Better Buying Power series of memoranda clearly emphasizes that its objective is not to obtain minimum-cost, technically acceptable DoD capabilities, but to improve the life-cycle cost-effectiveness of DoD capabilities. A major opportunity for such improvement is to improve the cost-effectiveness involved in the life cycle maintenance of the system. One of the major projects of the DoD Systems Engineering Research Center focuses on System Qualities (SQs) Tradespace and Affordability, where Affordability is defined as life cycle cost-effectiveness. The project has found that Maintainability is a particularly important SQ, not only for Total Cost of Ownership (TCO), but also for Dependability, Changeability and Mission Effectiveness support. In researching Maintainability, we have found that significant progress has been made for physical systems maintainability such as via performance-based logistics and multisensor-based smart systems, but that less progress has been made for software and information-technology aspects of systems. This section elaborates on these findings. After a context-setting introduction, it summarizes the SERC SQ Ontology and discusses the key roles of Maintainability therein. It then proceeds to summarize the differences between hardware and software maintainability and the main sources of high software total cost of ownership; presents a proposed Software Maintainability Readiness Level (SMRL) framework to focus attention on the sources of added system-software maintenance cost; and summarizes the associated conclusions.

1. Introduction

A recent US General Accountability Office report[1] identified software and information technology (IT) maintenance as a major target for life cycle cost savings. It found that of the roughly $79 billion in planned US Government expenditures on IT over the next year, roughly $58 billion or about 75% of the expenditures were going into IT maintenance. Some

improvements could be made in making the IT maintenance organizations more efficient, but a recent systematic technical review[2,3] found that most of such expenditures have been due to technical debt: a term for delayed technical work or rework that is incurred when short-cuts are taken or short-term needs are given precedence over the long-term objectives. If the debt is not paid off, then it will keep on accumulating interest, making it hard to implement changes later on. Worldwide, a 2010 Gartner report[4] estimated that the Global "IT Debt" was over $500 billion, and had the potential to grow to $1 trillion by the end of 2015. Minkiewicz[5] provides a good summary of the relationship between technical debt and software maintenance costs, including such sources of cost as lack of documentation, lack of adherence to processes or standards, missing tests, lack of modularization, architecture or design that is not well thought out or scalable, and failure to keep up with technology.

The DoD Better Buying Power (BBP) memoranda, summarized in BBP 3.0[6], include such relevant recommendations as Continuing to set and enforce affordability constraints, including production and sustainment affordability caps; Strengthening and expanding "should cost" as an important tool for cost management; Increasing the use of performance-based logistics (PBL); Removing barriers to commercial technology utilization; and Using Modular Open Systems Architecture to stimulate innovation. The current SERC project addressing Systems Qualities (SQs) Tradespace and Affordability has been focusing on Maintainability as a key to reducing the high costs of software and IT maintenance indicated above. It has found that the use of PBL is making good progress for the physical aspects of systems such as for spare parts management[7], but less progress in the software and IT aspects due to differences in hardware and software logistics. In researching the root causes of software technical debt, an attractive prospect for addressing them is to expand the previous concepts of Technology Readiness Levels (TRLs) and System Readiness Levels (SRLs; focused primarily on system performance vs. maintainability)[8], to define and incorporate the use of Software Maintenance Readiness Levels (SMRLs).

Section 2 summarizes the current SERC product quality ontology, based on recent insights in sorting out the various roles of Maintainability with respect to the primary stakeholder-value SQ classes of Mission Effectiveness, Life Cycle Expenditures, Dependability, and Changeability. Using Maintainability as an example, it summarizes the major sources of variation in the system's Maintainability values in terms of the system's Referents, States, Processes, and inter-SQ Relations.

Section 3 summarizes the major maintainability differences between a hardware system and a software system, and identifies software maintainability as the main target of opportunity for reducing Total Cost of Ownership (TCO).

Section 4 identifies the main sources of high maintenance costs for software-intensive systems, and summarizes research into improved Maintainability methods, processes, and tools for software-intensive systems, including Opportunity Trees for improving Maintainability and relevant quantitative models for estimation of TCO.

Section 5 presents and discusses a proposed Software Maintainability Readiness Levels (SMRL) framework for assessing the extent to which a project is achieving its key enablers to be a

highly maintainable software-intensive system.

2. The Key Roles of Maintainability in the SERC SQ Ontology

Table 1 shows the revised upper levels of the initial SQ stakeholder value-based means-ends class hierarchy in our basic SQ ontology paper[9], brought about by our recent Maintainability research and formalization of the SQ relationships[10].

Table 1. Upper Levels of Revised Stakeholder Value-Based SQ Means-Ends Hierarchy

Stakeholder Value-Based SQ Ends	Contributing SQ Means
Mission Effectiveness	Stakeholders-satisfactory balance of Physical Capability, Cyber Capability, Human Usability, Speed, Endurability, Maneuverability, Accuracy, Impact, Scalability, Versatility, Interoperability, Domain-Specific Objectives
Life Cycle Efficiency	Development and Maintenance Cost, Duration, Key Personnel, Other Scarce Resources; Manufacturability, Sustainability
Dependability	Reliability, Maintainability, Availability, Survivability, Robustness, Graceful Degradation, Security, Safety
*C*hangeability	Maintainability, Modifiability, Repairability, Adaptability
Composite QAs	
Affordability	Mission Effectiveness, Life Cycle Efficiency
Resilience	Dependability, Changeability

Within the Department of Defense (DoD), the emphases on Mission Effectiveness and Life Cycle Efficiency are highlighted in the Secretary of Defense's May 2015 Budget Request[11], "We at the Pentagon must... be open to new ideas and new ways of doing business that can help us operate more efficiently and perform more effectively in an increasingly dynamic and competitive environment." The emphasis on Changeability, and implicitly Dependability, are also reflected in the statement, as well as in another DoD senior leadership statement[12], "Simply delivering what was initially required on cost and schedule can lead to failure in achieving our evolving national security mission — the reason defense acquisition exists in the first place." Similar concerns are emphasized by leaders and commentators in government, industry, and services sectors worldwide.

With respect to Table 1, we have changed Resource Consumption to Life Cycle Efficiency to reflect the emphases in DoD and elsewhere. We have also generalized Flexibility to Changeability to be more consistent with the MIT Changeability ontology[13] and its formal representation[14]. Its mutually-exclusive means are internal change (Adaptability) and external change (Maintainability), which includes Modifiability to support Changeability and Repairability to support Dependability, due to its role as Mean Time to Repair (MTTR) in relating Reliability as Mean Time between Failures (MTBF) to Availability via the relationship Availability = MTBF/(MTBF+MTTR). Thus, we find Maintainability to be key not only to Changeability and

Dependability, but also to Life Cycle Efficiency via Maintenance Cost, and indirectly to Mission Effectiveness via Life Cycle Efficiency.

In pursuing the various contributions and types of Maintainability as a critical but complex SQ, we have developed Figures 1 and 2 as elaborations of Maintainability relationships. Figure 1 focuses on the role of Maintainability in support of Changeability. It shows that Maintainability and Adaptability are two alternative means of achieving Changeability, external and internal to the system. It also shows that Modifiability and Repairability are two alternative means of achieving Maintainability, one for changes and one for defects. It also shows some enabling (and), but not alternative (or) subclasses of Modifiability, Repairability, and support of verification and validation, called "V&V-ity." These latter are Automated Analysis, Peer Reviews, and Execution Testing; their rating scales are provided in[9], where they were taken from the Constructive Quality Model (COQUALMO) for estimating delivered software defect density[15].

Figure 2 further elaborates on how Maintainability supports both Changeability and Dependability, and elaborates on how these two combine to support Resilience. It also elaborates the means-ends subclasses of Reliability, and of Testing in support of V&V-ity.

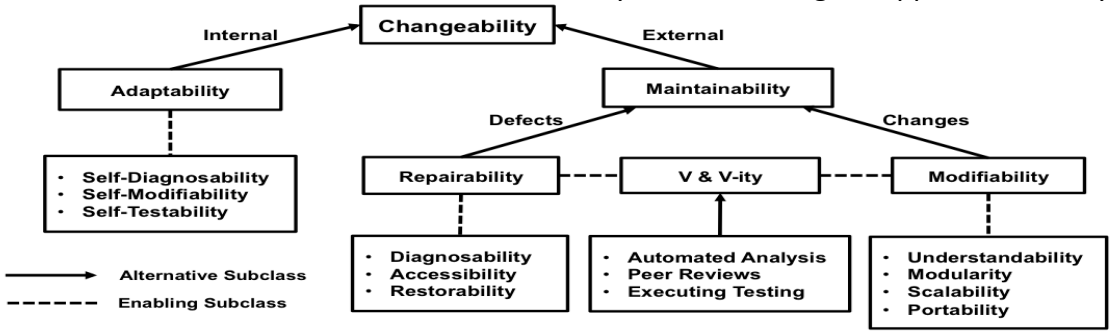


Figure 1. Role of Maintainability for Changeability

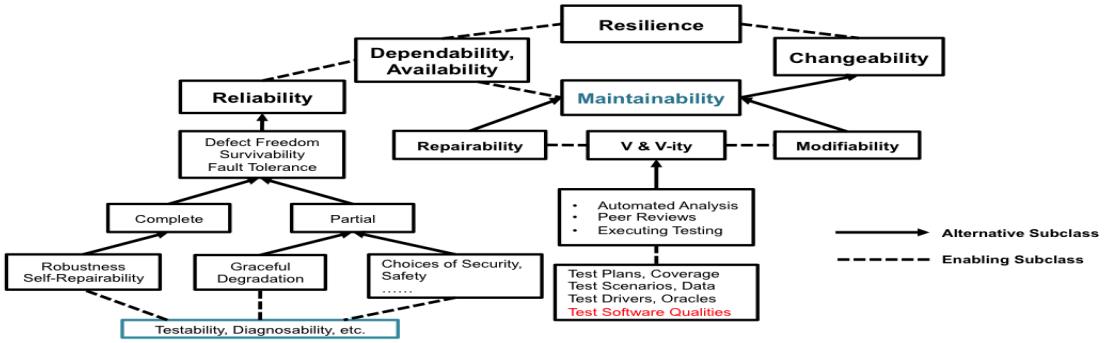


Figure 2. Role of Maintainability for Changeability, Dependability, and Resilience

Figure 2 also emphasizes the many-to-many relationships of Testability, as compared to the one-to-many relationship it has in ISO/IEC 25010 (ISO/IEC, 2011), where Testability only supports Maintainability. It is clear that Testability is also important to the achievement of numerous other SQs, such as the many aspects of Dependability in Figure 2. It also highlights the recursive complexities of the SQ hierarchy, as Testability also involves the test software, such as its classes (performance, stress, regression, etc.) and capabilities (test drivers, monitors, diagnosis aids, data managers, etc.) and the need to ensure its qualities as well, as seen in red in Figure 2.

Overall, Table 1 and Figures 1 and 2 summarize the class hierarchy of the primary stakeholder SQ end value classes of Mission Effectiveness, Life Cycle Efficiency, Dependability, and Changeability, along with their primary means-ends subclasses, and the primary composite SQs of Affordability and Resilience. Each of the SQ classes and subclasses are also characterized by the variation of their values with respect to their variation by Referents, States, Processes, and Relations, as summarized below and described in detail in the context of Maintainability in[10].

Using Maintainability and its metric Mean Time to Repair (MTTR) as an example, here are the major Ontology sources of variation in the system's Maintainability values. They will vary as a function of the system's Referents such as its stakeholders' priorities for higher or lower MTTR by criticality level. The MTTR will also vary by the system's States (internal, such as being in a live or dormant state; or external, such as needing field repair in the desert, jungle, or icefield); by the system's Processes (internal, such as undergoing a test or operational process; or external, such as undergoing a light vs. heavy workload), and by its Relations with other SQs, such as how a smartphone's MTTR will vary if the system is developed for ease of user battery replacement or developed for durability, where it is virtually impossible to open without extreme damage to its case[16].

3. Differences Between Hardware and Software Maintainability

The list below summarizes the major differences between hardware and software maintainability. Further differences and their discussions are in[17,18].

- Software components do not degrade due to wear and fatigue;
- No imperfections or variations are introduced in making copies of software components;
- Software results are unconstrained by any laws of physics. If a program expresses the force of gravity with the wrong sign, the calculations will produce the effect of anti-gravity;
- Software interfaces are conceptual rather than physical; there is no easy-to-visualize three-prong plug and its mate;
- There are many more distinct logic paths to check in software than in hardware;
- There are many more distinct entities to check. Any item in a large database may be a source of error.
- The failure modes are generally different. Software failures generally come with no advance warning, provide no period of graceful degredation, and more often provide no announcement of their occurrence;
- Repair of a hardware fault generally restores the system to its previous condition; repair of a software fault does not.

The major implication of these differences is that hardware and software maintenance strategies, processes, and costs are radically different. Making a hardware upgrade on a million hardware devices requires a million upgrade efforts. Making a software upgrade on a million instances of the software accessible via the Internet requires only one effort to perform the upgrade and electronically update the instances. However, the second-order effects of this

disparity are to make as many of the anticipated sources of change be implemented in software, making the volume of different software upgrades significantly higher. Further, the complexity and invisibility of the software makes the correct implementations of software upgrades much more labor-intensive and skill-intensive. Representative data[19,20] indicate that software upgrade costs are highly nonlinear with size, and that 47% of the maintenance effort involves understanding the software to be modified.

In terms of facilities, hardware maintenance involves a much larger logistics effort than does software maintenance. On the other hand, software maintenance generally requires more complex skill levels than does hardware maintenance. Facilities for hardware maintenance differ significantly from software maintenance facilities, although the prevalence of cyber-physical systems increases benefits from having a single facility addressing both hardware and software and both repairs and modifications. Recent trends in additive manufacturing or 3-D printing are simplifying hardware repair logistics (e.g., initiatives underway to put additive manufacturing capabilities on Navy ships) and moving hardware and software repair closer together. Another good example is the development of smart aircraft-maintainability systems, which populate the aircraft with thousands of nanosensors, which can monitor a component's degree of wear, structural integrity, or battery level, and provide the information in advance so that necessary repair capabilities are pre-positioned to effect rapid repair when the aircraft lands. This provides a much lower mean time to repair and much higher level of Availability, showing a strong synergy in increasing both Changeability and Dependability.

An excellent source on Repairability and Maintainability for hardware and to some extent software is the book Maintainability[21]. It identifies and provides guidance for the many sources of post-acquisition cost lying below the acquisition tip of the iceberg. These include operations cost, product distribution cost, maintenance cost, training cost, test and support equipment cost, software cost, technical data cost, supply support cost, and retirement and disposal cost. For individual cost components such as Repair, it identifies and provides guidance on the sequence of key activities involved in repair: detection, preparation for maintenance, localization and isolation, disassembly (access), faulty item repair or replacement, reassembly, alignment and adjustment, and condition verification.

In terms of opportunities for reducing system maintenance costs, it appears that recent results from hardware Performance-Based Logistics[7] and trends toward additive manufacturing and smart hardware-diagnostic systems indicate that reducing hardware maintenance costs have stronger existing opportunities. Also, more stable software maintenance organizations, particularly in maintaining embedded vehicle software, tend to have strong existing domain-oriented methods, processes, and tools. Greater software maintenance challenges appear to be in more dynamic, software-intensive application areas such as C4ISR, logistics, and information services. The next section will address potential counterpart opportunities for reducing software maintenance costs in the more diverse and dynamic, software-intensive application areas.

4. High Maintenance Costs for Software-Intensive Systems: Symptoms and Root Causes

Several classes of organizations generally do not have serious problems with high maintenance cost for their more diverse and dynamic software-intensive systems. Some have developers who continue with the project through its life cycle. Some whose business or mission depends

critically on high levels of service employ and support highly-capable in-house maintenance organizations.

Classes of organizations most needing to reduce high maintenance costs for their more diverse and dynamic software-intensive systems are those in which Research and Development (R&D) and for Operations and Maintenance (O&M) are separately funded and managed, organizations which outsource software maintenance to external companies, and organizations with in-house software maintenance centers that receive and maintain software developed either elsewhere in the organization or elsewhere. For such organizations, three of the primary symptoms of high maintenance costs are (1) unprepared maintainers; (2) underestimated workload; and (3) hard-to-modify software.

(1) Unprepared Maintainers. Frequently, the maintainers are belatedly notified that the software will be theirs to maintain. They will have been busy on other projects, with minimal opportunity to understand or influence the system's development. They will have minimal budget to prepare before the system enters maintenance. As a result, their current domain expertise, skills, and support environment may be a major mismatch to the maintenance needs of the new system.

(2) Underestimated Workload. Often, the software will have numerous hasty patches that were needed to pass its acceptance tests, a good many of which will have undesirable side effects. There is likely to be a large backlog of change requests and changed interfaces with interoperating co-dependent external systems. The development contract may have neglected to specify that the system be delivered with the latest release of the supporting COTS products. In gathering data for the COCOTS cost model, we encountered one project with 120 COTS products, 55 of which were using versions that had become unsupported by the vendor.

(3) Hard-to-Modify Software. The acquirers and developers make ambitious promises with respect to budgets, schedules, and deliverables, often on a fixed-price contract. This generally leads to a brittle, point-solution architecture that is minimally documented and expensive to modify.

Once the system enters maintenance, the best the maintainers can do is to address the symptoms. However, much more leverage on the Total Cost of Ownership is available by attacking the root causes of the symptoms during acquisition. Three of the primary root causes are (1) Separate funding sources, management foci, and the conspiracy of optimism; (2) Inadequate participation of maintainers in the development process; and (3) Inadequate preparation of maintenance methods, processes, and tools (MPTs).

(1) Separate Funding Sources, Management Foci, and the Conspiracy of Optimism.

Many organizations have separate funding sources for R&D and for O&M. This tends to cause sponsors and performers supported on R&D funds to focus on R&D performance, and for sponsors and performers supported on O&M funds to focus on O&M performance. The R&D sponsors compete for finite R&D funds, causing them to be optimistic both on the capabilities to be developed and on the funds needed to develop them. Accordingly, when their project is optimistically funded and they are running the competition to select the best supplier, they will encourage innovations, but frequently use a "lowest cost, technically acceptable" criterion for source selection.

Naturally, the bidders do not want to lose the competition by bidding too high. They will

therefore apply further optimism in what they promise to deliver and the productivity that will minimize the cost. In general, they will propose a point-solution architecture to provide a minimal interpretation of the requirements in the request for proposal. Frequently, though, even with these strategies, they will find that reality will not match their optimistic assumptions of getting the A team for development, for technology readiness, for non-developmental items (NDIs) to perform as advertised, for the team to stay stable, etc. This will lead to shortfalls in funding the capabilities to be developed, and with limited funds, the priorities will generally favor mission capabilities over maintenance capabilities such as diagnostics, version control, testing support, and documentation.

If funding shortfalls require across-the-board cuts, the first project organization to be impacted will be Systems Engineering. The usual result of this will be that they will have insufficient time time to specify the rainy-day use cases, to analyse alternative NDI candidates, to address risk, and to define complex interface protocols. Such systems engineering shortfalls have been found to be major sources of rework, technical debt, and software maintenance costs[5]. Calibration of the COCOMO II cost model found that added cost of rework and technical debt of minimal vs. necessary systems engineering is an exponential function of software size; an added 14% of project cost for a 10 KSLOC (thousands of source lines of code); 38% for 100 KSLOC, 63% of 1000 KSLOC, and 92% for 10,000 KSLOC[22]. A good example was a major project with 5000 KSLOC and a Conspiracy of Optimism budget of $2 billion. The project thought that it was doing well to allocate 30% of the budget ($600 million) for systems engineering, but the project's actual cost was $8 billion, meaning that the allocation to systems engineering had been only 7.5%. And here also, the necessary budget for maintenance capabilities such as diagnostics, version control, testing support, and documentation was severely curtailed.

(2) Inadequate Participation of Maintainers in the Development Process.

Although initiatives such as Acquisition Logistics often work well for hardware-intensive systems, they tend to work less well for software-intensive systems due to the differences in hardware and software maintenance discussed above. The most severe root cause of inadequate participation of maintainers in the development process is the vicious circle caused by previous inadequate participation. On several large projects, requests for maintainer participation in defining a new system have been met by regrets that the maintainers are too busy keeping up with the results of inadequate participation of maintainers in previous development processes.

Other root causes include the overconcern with the "voice of the customer" in such popular methods as Quality Function Deployment (QFD)[23]. Its use most often focuses on delighting customers and users, and on cost-effective development, while neglecting the voice of the maintainer. Similar overconcern with development often occurs in organizations supported by R&D funds. Related root causes are systems optimized for other qualities that degrade maintainability. For example, tightly-coupled High Performance Computing architectures and software improve computational Speed but reduce Maintainability. Easiest-first agile methods often improve initial Usability, but make architectural commitments that often subsequently degrade Scalability, Safety, and/or Security. Committing to use strong but rapidly-evolving and inscrutable components can improve initial mission effectiveness but pose threats to maintenance timeliness and dependability.

A further root cause is the inverse of the well-known Peter Principle[24], which says that "In a

hierarchy, every employee tends to rise to his level of incompetence." For software maintenance people, the Inverse Peter Principle tends to apply: "People rise to a position in which they become indispensible, and get stuck there forever." Besides having terminal career paths, they are often impacted by the Paul Principle[25], "People rise to an organizational position in which their technical skills become obsolete in five years." This is often compounded by misguided attempts to conserve on maintenance costs by limiting expenditures on training, continuing education, conference attendance, and professional society memberships.

Besides increasing the level of participation of maintainers in the development process, the next-highest leverage will come from increasing the competence levels of the maintainers. An excellent detailed roadmap for doing this is the CMU-SEI People Capability Maturity Model[26]. As with the CMMs for systems and software engineering, it has a 5-level progression including such practise as staffing, work environment, training and development, compensation, workforce planning, career development, empowered workgroups, continuous capability improvement, and continuous workforce innovation.

(3) Inadequate Preparation of Maintenance Methods, Processes, and Tools (MPTs).

Several of the root causes of inadequate preparation of maintenance MPTs have been addressed above, such as the primary emphasis on acquisition MPTs and concepts of operation vs. their maintenance counterparts; shortfalls in preparing for maintenance facilities and infrastructure; optimizing on other qualities such as high performance and easiest-first agility but which complicate maintenance; and the effect of development overruns on underdelivery of maintenance enablers such as the system's architecture, traceability information, change management capabilities, problem diagnostic capabilities, and test support capabilities. Other root causes include use of obsolete maintenance MPTs, lack of coordination of the MPT's used in developing system components, and development commitment to MPTs that are unscalable to the Full Operational Capability.

5. A Proposed Software Maintainability Readiness Levels (SMRL) Framework

The concepts of Technology Readiness Levels (TRLs)[27], Manufacturing Readiness Levels (MRLs)[28], and System Readiness Levels (SRLs)[29,30] have been highly useful in improving the readiness of systems to be fielded and operated. But except for one Systems Readiness Level table[30] indicating that for Operations and Support, State of the Art systems have high training cost and lack of support; State of the Practice systems have training and support readily available, and State of the Obsolescence systems have high cost of maintenance and increased training cost, the current SRL content does not address system maintainability readiness. Given the discussions above on the high costs, difficulty symptoms, and difficulty root causes of many software-intensive systems' maintainability problems, it appears worthwhile to develop and use a similar Software Maintainability Readiness Level (SMRL) framework to improve future systems' continuing operational readiness and Total Cost of Ownership (TCO). Most likely, its content could also help on hardware-intensive systems or cyber-physical-human systems.

Table 2 on the next page provides a proposed SMRL framework. Its columns are organized around the three root causes of high software-intensive system maintenance costs: the maintenance Operational Concept and management aspects; Personnel capabilities and

participation aspects, and Enabling MPT capabilities. In general one would expect a major defense acquisition project to be at SMRL 4 at its Materiel Development Decision milestone; at SMRL 5 at its Milestone A; SMRL 6 at its Milestone B; and SMRL 7 at its completion of Operational Test and Evaluation. Smaller systems would be expected to be at least at SMRL 3 at its Materiel Development Decision milestone and at SMRL 4 at its Milestone A. Note that it emphasizes outcome-based maintenance incentives such as with Performance-Based Logistics[6,7] or Vested Outsourcing[31] at SMRL 7, and maintainability data collection and analysis (DC&A) at SMRL 8. Some good examples of DC&A-based improvements in reducing technical debt and TCO via modularization around sources of change and identification of product line commonalities and variabilities are provided in references[32,33,34]. Some other good sources of insight in reducing software maintenance costs are[35,36]. We are also conducting empirical studies of software maintainability metrics via analyses of open-source software systems[37].

6. Conclusions

Major and growing sources of savings in a system's TCO can be achieved via improvements in software maintainability. Some savings can be achieved by improving software maintenance via better MPTs, but much greater savings can be achieved by addressing the root causes of increased maintenance costs due to overemphasis on initial acquisition cost-effectiveness. These include the maintenance Operational Concept and management aspects; Personnel capabilities and participation aspects, and Enabling MPT capabilities. Drawing on the successful use of the Technology Readiness Level, Manufacturing Readiness Level, and System Readiness Level frameworks, this paper proposes a similar Software Maintenance Readiness Level framework, based primarily on cumulative improvement of the three acquisition root causes of increased maintenance costs. Several organizations are interested in experimentally applying it.

Software-Intensive Systems Maintainability Readiness Levels			
SMR Level	**OpCon, Contracting: Missions, Scenarios, Resources, Incentives**	**Personnel Capabilities and Participation**	**Enabling Methods, Processes, and Tools (MPTs)**
9	5 years of successful maintenance operations, including outcome-based incentives, adaptation to new technologies, missions, and stakeholders	In addition, creating incentives for continuing effective maintainability. performance on long-duration projects	Evidence of improvements in innovative O&M MPTs based on ongoing O&M experience

8	One year of successful maintenance operations, including outcome-based incentives, refinements of OpCon. Initial insights from maintenance data collection and analysis (DC&A)	Stimulating and applying People CMM Level 5 maintainability practices in continuous improvement and innovation in, e.g., smart systems, use of multicore processors, and 3-D printing	Evidence of MPT improvements based on maintenance DC&A-based ongoing refinement, and extensions of ongoing evaluation, initial O&M MPTs.
7	System passes Maintainability Readiness Review with evidence of viable OpCon, Contracting, Logistics, Resources, Incentives, personnel capabilities, enabling MPTs, outcome-based incentives	Achieving advanced People CMM Level 4 maintainability capabilities such as empowered work groups, mentoring, quantitative performance management and competency-based assets	Advanced, integrated, tested, and exercised full-LC MBS&SE MPTs and Maintainability-other-SQ tradespace analysis
6	Mostly-elaborated maintainability OpCon, with roles, responsibilities, workflows, logistics management plans with budgets, schedules, resources, staffing, infrastructure and enabling MPT choices, V&V and review procedures.	Achieving basic People CMM levels 2 and 3 maintainability practices such as maintainability work environment, competency and career development, and performance management especially in such key areas such as V&V, identification & reduction of technical debt.	Advanced, integrated, tested full-LC Model-Based Software & Systems (MBS&SE) MPTs and Maintainability-other-SQ tradespace analysis tools identified for use, and being individually used and integrated.
5	Convergence, involvement of main maintainability success-critical stakeholders. Some maintainability use cases defined. Rough maintainability OpCon, other SCSHs, staffing, resource estimates. Preparation for NDI and outsource	In addition, independent maintainability experts participate in project evidence-based decision reviews, identify potential maintainability conflicts with other SQs	Advanced full-lifecycle (full-LC) O&M MPTs and SW/SE MPTs identified for use. Basic MPTs for tradespace analysis among maintainability & other SQs, including TCO being used.

	selections.		
4	Artifacts focused on missions. Primary maintenance options determined, Early involvement of maintainability SCSHs in elaborating and evaluating maintenance options.	Critical mass of maintainability SysEs with mission SysE capability, coverage of full M-SysE skills areas, representation of maintainability success-critical-stakeholder organizations.	Advanced O&M MPT capabilities identified for use: Model-Based SW/SE, TCO analysis support. Basic O&M MPT capabilities for modification, repair and V&V: some initial use.
3	Elaboration of mission Operational Concept (OpCon), Architectural views, lifecycle cost estimation. Key mission, O&M, success-critical stakeholders (SCSHs) identified, some maintainability options explored.	O&M success-critical stakeholders provide critical mass of maintainability-capable SysEs Identification of additional. M-critical success-critical stakeholders.	Basic O&M MPT capabilities identified for use, particularly for OpCon, Arch, and Total cost of ownership (TCO) analysis: some initial use.
2	Mission evolution directions and maintainability implications explored. Some mission use cases defined, some O&M options explored.	Highly maintainability-capable Systems Engineers (SysEs) included in Early SysE team.	Initial exploration of O&M MPT options
1	Focus on mission opportunities, needs. Maintainability not yet considered.	Awareness of needs for early expertise for maintainability. concurrent engr'g, O&M integration, Life Cycle cost estimation	Focus on O&M MPT options considered

Table 2. Software-Intensive Systems Maintainability Readiness Levels

References

1. U.S. General Accountability Office 2015. *2015 Annual report; Additional Opportunities to Reduce Fragmentation, Overlap and Duplication and achieve Other Financial Benefits,* GAO-15-404SP, April 14.
2. Huang L. 2015. Detecting and Evaluating Technical Debt of Software Systems, in Wade J. et al., RT 128: New Project Incubator, *SERC-2015-TR-128.1*, Stevens Institute, July 31.

3. Zhang Q., Huang L., Jan N., Lane J. A., and Zhang H. 2015. Detecting and evaluating technical debt in software systems: A systematic literature review. *Journal of Systems and Software*. Submitted May 31, 2015, under review.
4. Gartner 2010. *Gartner Estimates Global 'IT Debt' to Be $500 Billion This Year, with Potential to Grow to $1 Trillion by 2015*. http://www.gartner.com/newsroom/id/1439513.
5. Minkiewicz A. 2015. At the Intersection of Technical Debt and Software Maintenance Costs, *Proceedings, ICEAA Professional Development & Training Workshop,* June.
6. Kendall F. 2014. *Better Buying Power 3.0,* OUSD(AT&L), 19 September.
7. Insinna V. 2015. "Market for Performance-Based Logistics Grows." *National Defense*, March.
8. Sauser B, Verma D, Ramirez Marquez J, and Gove R. 2006. From TRL to SRL: the Concept of Systems Readiness Levels, *Proceedings, CSER*, April.
9. Boehm B., & Kukreja N. 2015. "An Initial Ontology for System Quality Attributes." *Proceedings, INCOSE International Symposium,* July.
10. Boehm B, Chen C, Shi L, & Srisopha K. 2015. The Key Roles of Maintainability in an Ontology for System Qualities, submitted to INCOSE IS 2016.
11. Carter A (SecDef). 2015. Submitted Statement, Senate Appropriations Committee-Defense (Budget Request) May 6.
12. Kendall F. 2015. *2015 Performance of the Defense Acquisition System,* USD(AT&L) Report.
13. Ross A. & Rhodes D. 2015. "Towards a Prescriptive Semantic Basis for Change-Type Ilities." *In Proceedings. CSER.*
14. Dou K., Wang X., Tang C., Ross A., & Sullivan K. 2015. "An Evolutionary Theory-Systems Approach to a Science of the ilities." *In Proceedings, CSER.*
15. Boehm B, Abts C, Brown AW, Chulani S, Clark B, Horowitz E, Madachy R, Reifer D, & Steece B. 2000. *Software Cost Estimation with COCOMO II.* Prentice Hall.
16. iFixit Corp, Smartphone Repairability, https://www.ifixit.com/smartphone-repairability
17. Boehm B. 1975. Guest Editorial, Special Issue on Reliable Software, *IEEE Trans. SW Engr.,* June, 135-137.
18. Boehm B. 1994. Integrating Software Engineering and System Engineering, *J.NCOSE 1.1,* July-September.
19. Selby R. 1988. Empirically Analyzing Software Reuse in a Production Environment, in Tracz W. (ed.), *Software Reuse: Emerging Technology*, IEEE-CS Press, 176-189.
20. Parikh G. & Zvegintzov N. 1983. The World of Software Maintenance, *Tutorial on Software Maintenance,* IEEE-CS Press, 1-3.
21. Blanchard B, Verma D, & Peterson E. 1995. *Maintainability: a key to effective serviceability and maintenance management*. John Wiley and Sons.
22. Boehm B., Valerdi R. & Honour E. 2008. The ROI of Systems Engineering: Some Quantitative Results for Software-Intensive Systems, *Systems Engineering*, 11, 3, April, 221-234.
23. Akao Y. 1994. "Development History of Quality Function Deployment". *The Customer Driven Approach to Quality Planning and Deployment*. Minato, Tokyo 107 Japan: Asian Productivity Organization. p. 339. ISBN 92-833-1121-3.
24. Peter L. and Hull R. 1969. *The Peter Principle,* William Morrow.
25. Armer P. 1966. *Computer Aspects of Technical Change, Automation, and Economic Progress,* Rand Corp., P-3478, November.
26. Curtis B., Hefley W., & Miller S. 2002. *The People Capability Maturity Model,* Addison

Wesley.
27. Meystel A., Albus J., Messina E., & Leedom D. 2003. Performance Measures for Intelligent Systems: Measures of Technology Readiness, *PERMIS '03* White Paper.
28. Cundiff D. 2003. Manufacturing Readiness Levels (MRL), Unpublished white paper.
29. Sauser B., Verma D., Ramirez-Marquez J., & Gove R. 2006. From TRL to SRL: The Concept of Systems Readiness Levels, *Conference on Systems Engineering Research*
30. Sauser B. 2007. *System Maturity Metrics for Decision Support in Defense Acquisition Users Guide: Version 1.0*, US Army ARDEC Report, November.
31. Vitasek K., Ledyard M., & Manrodt, K. 2013. *Vested outsourcing: five rules that will transform outsourcing*. Palgrave Macmillan.
32. Boehm B, Lane J, and Madachy R. 2011, Total Ownership Cost Models for Valuing System Flexibility, *Proceedings, CSER*, April.
33. Poulin J. 1996. *Measuring Software Reuse.* Addison Wesley.
34. Jacobson I., Griss M., & Jonsson P. 1997. *Software Reuse.* Wiley.
35. Gilb T. 2008. Designing Maintainability in Software Engineering: a Quantified Approach**,** *Result Planning Ltd.*
36. Reifer D. 2012. *Software Maintenance Success Recipes*, CRC Press.
37. Chen C, Shi L., & Srisopha K. 2015. "How does software maintainability vary by domain and programming language?" In *Proceedings, IEEE SW Tech Conference.*

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

Objectives:

One long-term objective of Task 3 has been to develop COSYSMO 3.0, an up-to-date model for estimating systems engineering costs. For Phase 4, these were the objectives for COSYSMO 3.0:

- Develop a complete COSYSMO 3.0 model with these properties:
  - Integrates the features of the major previous models (basic cost estimation, development with reuse, requirements volatility, development for reuse);
  - Has been calibrated via expert opinion;
  - Models the interoperability property (applicable to systems-of-systems);
  - Has reorganized and improved definitions of cost drivers involving "understanding";
  - Has a plausible body of validation projects (completed projects with actual data available to the researchers); and
  - Is "polished", in the sense of having minor changes that improve predictive capability.
- Initial development of a "Rosetta Stone" that allowed estimation parameter values using earlier versions of COSYSMO to be translated to COSYSMO 3.0 parameter values.
- Initial development of a questionnaire for gathering information for model validation.

Tasks:

- Conduct weekly or biweekly teleconferences coordinating model development with a Working Group of advisors from industry, government, and academia.
- Conduct workshops where the Wideband Delphi technique is used to gather expert opinion about numeric parameters of the model.
- Conduct other workshops as needed.

Results:

- Under Task 3, we had developed an initial model that included basic cost estimating; that served as our baseline for Task 4 development.
- We developed a version of the model that used the Bloom taxonomy to completely revise the "understanding" cost drivers. This seemed to improve the definitions of these cost drivers. However, it was realized that validation data from previous versions of the model could not in general be carried forward to COSYSMO 3.0 validation; *i.e.*, our objective to improve these definitions was in conflict with our objective to have a plausible body of validation projects. We therefore abandoned our effort to improve the "understanding" definitions, and constrained our remaining changes to allow reuse of validation data from previous models, which constitutes our plausible body of data.
- We conducted sessions at these workshops, addressing the indicated topics:
    - INCOSE International Workshop (Torrance CA, January 26): Critique of the improved-understanding-cost-drivers model, plus a Delphi.
    - Ground Systems Architecture Workshop, conducted by The Aerospace Corporation for SMC (Los Angeles, March 4): Delphi, plus a pilot of the existing model.
    - SERC Technical Research Review at USC (April 2): Presentation of existing model.
    - Annual Research Review at USC (Los Angeles, April 15): Presentation and detailed discussion of a model that reverted from the improved-understanding cost drivers back to older definitions. This model added CONOPS Understanding, Interoperability, and Process Capability cost drivers; also a multi-subsystem model (applicable to systems-of-systems). Several minor issues were raised, which were resolved during the year.
    - Technical Interchange Meeting with The Aerospace Corporation (Los Angeles, April 20): General discussion of the model.
    - INCOSE 2015 (Seattle, July 13): Workshop; discussion of model with Requirements Volatility added.
    - Software and IT Cost IPT Meeting (Washington DC, August 11): Delphi.
    - NDIA Systems Engineering Conference (Washington DC, October 29): Presentation of model with development-for-reuse.
    - COCOMO/Systems and Software Cost Modeling Forum (Washington DC, November 17): Discussion of model with development-for-reuse added (the complete model); Delphi; distribution of initial Rosetta Stone and questionnaire.
- We conducted weekly or biweekly Working Group meetings, where many minor

changes were raised, discussed, and resolved.

Technology Piloting:

- As noted above, we piloted a COSYSMO 3.0 model at the Ground Systems Architecture Workshop.

1.3 Future Plans for Phase 5

1.3.1 Task 1, Foundations

USC will extend its Phase 4 depth-first approach to Maintainability to cover the other components of the SQ Ontology, and will collaborate with MIT and U.Virginia in coordinating the USC and MIT ontologies and in formalizing the ontologies (also underway in the complementary U.Virginia-USC NSF grant). The table below provides a mapping between the current USC ontology's definitions (Labels) and those of MIT as shown in Figure 3 of the MIT report, as a first step toward reconciling the two ontologies.

USC-MIT Semantic Mappings

USC Impetus				USC Outcome				
Perturbation	Agent	Nature	Aspect	Effect	Aspect		USC, MIT Labels	MIT Differences
Disturbance	Internal	Increase	Form	Same	Form			
Defect	External	Decrease	Scope	Not-Same	Scope			
Opportunity	any	Re-host	any	Reduced	any			
any		any		Increased				
				any				
Disturbance	any	Increase	any	same, Increased	any		Resilience	Not included
any	any	any	any	not-same	any		Changeability	Same
Disturbance	Internal	any	any	some	any		Robustness	Shift vs. Disturbance
Disturb, Opp'y	Internal	any	any	not-same	any		Adaptability	Same
Defect	Internal	any	any	same	any		Fault-Tolerance	Not included
Defect	Internal	any	any	not-same	any		Self-repairability	Not included
Disturb, Opp'y	External	any	any	not-same	any		Modifiability	Just External (Internal = Adaptability)
Defect	External	any	any	any	any		Repairability	Not included
any	External	any	any	any	any		Maintainability	Not included
any	Internal	any	any	Reduced	any		Graceful Degredation	not included
any	any	any	Form	not-same	Form		Reconfigurability	Similar
Disturb, Opp'y	any	Increase	Scope	not-same	Scope		Extensibility	Same - Also Scalability-Up
Disturb,	any	Decrease	Scope	not-same	Scope		Contractability	Not included - Also Scalability-

Opp'y								Down
Disturb, Opp'y	any	Re-host	Form	Same	Form		Portability	Not included
Disturb, Opp'y	any	any	Scope	Same	Scope		Versatility	Similar
							Survivability	Robustness + Adaptability?
							Evolvability	Same as Changeability?
							Flexibility	External Adaptability?
							Agility	Focus on fix speed
							Reactivity	Focus on duration of fix needs
							Exchangeability	Like Modifiability?
							Variations by	Value, Form, Function, Operation

1.3.2 Task 2: Methods and Tools Piloting and Refinement

USC proposes to coordinate the Maintainability Readiness Levels for Software-Intensive Systems framework with interested organizations. The Aerospace Corp. is investigating its use, and we are proposing to present it as a keynote address at the Army Practical Systems and Software Measurement Users Group Workshop in February, and to present it as a paper at CSER 2016 in March.

1.3.3 **Task 3. Next-Generation Cost Estimation Models**

USC, NPS. Full definition and early data collection for COSYSMO 3.0 systems engineering cost model and major-system version of COCOMO III software cost model, including integration of reuse, requirements volatility, systems-of-systems interoperability, agile methods, and model-based systems and software engineering. Redefinition of rating scales for distributed development, tool support, sizing, and personnel experience

USC, NPS, GTRI. Extensions of previous GTRI-USC integration of SysML parametrics with COSYSMO for cost-effectiveness analyses.

Technology Piloting. The integration of SysML parametrics with COSYSMO serves as a pilot of the COSYSMO 3.0 model.

Transition. The COSYSMO 3.0 Working Group includes, among others, experienced industrial users of previous COSYSMO models and representatives of tool vendors. Our plans are to provide a document defining the COSYSMO 3.0 model to them; thereafter we plan to work with them to get the model into industrial use through both paths.

2. Massachusetts Institute of Technology (MIT)

2.1 Summary of Phases 1 and 2 Results

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 1.

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."

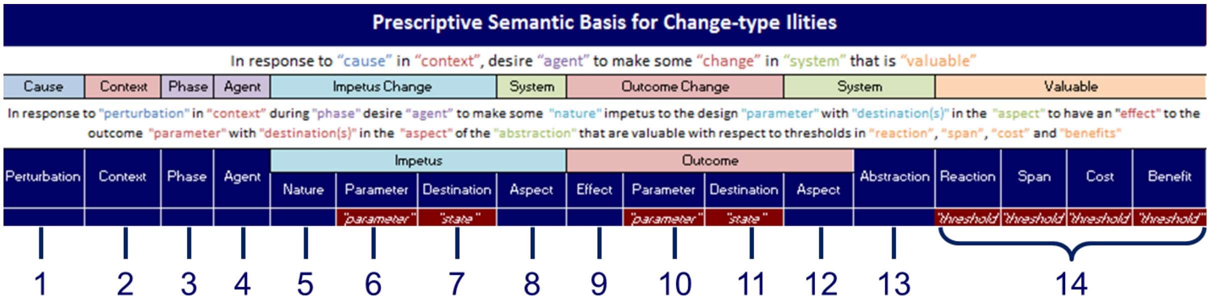


Figure 1. Change-type prescriptive semantic basis in 14 categories. (Ross, Beesemyer, Rhodes 2011)

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"

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 Change					Mech	Outcome Change					System	Valuable* (this category is not complete)			
				Impetus* (optional)					Mech	Outcome									
Perturbation	Context	Phase	Agent	Nature	Parameter	Origin	Destination	Aspect	Mechanism	Effect	Parameter	Origin	Destination	Aspect	Abstraction	Reaction	Span	Cost	Benefit
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 2. 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 Phases 1 and 2. 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 3).

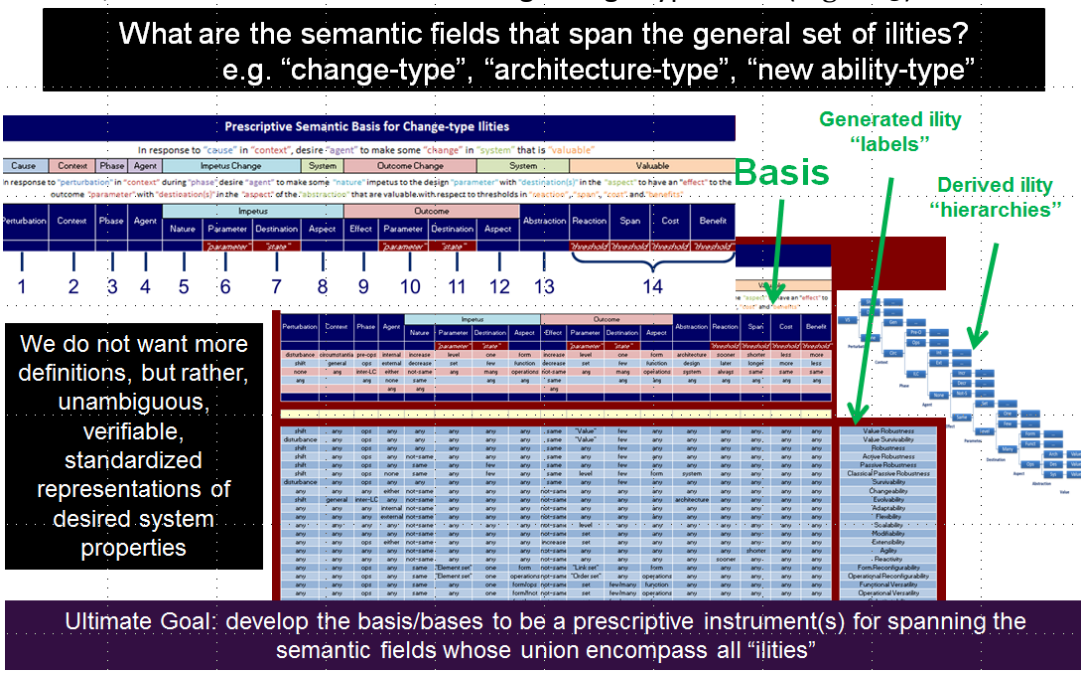


Figure 3. Basis began with 14 categories.

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. 4. 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)				

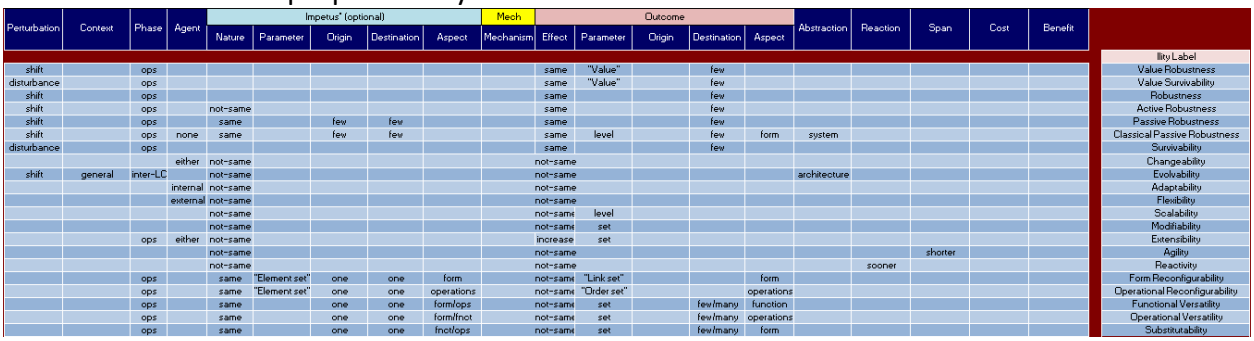
In response to "perturbation" in "context" during "phase" desire "agent" to make some "nature" impetus to the system "parameter" from "origin(s)" to "destination(s)" in the "aspect" using "mechanism" in order to have an "effect" to the outcome "parameter" from "origin(s)" to "destination(s)" in the "aspect" 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 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. 4. 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 5. 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 5. 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 6. 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.

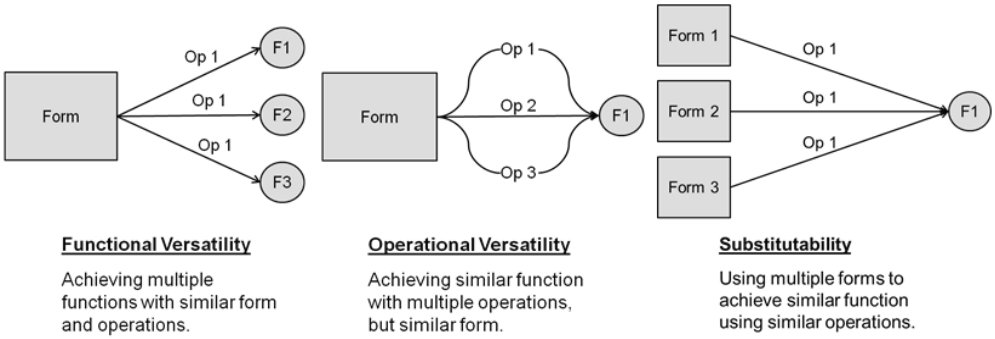


Figure 6 (a) Functional versatility; (b) operational versatility; (c) substitutability as suggested by the semantic basis

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

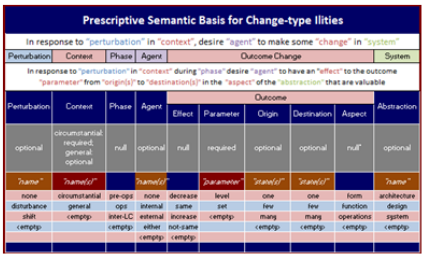
In response to "loud noises" in "night", during "ops" desire "owner" to be able to "increase" the "level of volume" of the "his stereo" in "less than one second"

{10 columns}

Scalable

Flexible

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



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

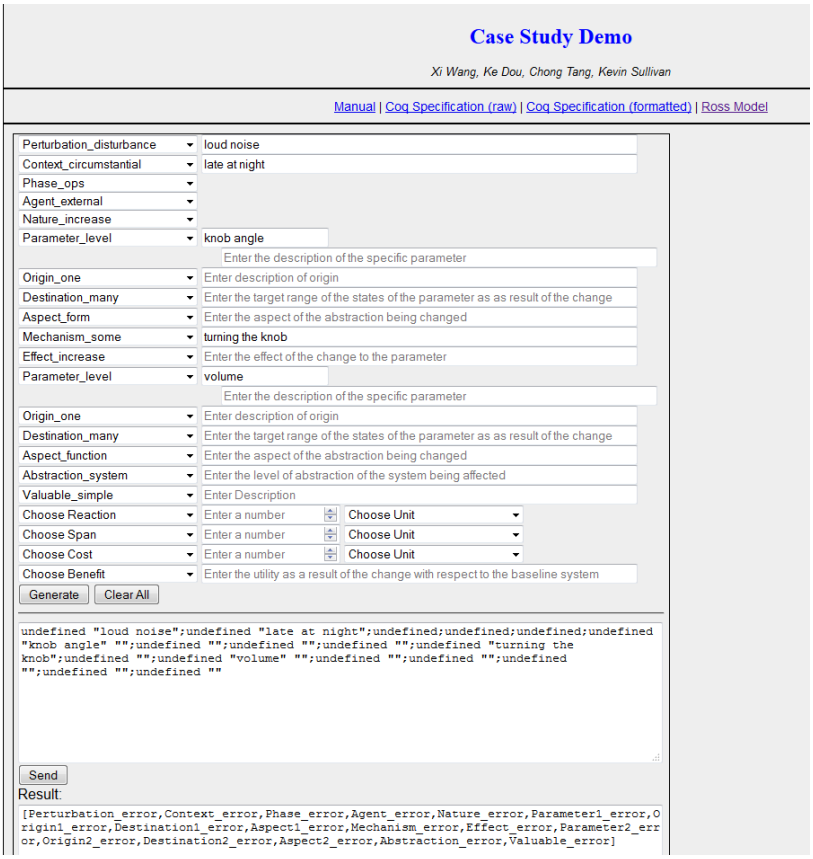
In response to "improved IEDs and ambush tactics" in "urban warfare", during "pre-ops" desire "AM General" to make an "increase" to the "armor" from "baseline" to "above baseline" in the "form" of the "HMMWV design" that is valuable

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.

References

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

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

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

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

Ross, A.M., Beesemyer, 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.

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 PHASES 4 AND 5 PLANS

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 (UVa)

3.1 Past Results

In Phases 1, 2 and 3, UVa explored alternative methods of formalizing the SQs and their relationships, including development and exercise of prototypes. This work converged on the Coq formalization of entities and relationships, and its application in analyzing the consistency of the MIT SQ semantic basis, leading to several improvements in the semantic basis and the submission of an accepted paper of CSER 2015.

3.2 Phase 4 Results

UVa presented the CSER 2015 paper, and participated in several mini-workshops with USC and MIT at CSER 2015, INCOSE IS 2015, and a 3-day working session at USC to apply the Coq formalization to the USC stakeholder value-based, means-ends ontology framework. The result was a successful formalization of the Dependability means-ends framework, enabling analysis of the degree to which the evidence of the Dependability means (Reliability, Availability, Maintainability, Safety, Security, etc.) assured the evidence of the Dependability ends. UVa continued to elaborate the Coq formalization to the rest of the USC stakeholder value-based, means-ends ontology framework.

Based on these results, UVa and USC obtained a joint NSF grant to extend the formalization and use it to develop and test an underlying theory enabling consistency analysis of combinations of SQs.

3.3 Phase 5 Plans

UVa will continue to elaborate the theory, and will develop and test prototype tools for performing consistency analysis of SQ combinations, in collaboration with USC and MIT.

Reference

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.

4 Wayne State University (WSU)

4.1 Introduction

WSU's research is focused on developing affordability and tradespace MPT that are tightly linked to the activities and decisions made during system acquisition for DoD Programs of Record. Our focus has been on the need to develop and acquire cost-effective systems that are operationally suitable and effective when fielded, despite long development lead times, and can be cost-effectively upgraded and extended during extended service lives. Within the DoD system acquisition process, these challenges are addressed through the acquisition strategy and system requirements. Our goal in this project is to develop, test, demonstrate and transition SE Methods, Procedures and Tools to acquisition commands to solve these challenges.

Our approach is to work with TARDEC/RDECOM and NAVSEA acquisition commands to identify needs and challenges, to identify the solution features and capabilities, then to conduct applied research and development towards the MPT. Our research approach has been to engage exemplar end-users, TARDEC/RDECOM and NAVSEA, iteratively throughout the process.

The exemplar DoD customer elements are engaged in all aspects of SE in DoD system acquisition. This helps ensure that the methods, procedures and tools we develop are generally applicable to other DoD elements engaged in concept development, requirements specification and tradeoffs during engineering and manufacturing development that share the SE functions and systems acquisition lifecycle model.

Our approach is to develop, demonstrate, and transition the MPT working with our early-adopter partners. Specific example realizations and applications to demonstrate relevance and suitability are in the context of the ground vehicle and surface ship domains. It is necessary to focus on a specific domain in order to be able to get to the level of detail and completeness needed to demonstrate that the MPT are practical and relevant in the context of real DoD acquisition Programs of Record.

Past

Prior to FY2015, we primarily conducted research on needs and challenges, solution features and capabilities, support tools, and the vision for MPT to address the needs and challenges. This framed our 2015 research. Key system acquisition tradespace and affordability needs and challenges came back to the fundamental problem of changing operational conditions and capability needs, and budget priorities, in a world whose changes cycle faster than the DoD system acquisition life cycle, and that changes dramatically over the service life of long-lived DoD systems.

Present

In 2015 we identified solution opportunities in DoD acquisition methods and strategies, and solution opportunities requiring implementation by concept and engineering developers. This is a significant difference.

DoD acquisitions can be structured in such a way that provide the Government with greater range of future down-selection options to choose from, and that provide greater modularity and adaptability without significant increase in development cost through economic motivation

(not incentives, but motivation) for developers. This method is applies the principles of Set-Based Design (SBD) to formulate an acquisition strategy.
We also develop rigorous analytical frameworks for SBD as a design-decision method for developers. One formulation addressed the problem of protracted acquisition. The other formulation addressed the need for adaptability and extensibility in long-lived systems.

Future
In FY 2016 we plan to socialize the proposed methods for set-based acquisition, applying the principles of SBD to the acquisition strategy and method, with our acquisition command transition partners. Our goals are to clarify and refine the approach, and initiate a trial application. The trial application will demonstrate transition of the methods to a DoD user. Further promulgation to the greater acquisition community will proceed from there, referencing the application results.
In FY 2016 we plan to complete the formal framework and MPT applying SBD to the two challenge areas, initiate "bench-level" tools, and initiate demonstration applications using a recent or current Program of Record. We anticipate completing the demonstration applications in 2017, working with one of the transition partners. These methods involve complex algorithms, and implementation as tools for complex systems is non-trivial. Obtaining design-level data is also non-trivial. We do not anticipate "turn-key" full-system MPT, but rather methods that can be judiciously applied to key system design trades and decisions.

Summary of System Acquisition Challenges Addressed
Our research is developing analytical tradespace and affordability MPT to address two issues in DoD system acquisition:

(1) During protracted systems acquisition, operational conditions, needs, budgets and priorities change, causing turbulence in development, increased time and cost, sub-optimal system design, and sometimes program cancellation. In order not to be limited to technologies that are mature at the inception of the development program, DoD allows concurrent development in which subsystems are matured in parallel with the system development program. This introduces additional uncertainty regarding the cost, performance, burdens and compatibility of these technologies. MPT are needed to improve the resilience of system development with respect to these various sources of turbulence and uncertainty.
(2) During the long service lives of DoD systems, operational conditions and needs change, and adversaries adapt to avoid our systems' strengths and exploit our systems' limitations. We upgrade our systems to adapt to these changes and to incorporate new technologies. The ability to upgrade, and the cost of upgrade, depend on the reserve capacity and modularity of the initial design. Greater reserve capacity and modularity increase initial cost, but increases the scope of upgrades, reduces the cost of upgrades, and defers the need for a new system development. Decisions regarding reserve capacity and modularity involve complex tradeoffs.

Summary of Solution Approaches and Progress

A significant insight was that if the Government had a greater variety of solutions to choose from, with different combinations of performance and cost, the Government could more easily adapt to changes in operational conditions and capability needs, budget priorities, technology advances, etc. This would give the Government more options, while allowing the Government the flexibility for changing needs and priorities. We investigated how the Government could create this outcome, without significant change in development cost or acquisition procedures. We identified changes in acquisition methods or strategy that would produce this result. When we examined the implications for system design and development, we saw that it would also produce more versatile, adaptable and extensible systems as a result of the economic approaches that contractors would naturally take in their own best economic interests. This is a potentially significant result with large beneficial impacts, yet low cost and technical risk.

We developed frameworks for rigorous formulation and technical implementation of SBD methods. We developed a stochastic sequential decision-making formulation for SBD during protracted design. To address the challenge of the need for adaptability and extensibility in long-lived systems, we re-envisioned the production system (at Milestone C) as the root for a set of potential future variants. This is the historical precedent, but this vision has not been used as the basis for selection. We applied the principles of adversarial risk analysis (an offshoot of game theory restricted to the next few moves) to prioritize adaptation/extension capability.

4.1.2 SBD as an Acquisition Principle

These principles can be applied to address several challenges faced by DoD system acquisition:

(1) Operational conditions and needs and budget priorities can and do change during system development. The best estimates at Milestones A and B, which are written into Technology Development and into Engineering and Materiel Development contracts as performance requirements and cost targets, are only estimates of what the needs and priorities will be at the Milestone C production decision.

(2) In an effort to avoid being locked-in to old but mature technologies, contractors are allowed to propose new but not fully mature technologies in the expectation that they will mature during system development. The eventual cost, performance, burdens, and compatibility of the technologies are somewhat uncertain. More accurate estimates with less uncertainty are developed over time.

(3) Selection decisions and technology choices that made sense at an earlier milestone sometimes prove to "miss the mark" and be expensive to correct at a later milestone when conditions and needs have changed and new data on technologies are available.

(4) Successful DoD systems have long service lives – 60 years for a system and 20 to 60 years for an individual items of equipment are not uncommon. During this time, operational conditions, needs, and available technologies change. The systems and individual items of equipment are upgraded to add new mission capabilities, extend performance with different technologies, and increase the infrastructure capability to host added functions and technology burdens. The cost of upgrades and the limits to

which a system can be upgraded depend on the design selected at the original development decisions.

Current DoD practice is to publish performance requirements and an AUPC cost target at Milestone A for Technology Development, then issue multiple awards against the same requirements ("Competitive Prototyping"). When the conditions, operational capability needs, or budget priorities change, the Government is stuck: they have different solutions to the wrong problem. Sometimes DoD acquisition continues competitive prototyping into EMD, but still with the same set of requirements.

The DoD has attempted to generate solutions with somewhat different capabilities by publishing "threshold" and "objective" levels for some performance requirements, and tiered requirements with complicated leveling criteria (e.g., Tier 1 requirements are at the threshold level for key requirements. All tier 1 requirements must be met before any credit is given for tier 2 requirements. Tier 2 requirements are the objective level of key requirements. Proportional credit is given for partially reaching objective levels. Tier 2 requirements must be met before any credit is given for tier 3 requirements. Tier 3 requirements are not key and can be waived.). Contractors have objected to this system as being unclear as to how it will be applied in award decisions. While it is designed to give the Government some decision flexibility, it can be improved, simplified, and overcome contractor objections.

Current DoD objectives recognize the need for "resilient" systems that can be upgraded and/or adapted for different capabilities and technologies. Current practice is to address this need by requiring (a) some degree of "reserve capacity" (aka "design margin") in the system infrastructure, and (b) consideration of modular design. This approach is hampered by lack of appropriate SE MPT. There are no SE MPT to decide how much of what type of reserve capacity is needed, trading off the cost of providing the reserve capacity versus its potential benefit. MPT to evaluate the costs and benefits of different extent and level of modular design are lacking. Consequently, there are no standards and basis to justify costs or make cost-effectiveness tradeoffs. (Current DoD practice has also been to acquire families of systems, i.e., a set of variants of an underlying system design. This ensures, to some extent, that the underlying design is adaptable and extensible. It provides a constructive proof.)

Current DoD practice is commonly to accept a proposal for a single design contract. In many cases a contractor can submit multiple proposals, but each proposal is for a design concept.

These practices limit the Government's options to adjust to changes in need, budget priorities, etc. These practices tend to ensure that the Government receives proposals and designs that provide very similar capabilities for similar costs.

The Government could achieve a more robust and resilient acquisition system with minor changes to acquisition practices by employing the principles of SBD to the acquisition strategy. The guiding principles of SBD are

(1) To pursue multiple different solutions in parallel, i.e., a set of solutions with different characteristics
(2) To defer specification or down-selection decisions until there is more information to make a better decision

The goal is to produce greater range of cost and performance options at down-select decision points without increasing development cost. We formulated two methods to incorporate the principles of SBD into an acquisition strategy. The first is a very small change in practice that gives some of the benefits of SBD. The second has greater impact, by structuring the acquisition in a way that leads the contractors to generate sets of alternative solutions from which the Government can choose.

Simple Approach

There is a simple change in acquisition practice that would achieve some of the benefit through limited application of SBD principles. Currently the Government often issues multiple contracts for Technology Development for "Competitive Prototyping." All contractors get the same requirements. Sometimes this competitive prototyping is continued in EMD. Because all the solutions are designed to the same requirements, they tend to be not much different in cost and performance. If the conditions have changed, if the needed operational capabilities have changed, or budget priorities, the Government is stuck: they have multiple different solutions to the same wrong target. A simple change would give the Government the ability to defer final specification of requirements and costs. Instead of issuing multiple prototyping awards to one set of requirements, the Government could issue one award to each of multiple different sets of requirements. Then at down-select the Government would have greater cost and performance variety to choose from, and could pick the solution that best met the current understanding of needs and budget priorities.

More Complete Approach

Several changes to acquisition practice are needed to incorporate SBD as an acquisition strategy. First, instead of publishing point requirements or complicated tiered structures with threshold and objective levels, the Government could publish a range for each requirement saying that the levels for down-selection at the next milestone are expected to be somewhere in that range. The down-select criteria are not published at contract award, but guidance as to the potential range of down-select criteria are published in order have options with a wider range of cost and performance. Second, allow a contractor to propose a set of solutions based on common core elements and modular elements, such that different configurations of different modular yield designs with difference performance and cost. Contractors will automatically choose an appropriate degree of commonality and modularity that enables them to generate designs with a wide range of performance at low cost (This approach is sometimes called "kit based design". It is the modern approach to automobile and truck design. Volkswagen is the lead in this design approach among the auto industry). Third, at down-select, when the Government has updated information regarding operational conditions, capability needs, cost targets, technology cost and performance, etc., the Government will have a wider range of options and can select the option or subset of options most closely matching the current understanding of need. A significant ancillary benefit is that since designs are designed as modular assembly variations, they will have been naturally designed to facilitate later modification and upgrade.

The sequence of events within the standard acquisition life cycle model could go something like this:

(1) At Milestone A, the Government publishes capabilities needed, performance requirements and cost target as they do now. The change is that ranges are given for

capabilities, performance and cost. This is not a great change since the Government currently does something like this with "threshold" and "objective" values. The changes are to include cost, and to present the requirements range not as "threshold and objective" but as the range of what might be used at down-selection at the end of Technology Development.

(2) Contractors are allowed to propose pursuing multiple solutions during Technology Development as part of a single proposal, to span the range cost and performance range. Contractor will naturally tend towards modular design of common core elements in order to reduce their Technology Development cost and provide a greater range of cost and performance realizations in their set of solutions. Another way of explaining this is that contractors can propose a modular solution with common core elements that can be configured in different combinations yielding solutions with different performance levels and cost.

(3) At Milestone B, the Government publishes the requirements and cost targets for EMD phase. The new requirements reflect changes in operational needs and budget priorities as well as what was learned about costs and capabilities from competitive prototyping during Technology Development. Once again, the requirements and cost targets are the ranges of what might be used at the Milestone C decision.

(4) Contractors are allowed to propose pursuing multiple solutions during EMD as part of a single proposal, to span the range cost and performance range. Contractor will naturally tend towards modular design of common core elements in order to reduce their Technology Development cost and provide a greater range of cost and performance realizations in their set of solutions. Another way of explaining this is that contractors can propose a modular solution with common core elements that can be configured in different combinations yielding solutions with different performance levels and cost.

(5) At Milestone C, the Government selects a configuration (or none) for Low Rate Initial Production. However the operational conditions, budget priorities, etc. have changed, these changes are reflected in the configuration selection criteria. This approach gives the Government choices at Milestone C beyond Go/NoGo. There is an added benefit in that this approach will tend to yield a resilient design, i.e., one that can economically be modified later because, by design, it could be converted to one of its "siblings" by swapping modular elements. At Milestone C, the Government could consider resilience in the selection criteria – the cost of converting to siblings with different performance levels.

This approach does not require the Government to use or require any novel SBD algorithms or tools. It does not require the Government to develop requirements for modularity. It simply uses economic incentives that will cause the contractors to develop modular solutions, to give the Government selection options, and allow the Government to defer specifying the final decision criteria. It has the additional benefit of producing solutions that are naturally resilient by design. The implementation requires only a small change in Government acquisition

practices: giving a range of potential performance requirements and cost target, not as "threshold" and "objective" but simply that levels somewhere in the range will be used for down-selection. Contractors are allowed to produce a solution with multiple configurations than span the range of potential requirements and cost targets.

4.1.3 SBD in Design for Resilient Development during Protracted DoD System Acquisition

Summary

DoD material acquisition is often a protracted process. Consider the Joint Light Tactical Vehicle (JLTV), a tactical truck replacement for the HHHWV: the Technology Development phase from Milestone A to B took 4 years, and the Engineering and Manufacturing Development (EMD) phase from Milestone B to C took another 4 years. Many programs take longer. During this time, operational needs can change, budgets can change, and program management can change. These changes can impact system requirements and priorities. The JLTV program is an example. Several months before the Milestone B decision, program management decided that the average unit production cost (AUPC) target of $600K was untenable, and that the cost had to be competitive with an upgraded HMMWV, in the vicinity of $325K. This required the program to quickly explore concept changes and requirements tradeoffs to generate a new concept and new requirements for $325K AUPC. In other cases, such as the Future Combat Systems, the concept and requirements were not resilient with respect to changing operational needs, and this was a major factor leading to the program being cancelled.

SBD has been suggested as a development methodology that is resilient with respect to changes priorities and operational capabilities during the development process, as well as being resilient with respect to new information regarding the cost, capability, and compatibility of subsystems and technologies in the system concept. The principle of SBD for resilient development is to carry multiple diverse concepts forward, downselecting and revising the set (1) as new information arrives, and (2) as the system development progresses. However, Systems Engineering MPT to select which and how many concepts to carry forward, and at what points during development to revise the concept set are lacking. A rigorous formulation for system development SBD is needed, upon which MPT for SBD decisions within the DoD system acquisition lifecycle can be developed.

One of our major research areas in 2015, and going forward during 2016, addresses this deficiency. The research goal is to develop a rigorous analytical formulation of SBD for resilient system development, and practical MPT to realize and implement the capability. In 2015 we developed the rigorous analytic formulation, and made substantial progress defining the application procedures and beginning to develop the application mathematical tools. This included specifying "helper" tools and data requirements to integrate the use of the SBD methods with development program planning and scheduling.

Details

This section describes a formalism and approach for developers to employ SBD in concepting and design in order to maximize value to the Government at some future decision point. The premise is that at the Government publishes performance requirements and cost targets as point values (not ranges as described in the previous section). The cost targets, performance requirements and priorities, etc. may change during the development process. Updated information regarding the cost, performance, burdens and compatibility of some technologies may become available during system development. These uncertain and variable factors are

referred to as *externalities* because they are exogenous inputs to the design process. They are treated as random variables with probability distributions that change over the course of the acquisition life cycle.

We assume there exists a function to calculate the value of a specific design given specific values of the externalities. The value function includes average unit production cost a performance on multiple dimensions. The value function does not include development costs, which are treated separately since they are a set property. The design value function produces a scalar output for specific values of the externalities.

When we have a set of designs to choose from, and specific values of the externalities, the value of the set is the maximum value overall the designs in the set, i.e., we get to choose the best solution. During development, the externalities have some probability distribution. Applying the set value function for a set of designs yields a distribution of value. For a given solution set, different realizations of the externalities will lead to different choices. Since the externalities have a probability distribution, we get a distribution of set value.

We need to be able to compare the value of sets under uncertainty to chose the "best" set to pursue. To do this we need to convert the distribution of value to a scalar. A robust approach is to pick a percentile of the distribution as a reference point. The scalar value of a set under uncertainty is then the Nth percentile of the distribution of set value given the distribution of the externalities. Choosing the 50^{th} percentile uses the median value. Choosing a low percentile is risk adverse; choosing the 0^{th} percentile uses the worst-case value. Choosing a high percentile is optimistic; choosing the 99^{th} percentile uses the best-case value.

(An alternative formulation is possible that compares sets over the distribution of externalities, i.e., sets are compared to each other over the distribution of externalities. This approach is not being pursued.)

We consider a framework with three decision points: Milestones A, B and C. Prior to Milestone A, there is some universe of candidate solutions. An initial set of solutions is chosen at Milestone A. The cost to develop the set from Milestone A to B is not simply the sum of individual costs since some of the solutions in the set may share design elements and subsystems. At Milestone B, the cost to develop a set of solutions to Milestone C depends on which solutions were in the set chosen at Milestone A. Adding a new solution might not cost much if it were similar to a solution in the set at A or if it shared features with several different solutions. We assume these data are available.

At Milestone C all development costs are sunk costs and do enter into consideration. Prior to Milestone C, set selection needs to consider development cost as well as set value. We assume that, prior to Milestone C, there is some function that combines cost to develop to the next milestone and set value.

At this point we can case SBD as a sequential decision process, with a stochastic optimization solution. At Milestone A, we consider all possible solution sets and pick the solution set with the highest value assuming that at Milestone B we only prune the set and do not add new solution, and at Milestone C we pick a solution from the set remaining after Milestone B. Value includes set development cost from A to B, and development cost for the reduced set from B to C. At Milestone B we again consider all possible solution sets (not just subsets of the set chosen at A but can add solutions), and pick the solution set with the highest value assuming that at Milestone C we will pick a single solution. Value includes the set development cost from B to C.

At Milestone C we consider only solutions developed up to this point and pick the singleton solution with the highest value.

SBD approach dominates the single-point solution, i.e., the expected outcome is never worse, because, as developed, the SBD methodology could yield a singleton set, i.e., a single design in the solution set. Point-based design is a subset of SBD. SBD has the potential to be more efficient and effective because it considers the cost of changing designs and the potential performance shortfall with respect to changing requirements.

4.1.4 MPT to Determine Initial Reserve Capacity Requirements for Long-Lived DoD Systems

Summary

DoD systems tend to be long-lived. Air, land and sea systems are commonly planned to be in the use for 30 or more years, but end up being in use for 50 or 60 years. Individual vehicles, aircraft and ships often remain in use (with recapitalization) for twice their intended service life. During this time the system designs are upgraded, and individual platforms are upgraded during recapitalization to adjust to changing operational needs and available technologies. As an example, the M113 had five major block upgrades during its 60 years in service.

During development

- MPT are needed to assess the platform needs to support the range of potential future variants' functional and performance capabilities, and the cost of the upgrades
- MPT are needed to Analyze the likelihood of future needs relative to the capability fielded (e.g., adversarial risk models)
- MPT are needed for time/cost/risk tradeoffs among initial production and potential future upgrade capabilities and costs

Basic system limitations limit what can be done with them. The HMMWV was and is a versatile platform. But as we added weapons and an armored cupola, we raised the center of gravity (CG) and increased the propensity to roll-over (a very serious problem and cause of loss of life). There was no performance envelope for changes in the CG, no requirement for how much the CG could shift without causing roll-over in the reference conditions. When we needed to increase protection, mobility was critically degraded because the powertrain, suspension and steering did not have the reserve capacity to handle the increase weight and angular inertia of additional armor. A thorough understanding of potential upgrades, their burdens and implications for other dimensions of system performance are needed to acquire systems with potential to be adapted for evolving operational needs.

The cost of upgrading a system depends on the original concept and design. The original concept and design enable some upgrades and make others infeasible or unaffordable. When different upgrades and different mission packages are installed, the original platform spawns a family of variants for different missions. The services recognize these dynamics and has begun to include requirements for

(1) Reserve capacity (aka design margin) in the basic platform infrastructure to support later upgrades and additional demands for size, weight, power, cooling, etc.

(2) Modularity to facilitate replacement of subsystems with new technologies, and to be able to incorporate new mission packages.

Recent ground vehicle acquisitions have included requirements for electrical power reserve capacity for X% growth in demand, engine power reserve capacity for X% growth in weight, and so on. The requirements also include a set of mission variants – acquiring a family of systems – and thus ensure some degree of mission modularity. The JLTV and the Armored Multi-Purpose Vehicle (AMPV) are recent examples.

However, the acquisition commands lack the MPT to derive and justify these requirements. There are development and AUPC costs to reserve capacity and modularity. If the costs cannot be justified, the requirements can be rejected as "gold plating" – requiring more than is needed. Furthermore, the benefits of reserve capacity will not be realized unless the reserve capacity requirements are complete and balanced. A requirement for electrical power reserve capacity for 50% growth should have a companion requirement for cooling reserve capacity for 50% growth. A requirement for engine power for 25% weight growth should have companion requirements for the suspension and running gear as well as the rest of the drive train. A blanket statement such as "design all subsystems for 25% weight growth" is good at the level of a capability development document, but has to be devolved into quantitative subsystem requirements organized by the Work Breakdown Structure and system architecture.

We are conducting research to develop practical and effective MPT to derive and justify requirements for reserve capacity and modularity. In 2015 we focused on reserve capacity and plan to complete development of these MPT before expanding to address modularity. In 2014 we developed a framework for tradeoff tradespace framework for bi-directional qualitative and quantitative relationships between the product design and the requirements specification – to inform performance-based requirements and design tradeoff decisions and to ensure complete and balanced derived subsystem requirements. We did not elaborate this framework in 2015, but plan to instantiate it in 2016 as a proof of concept and as part of a demonstration of a complete package of MPT for reserve capacity requirements.

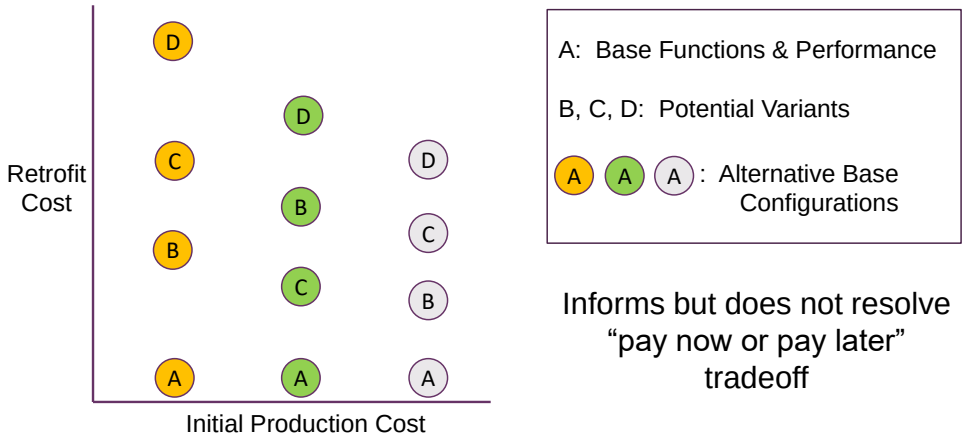
Our approach to deriving reserve capacity requirements uses a SBD formulation combined with adversarial risk analysis. The key concept underlying the SBD formulation is that the original system design should not be viewed and evaluated as simply as an item with some cost and performance, but rather as a both a specific realization with cost and capabilities and set of potential variants with conversion costs and differentiating capabilities. Reserve capacity may increase the initial AUPC, but increases the scope of potential variants and reduces their cost.

Future needs are uncertain, but adversarial risk analysis provides an analytic framework to choose a starting point that gives us the capability to adjust to changing operational conditions, shape threat options, and limit threat effectiveness. Adversarial risk analysis assumes that future threats will tend to choose conditions, tactics and material that exploit limitations of our systems while avoiding our strengths. The SBD formulation uses a tradespace framework that combines a tradespace of system capabilities versus the set of potential variants, together with adversarial risk model of likelihood of need, to choose a resilient and cost-effective initial design and family of variants vis-à-vis the requirements for reserve capacity.

Details

The selection for initial production is the base configuration. It has some initial production cost, some initial capabilities, and retrofit costs to achieve upgraded capabilities (the term "cost" refers to both monetary cost and the cost of lives, tactical and strategic advantage lost due to delay in fielding higher capability systems). For a given initial configuration, we can estimate

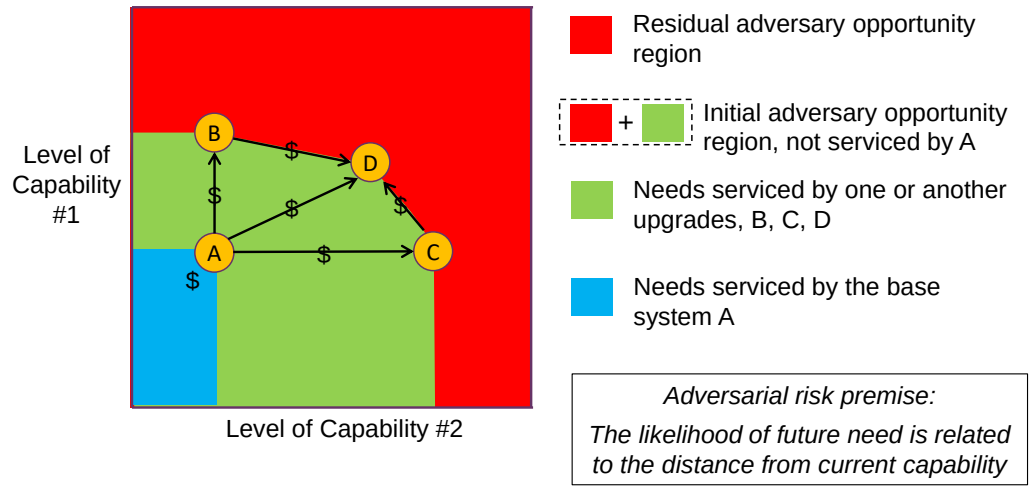
the costs to retrofit/upgrade to achieve different levels of capability in variants (see illustration). This picture is insufficient to inform decision-making because it does not address the relative likelihood that the different future capabilities will be needed, and the likely time until they will be needed.



Adversarial risk analysis provides methods to get insight into the relative likelihood of different capability needs. Adversarial risk analysis posits that future adversaries will choose theaters, conditions, tactics and materiel to exploit the limitations of our systems while avoiding their strengths. This provides an approach to bound the problem. It does not address the technical or geo-political influences.

The second figure provides a richer picture. A system has many different capability dimensions. For purposes of illustration, we only consider two in the illustration.

This picture is insufficient to inform decision-making because it does not address the relative likelihood that the different future capabilities will be needed, and the likely time until they will be needed.



A system of some type is meant to fill a role as a member of a larger portfolio of systems. Each has their own range of capability, with some overlap. Each has a limit. If it cannot service the threat, another system takes over. If it vastly overmatches the threat, the threat is handed off (don't use a hammer to swat a fly). For a given system type, there is an upper and lower bound

on the level of capability needed to accomplish the range of missions for the niche it fills in our portfolio of systems. This defines the "capability cube" shown in two dimensions.

The initial operating capability (A) sits at some point in the capability cube, with some AUPC. It can handle any combination of capability needs to the lower left. In the future, we may need a system somewhere else in the cube.

In the adversarial risk rate model, the rate that some adversary emerges outside of the region the initial system (A) can handle is inversely proportional to the region of the cube that system (A) can handle.

The adversarial risk formulation generally assumes adversaries avoid our strengths and exploit our weaknesses. In the figure, our system is weaker in one capability than another, so an adversary is more likely to emerge (or more properly, will emerge at a higher rate) needing a higher level of service on capability #2.

We identified three main alternative adversarial models. Two of them bound the problem, and two provide principled reference points.

- Non-adaptive adversaries. Adversaries emerge independent of our capabilities. The distribution of adversary situations is uniform across the cube
- Perfectly adaptive adversaries for this system mission, capability requirements are at the upper right hand corner
- We-were-smart and they-are-smart (We balanced our portfolio of capabilities to overlap limiting threat options between what this system can do and what its neighbors can do. The potential adversary makes their best choices under these constraints)

Our initial choice of (A) has an AUPC and some capability on each dimension. But it is also the seed for a family of variants and needs to be evaluated in that context. From the initial system (A) we can transition to variants (B), (C) and (D) that give increase capability at some cost depending on the trajectory over time. A different starting point will have different trajectory paths.

Full analysis of how the shape of the network and costs, and uncertainty as to when what capabilities are available and costs, requires further research. Much of the "data" needed for such analyses is speculation, and therefore methods are needed that are robust with respect to data accuracy and uncertainty.

The competing rate model gives some further insight into relative time until a particular capability is needed as related to the probability the capability. In the competing rate model, adversaries requiring different capabilities appear at some rate, i.e., there is some mean time until they appear, and the time until they appear has a negative exponential distribution. This is sufficient to construct a stochastic time trajectory of what capabilities will be needed when. This can be combined with standard "Net Present Value" methods for the time value of money to inform cost-effective initial decision. These "best practice" methods have to be combined the non-linear budget ceilings, and motivation considerations: the Program Manager responsible for bringing the initial production in on schedule and within budget will be gone in a few years, while the system will be in service long afterwards.

5 Georgia Tech Research Institute (GTRI)

5.1. Summary of Phase 1 and Phase 2 Results

5.1.1 Phase 1

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 tradespace exploration for early phase design of military ground vehicles (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, the FACT framework 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, 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.

The Systems Modeling Language (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 model-based systems engineering (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

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.

OpenMDAO is an open-source Multidisciplinary Design Analysis and Optimization (MDAO) framework developed by NASA Glenn and Langley Research Centers (OpenMDAO, 2014). It has

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

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 7**, 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 Django[3] and 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 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

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

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.

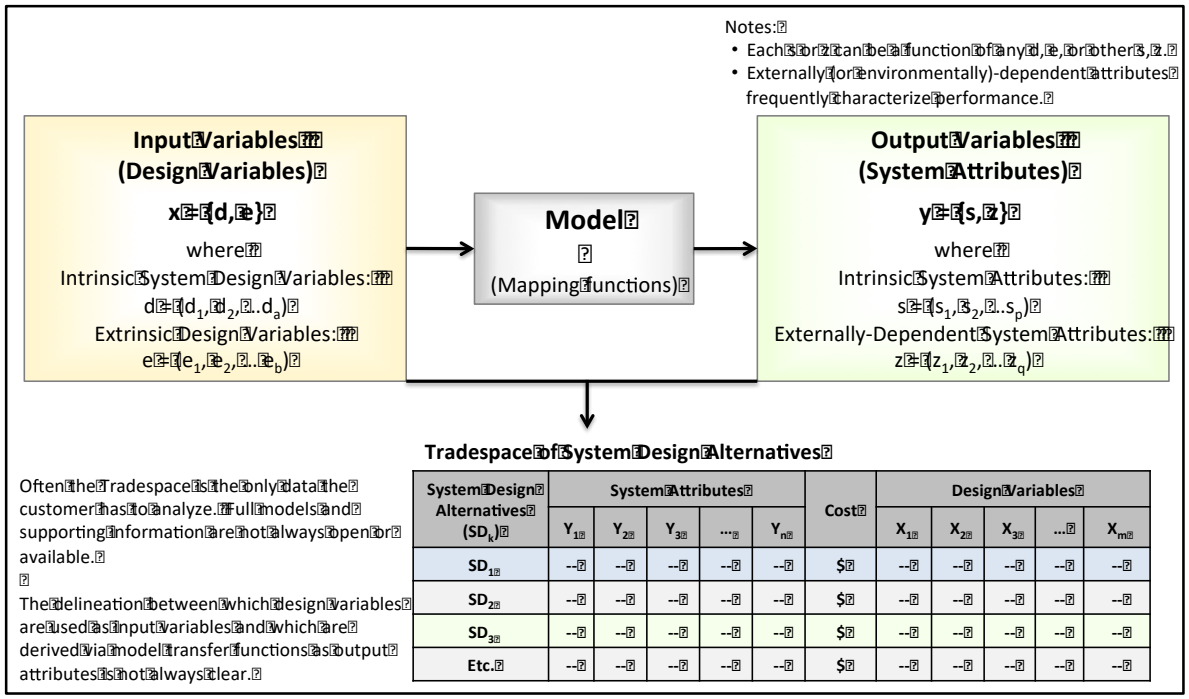


Figure 7: Defining a Tradespace

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 8**. The toolset, workflow, and example operationalized construct were described and published as part of INCOSE 2014 (Sitterle, Curry, Freeman, and Ender 2014).

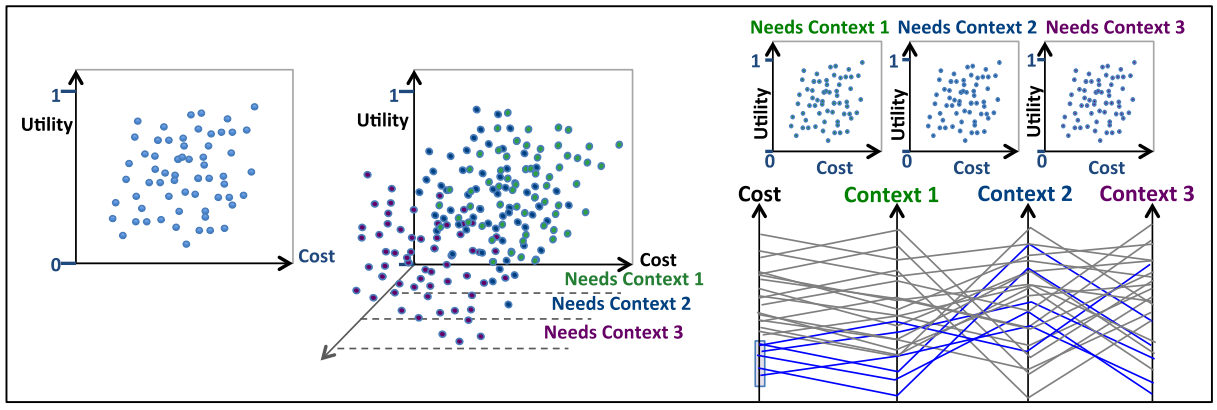


Figure 8: Classical 2D vs. Needs Context 3D Utility

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. SUMMARY OF PHASE 3 RESULTS

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 9**. 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.

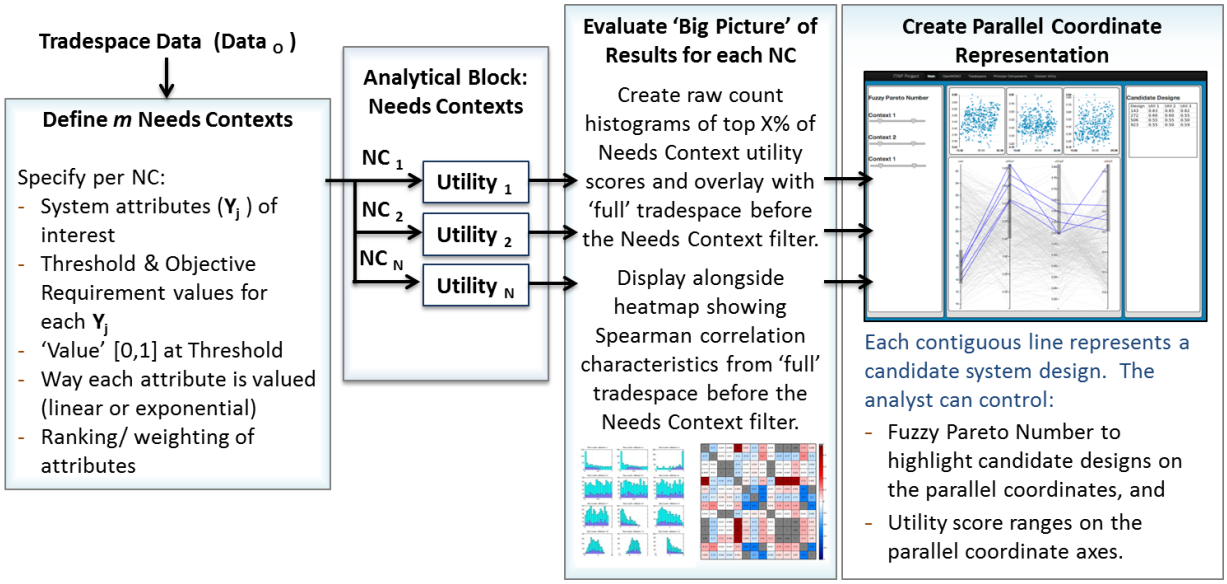


Figure 9: Graphical depiction of Needs Context Steps and Workflow

We 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, we 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 Python 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 pipeling 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

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. 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 further investigated methods whereby we may operationalize these formalisms into measureable and executable constructs to support Pre-Milestone A analysis. The work proposed for Phase 5 in FY 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 has sought to preserve an open framework and approach that promotes quantitative and qualitative transparency of the tradespace refinement. Our goals were 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. Keys to synthesizing these constructs are scalability, flexibility, and modularity of the construct as well as the workflow and processes we are striving to integrate. Our objective has been 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 in Phase 4, GTRI continued 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 added modeling and analysis dimensions 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 enables the evaluation of different environment influences relevant to fielded operational performance.

There are many –ilities being investigated across the ITAP teams that can 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 (ANSI/AIAA G-043- 1992, “Guide for the Preparation of Operational Concept Documents”). 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 continued 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 had 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 |

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 has been the supporting computational methods, processes, and tools that also required 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 the Phase 4 framework along with understanding the algorithms needed and implementing the salient portions in a modular fashion.

5.4. PLANS FOR PHASE 5

For Phase 5 in FY 2016, we plan 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 region 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 work performed in Phase 4.

5.5 REFERENCES

1. Assistant Secretary of Defense (Networks and Information Integration) and DoD Chief Information Officer, DoD Information Assurance Certification and Accreditation Process (DIACAP), Department of Defense Instruction 8510.01, 2007.
2. Assistant Secretary of Defense (Networks and Information Integration) and Chief Information Officer, Clarifying Guidance Regarding Open Source Software (OSS), Department of Defense Memorandum, 2009.

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. Hassine-Guetari, Soumaya Ben, Jérôme Darmont, and Jean-Hugues Chauchat. "Aggregation of data quality metrics using the Choquet integral." *Aggregation of data quality metrics using the Choquet integral* (2010).
5. 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.
6. Charnetski, Johnnie R., and Richard M. Soland. "Technical Note—Statistical Measures for Linear Functions on Polytopes." *Operations Research* 24, no. 1 (1976): 201-204.
7. Charnetski, Johnnie R., and Richard M. Soland. "Multiple-attribute decision making with partial information: The comparative hypervolume criterion." *Naval Research Logistics Quarterly* 25, no. 2 (1978): 279-288.
8. Christel, Michael G., and Kyo C. Kang. *Issues in requirements elicitation*. No. CMU/SEI-92-TR-12. Software Engineering Institute, Carnegie Mellon University, Pittsburgh, PA. 1992.
9. 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.
10. Fricke, E., & Schulz, A. P. Design for changeability (DfC): Principles to enable changes in systems throughout their entire lifecycle. Systems Engineering, 2005; 8(4).
11. Fu, Guangtao, Jim Hall, and Jonathan Lawry. "Beyond probability: new methods for representing uncertainty in projections of future climate." *Tyndall Centre for Climate Change Research Working Paper* 75 (2005).
12. Goerger, S.R., Madni, A.M., and Eslinger, O.J. "Engineered Resilient Systems: A DoD Perspective," *Conference on Systems Engineering Research (CSER 2014)*, Procedia Computer Science, 2014; 28, 865-872.
13. Grabisch, Michel. "The application of fuzzy integrals in multicriteria decision making." *European journal of operational research* 89, no. 3 (1996): 445-456.
14. Grabisch, Michel, and Christophe Labreuche. "A decade of application of the Choquet and Sugeno integrals in multi-criteria decision aid." *Annals of Operations Research* 175, no. 1 (2010): 247-286.
15. Grabisch, Michel, and Marc Roubens. "Application of the Choquet integral in multicriteria decision making." *Fuzzy measures and integrals* 40 (2000): 348-375.
16. Hazelrigg, George A. "A framework for decision-based engineering design." *Journal of mechanical design* 120, no. 4 (1998): 653-658.
17. Lahdelma, Risto, Joonas Hokkanen, and Pekka Salminen. "SMAA-Stochastic multiobjective acceptability analysis." *European Journal of Operational Research* 106, no. 1 (1998): 137-143.
18. Lahdelma, Risto, Kaisa Miettinen, and Pekka Salminen. "Ordinal criteria in stochastic multicriteria acceptability analysis (SMAA)." *European Journal of Operational Research* 147, no. 1 (2003): 117-127.
19. O'Neal, Michael, Tommer R. Ender, Daniel C. Browne, Nicholas Bollweg, C. Justin Pearl, and Joe L. Brico. 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.
20. OpenMDAO; 2014. http://openmdao.org/
21. Parnell, Gregory S., Jack A. Jackson, Roger C. Burk, Lee J. Lehmkuhl, and Joseph A. Engelbrecht. "R&D concept decision analysis: using alternate futures for sensitivity analysis." *Journal of Multi-Criteria Decision Analysis* 8, no. 3 (1999): 119-127.
22. Ross, A., and Rhodes, D. 2008. "Architecting Systems for Value Robustness: Research Motivations and Progress," 2nd Annual IEEE Systems Conference, Montreal, Canada, April 2008.
23. Ross, Adam M., and Donna H. Rhodes. "Using natural value-centric time scales for conceptualizing system timelines through epoch-era analysis." In *INCOSE International symposium*. June 2008.
24. Ryan, E. T., Jacques, D. R., & Colombi, J. M. An ontological framework for clarifying flexibility-related terminology via literature survey. Systems Engineering, 2013; 16(1), 99-110.
25. Schulz, Armin P., and Ernst Fricke. "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, 1999.
26. Sitterle, V., Curry, M., Freeman, D., and Ender, T. "Integrated Toolset and Workflow for Tradespace Analytics in Systems Engineering," 24th Annual International Council on Systems Engineering (INCOSE) Symposium, Las Vegas, NV, June 30- July 3, 2014.
27. Tervonen, Tommi, and Risto Lahdelma. "Implementing stochastic multicriteria acceptability analysis." *European Journal of Operational Research* 178, no. 2 (2007): 500-513.

5.5 SysML-based Cost Modeling (GT, USC, NPS)

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.

5.5.1 Introduction & ITAP Context

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.

5.5.2 Approach towards 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.

5.5.3 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.

5.5.4 CANDIDATE EXTENSIONS, ADDITIONAL APPLICATIONS, AND CASE STUDIES

Our 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 (https://github.com/Open-MBEE), 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.

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, the Constructive Cost Model (COCOMO) for software, as well as COCOMO extensions for quality. Efforts are currently underway to update the COSYSMO and COCOMO cost models (COSYSMO 3 and COCOMO III). When these new cost model calibrations are complete, they can be migrated to the SysML based cost modeling framework.

Other envisioned applications, some based on planned upgrades to COSYSMO and COCOMO, 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 10.

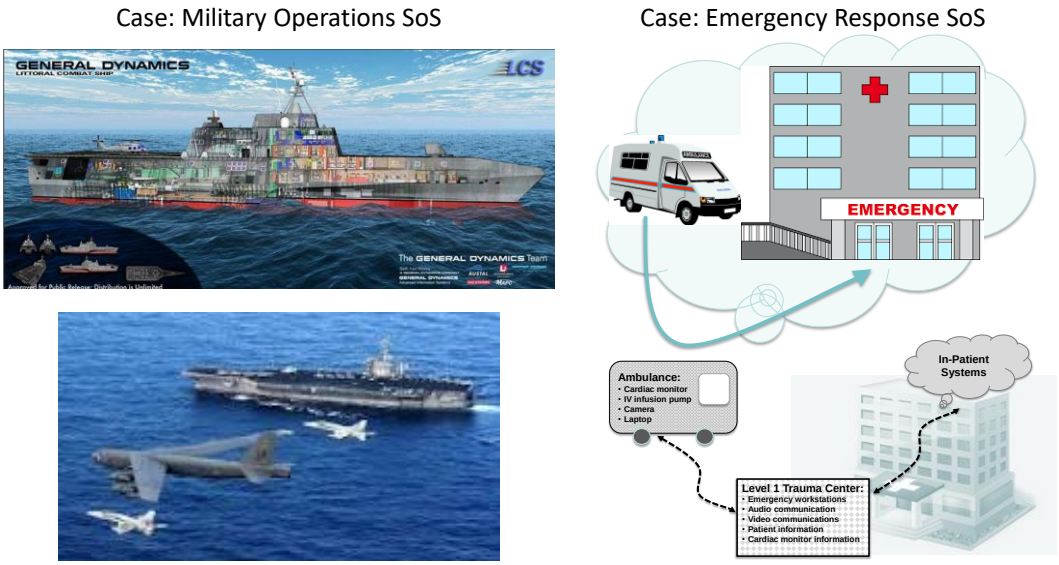


Figure 10 – Candidate future case studies.

5.5.5 SUMMARY

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).
  - Acausal; 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.

5.5.6 REFERENCES

M Culler (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.

JA Lane (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.

JA Lane (2010) Evolution of Systems Engineering Cost Estimation. Chapter 6 (unpublished draft) for cost modeling book. http://csse.usc.edu/

RJ Madachy (May 1997) Heuristic Risk Assessment Using Cost Factors. *IEEE Software*.

RJ Madachy (2013) Expert COSYSMO - Systems Engineering Cost Model Risk Advisor, http://csse.usc.edu/tools/ExpertCOSYSMO.php (as accessed Nov 2013)

RS Peak and JA Lane (2014) SysML Building Blocks for Cost Modeling: Towards Model-Based Affordability Analysis. INCOSE International Workshop (IW14), Torrance, California.

RS Peak, CJJ Paredis, LF McGinnis, SA Friedenthal, RM Burkhart, 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/*

Y Romaniw, B Bras, T Guldberg (2011) Sustainable Manufacturing Analysis using Activity Based Costing in SysML. ASME IDETC/CIE, Washington DC.

Y Romaniw and B Bras (2010) Sustainable Manufacturing Analysis using an Activity Based Object Oriented Method. SAE Journal of Aerospace 2(1) 214-224.

DR Tamburini, RS Peak, CJJ Paredis (2005) Composable Objects (COB) Requirements & Objectives v1.0. Technical Report, Georgia Tech, Atlanta. *http://eislab.gatech.edu/projects/nasa-ngcobs/*

6 Penn State University (PSU)

6.1 Activity 1 (Summary of Phases 1 and 2 Results (summarizing the counterpart Phase 2 report content for the activity)

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 has its roots in the domain of marketing, which is fundamentally concerned with how consumers choose. While there are many different approaches to modeling 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 Fig. 1, 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].

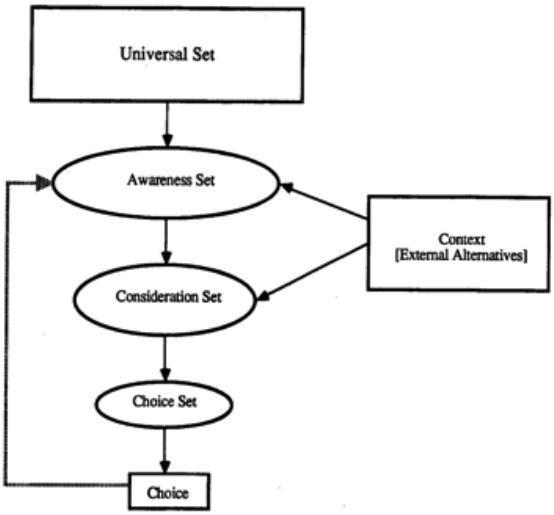
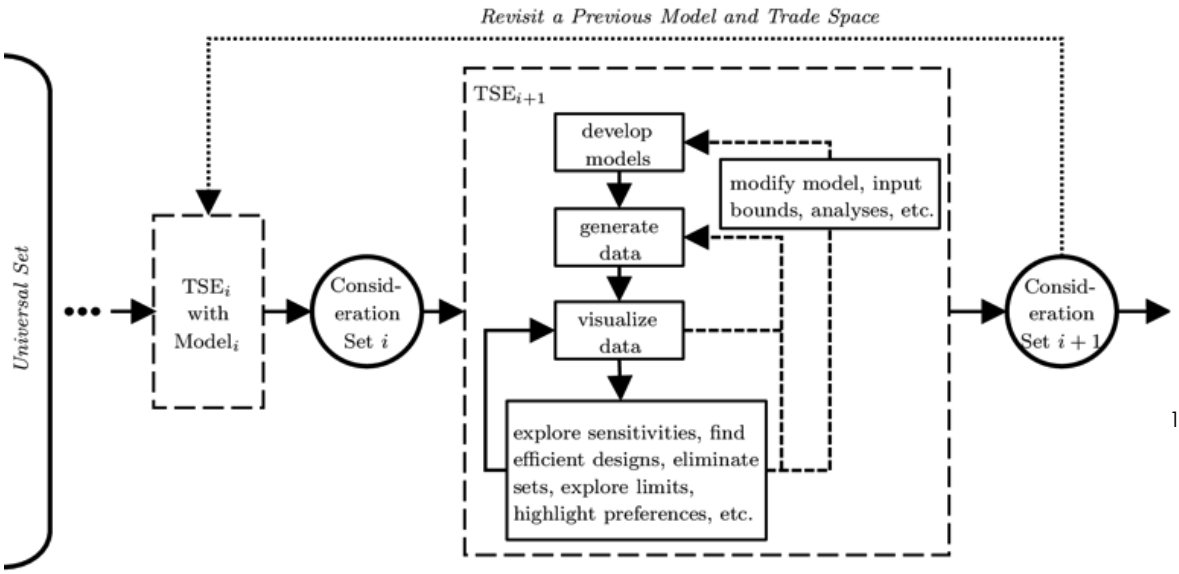


Figure 11. Model of decision making in marketing

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 2 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.



6.2 Phase 3 and 4 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-modeling 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

$$c_i = a_i + b_i q_i .$$

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.

6.2.2 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.

6.2.3 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:

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

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

6.2.4 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):

$$M_3$$
$$M_1 M_3$$
$$M_2 M_3$$
$$M_1 M_2 M_3$$

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:

$$c_3 = a_3 + b_3 q$$
$$c_{13} = a_1 + a_3 + b_1 q + b_3 q_1$$
$$c_{23} = a_2 + a_3 + b_2 q + b_3 q_2$$
$$c_{123} = a_1 + a_2 + a_3 + b_1 q + b_2 q_1 + b_3 q_2$$

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.

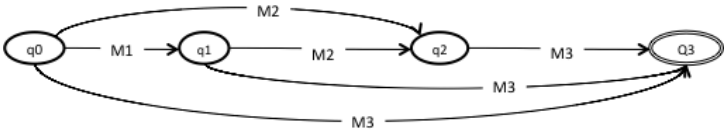


Figure 12. Sequencing of models

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

$$M_{123} = \min\left[M_1 M_3, M_{12} M_3, M_3\right]$$

while the best modeling approach to get to state q_2 is

$$M_{12} = \min\left[M_1 M_2, M_2\right]$$

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

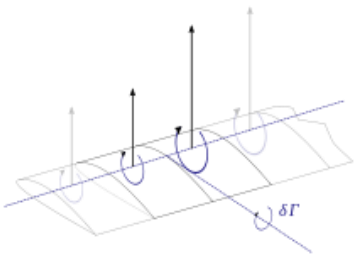
$$\text{number of combinations} = \binom{N+1}{2}.$$

6.2.5 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.



An 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 (**Error! Reference source not found.**). This improved modeling approach involves both increasing the number of 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 4. 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.

6.2.6 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 $g_l(x) \leq \min_y g_d(x, y) \leq g_u(x)$. 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 13 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*.

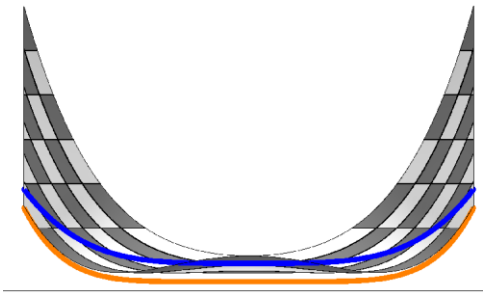


Figure 13. Model value space

Reconsidering the wing design problem, let each of 201 parameters of $\{a_1, \text{chord}_1 \cdots, \text{chord}_{200}\}$ 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 *M2* and *M3* as concept and detailed respectively, then

$$X = \{a_1, \text{chord}_1 \cdots, \text{chord}_{20}\}$$
$$Y = \{\text{chord}_{21} \cdots, \text{chord}_{200}\} .$$

The model *M3* takes in all of *M2*'s variables plus additional ones. For M_2 to be a proper conceptual model with respect to M_3 it should return two values,

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

6.3 Phase 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.

6.2.7 Phase 4 End Results

Specifically in Phase 4, PSU focused 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 implemented the simple wing design model with varying number of wing segments from 10s to thousands, and treated 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. We are preparing to publish at upcoming conferences, and to submit to a journal our detailed results.

Under separate funding, PSU experimentally deployed 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 offered a first opportunity to prove out the techniques on practical problems.

6.2.8 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.

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 (AFIT)

7.1 Phase 1 and 2 Results

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 infervalue 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 Masters 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 Phase 4 Work - Combined AFIT and NPS RT-137 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 Phase 4 tasks were performed 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 produced 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 was 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.

Phase 5 Plans

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 (NPS)

8.1 - Summary of Phases 1 and 2 Results

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 - Phases 3 and 4 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.

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) [5][6] 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 14**.

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.

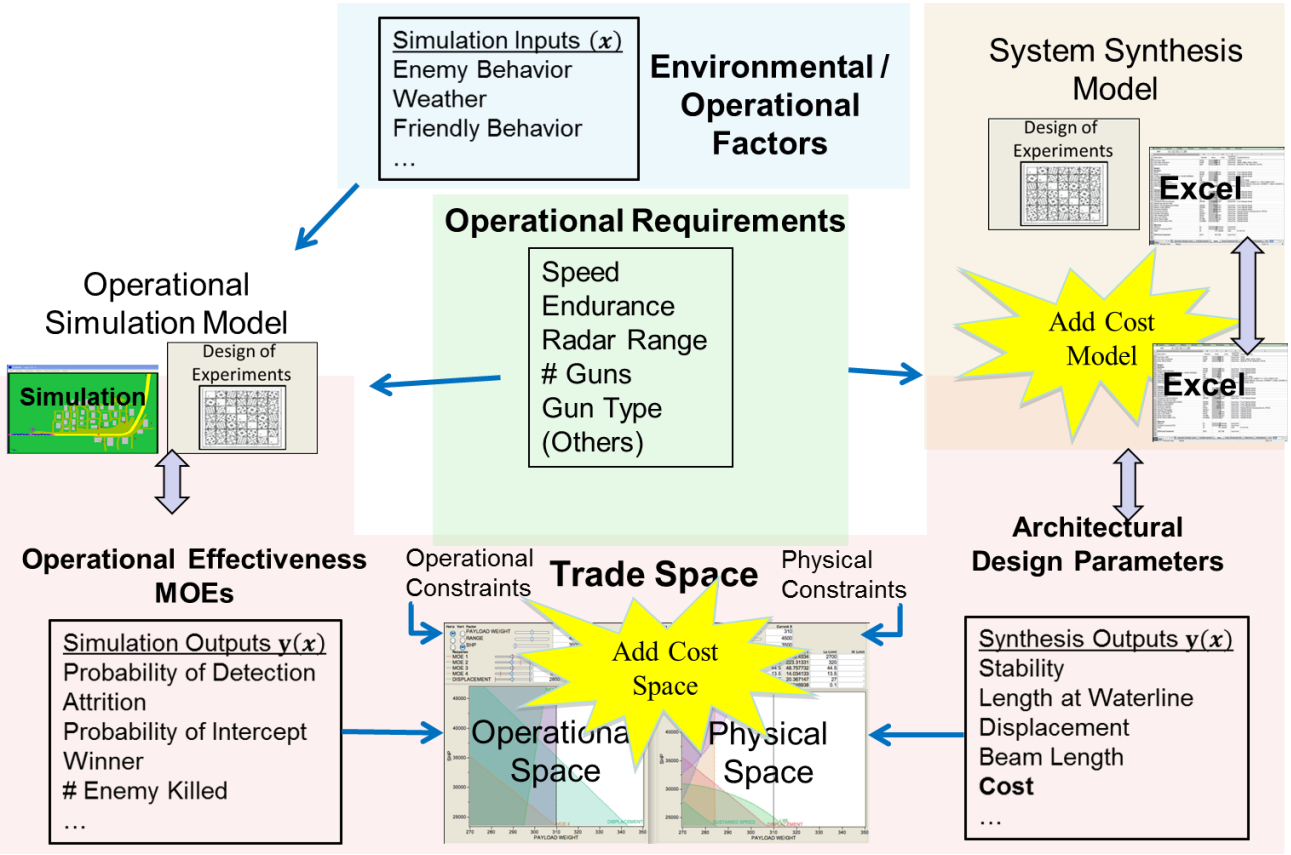


Figure 14: MBSE Design Process

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 15**. 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.

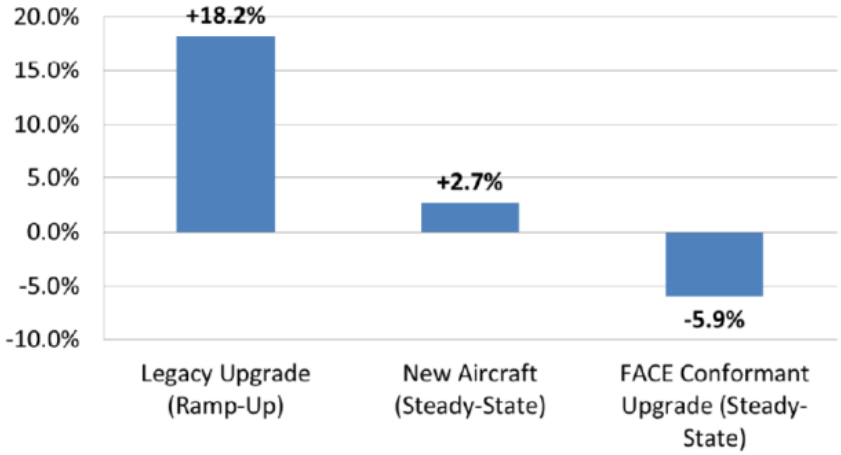


Figure 15: Representative Impact of FACE Architecture on Effort

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.

The parametric cost modeling and survey enhancement activities continued into Phase 4. Assumptions were evaluated and parameter values refined. The ongoing study results continued to provide insights into sources of cost savings.

Task 2 also continued a collaboration with AFIT for a joint Intelligence, Surveillance and Reconnaissance (ISR) mission application involving heterogenous teams of autonomous and cooperative agents. NPS provided 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 continue to develop and refine a common application and integrated effort for validating several of the approaches they have been developing and refining during Phases 3 and 4. A System-of-System level architecture will be refined 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.

8.3 Phase 4 Results

- **Developed an initial 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, began developing the Parametric View.

- **Transitioned the initial baseline architecture to the Monterey Phoenix environment.** The MP process can 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.

- **Utilized the executable architecture modeling framework of MP to perform initial 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. Initial sets of scenarios generated by MP were transformed by automated tools into implementation testing suites, another huge benefit providing the continuity of design from the early architecture models to the implementation.

- **Designed and Demonstrated an initial 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 can 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 can 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.

- **Developed initial 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. Experimented with attaching cost data to every actor and event in every possible scenario, computing the cost of each scenario, were it to occur. With the above steps and Phase 5 extensions, 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 plan to compare, contrast, and possibly integrate methods in Phase 5 for cost modeling in MP and SysML based 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 produces 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.

8.4 Phase 5 Plans

- **Continue to evolve the initial 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, continue to develop the Parametric View.

- **Continue to migrate the initial baseline architecture to the Monterey Phoenix environment.**

- **Extend the initial executable architecture modeling framework of MP to perform automated assertion checking and find counterexamples of behavior that violate the expected system's correctness**.

- **Continue to Extend and Demonstrate the Initial ISR UAV tradespace**. The tradespace capabilities will include 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.

- **Continue to extend and evolve 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**. Use the results of Phase 4 to attach cost data 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.

One of our objectives is to increase the resolution of source data used for cost model computations. We plan to compare, contrast, and possibly integrate methods in Phase 5 for cost modeling in MP and SysML based 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.

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 developed 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 engaged domain experts for Delphi estimates, evolved 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, Phase 5 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 Phase 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 will 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, we experimented with extending MP with cost modeling capabilities. In Phase 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.

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

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 16 shows the inputs and outputs for the system-level product line model.

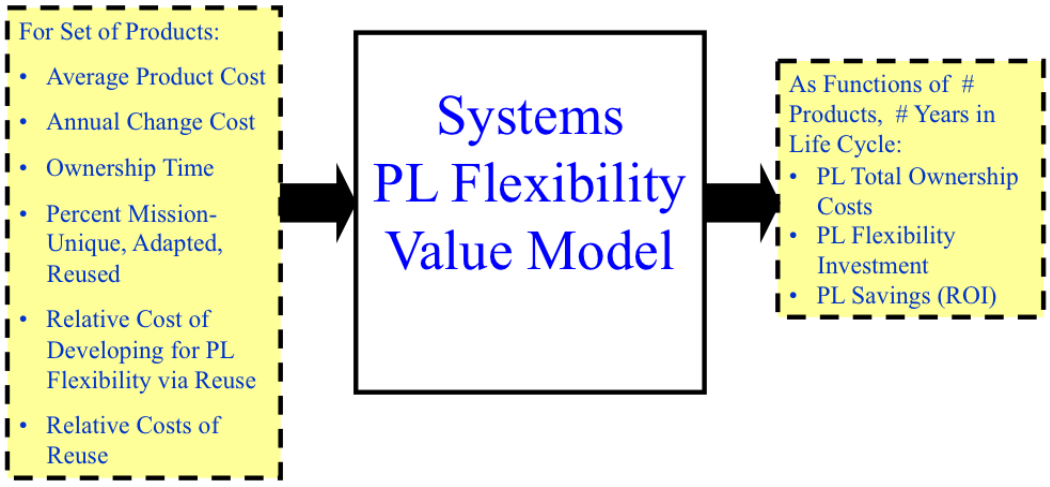


Figure 16: Systems Product Line Flexibility Value Model

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 17** shows the consolidated TOC and ROI outputs.

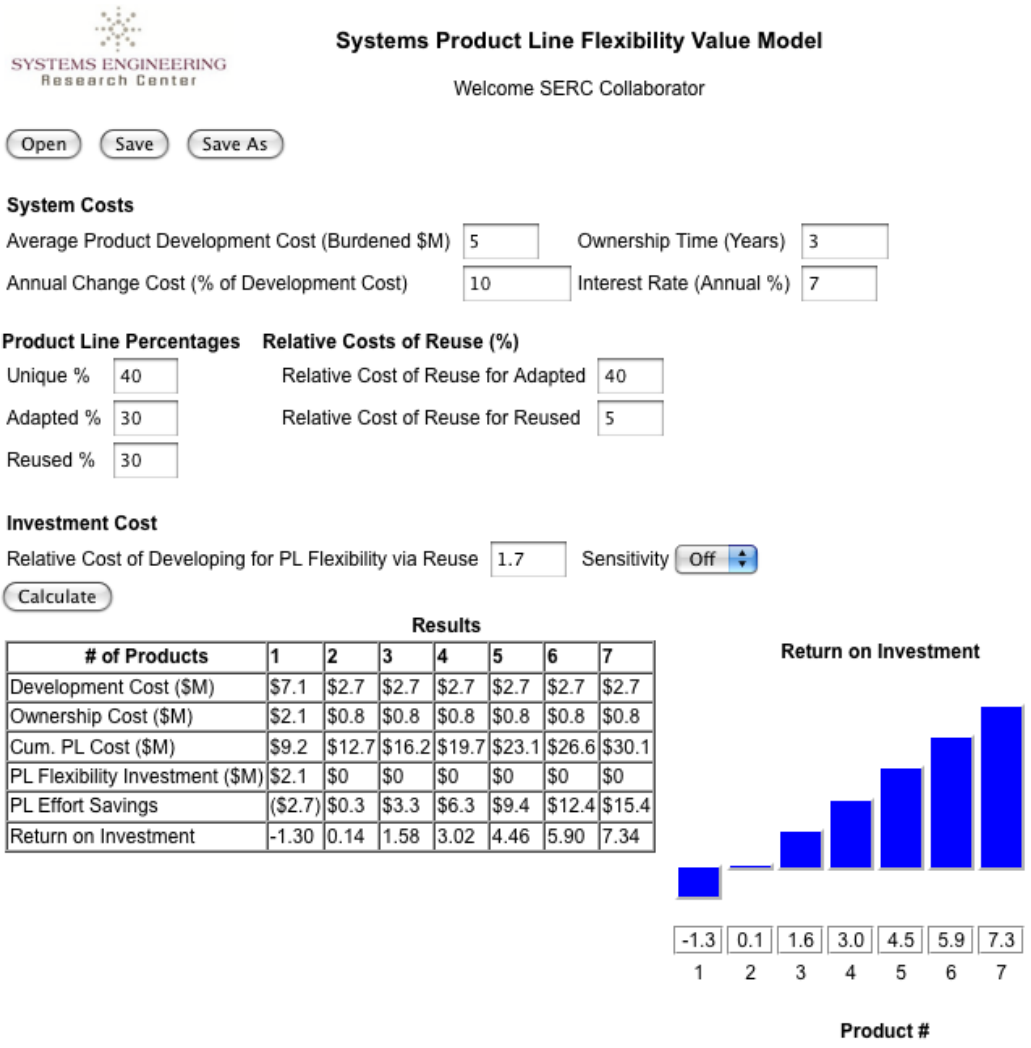


Figure 17: Product Line Flexibility TOC and ROI Results

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 18** 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 18** situation, the best level of investment in developing for reuse is an added 60%.

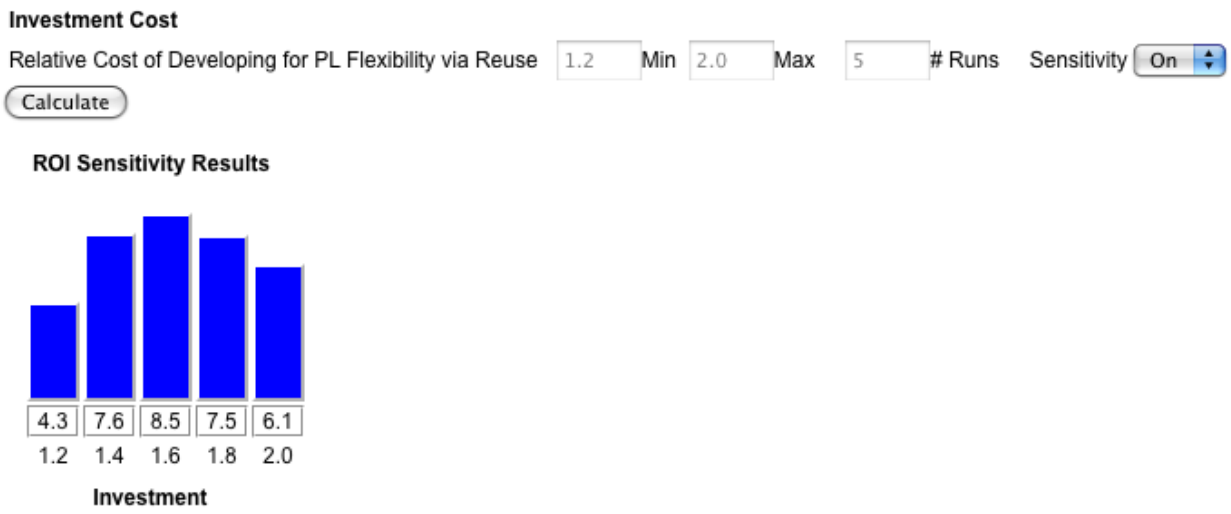


Figure 18: Example Sensitivity Analysis (ROI Only)

Other types of sensitivity analyses can be conducted. **Figure 19** shows example results of assessing the sensitivity of TOC across a range of product ownership durations.

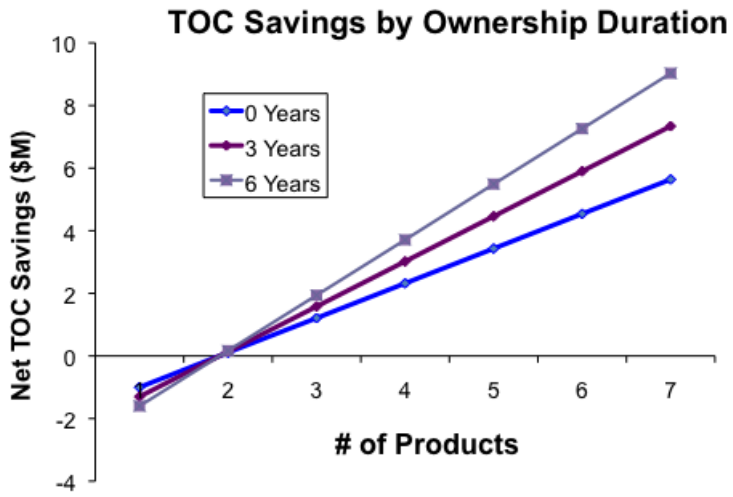


Figure 19: TOC Sensitivity by Ownership Duration Results

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 20**.

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.

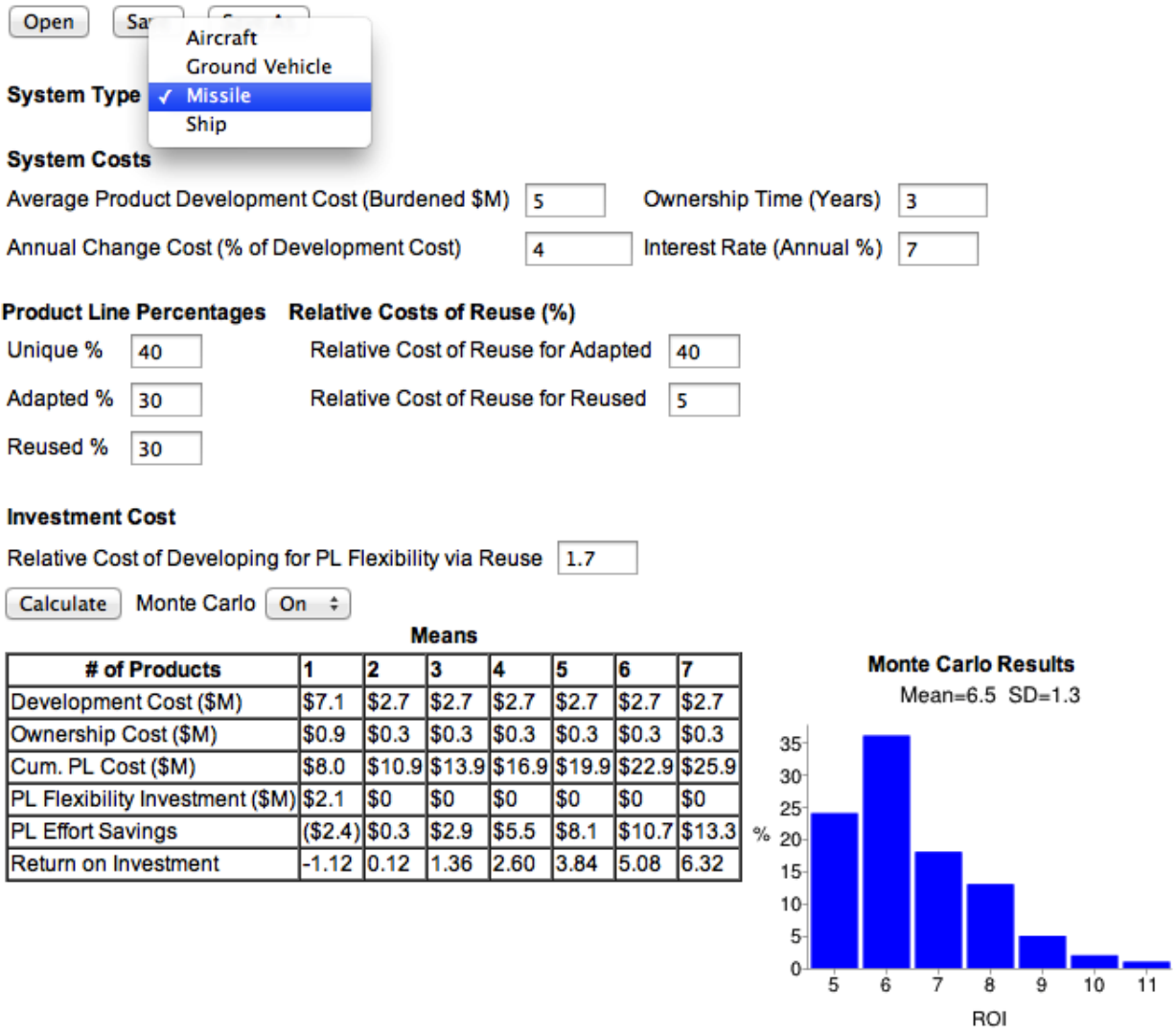


Figure 20: DoD Application Domain and Monte Carlo TOC-PL Results

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.