Affordable Systems: Balancing the Capability, Schedule, Flexibility, and Technical Debt Tradespace

Jo Ann Lane, Supannika Koolmanojwong, and Barry Boehm

University of Southern California

941 Bloom Walk

Los Angeles, CA 90089-0781

Copyright © 2013 by Lane, Koolmanojwong, and Boehm. Published and used by INCOSE with permission.

Abstract. With the increasing demands for affordable system capabilities that can be provided quickly to the user community, developers must explore a variety of options for identifying "satisficing" solutions. The system capability affordability tradespace must balance expedited systems engineering to reduce schedule and cost, encourage flexibility in architecture decisions to support future evolution of the system, and minimize technical debt that either results in later rework or adversely impacts future options. This paper shows how the University of Southern California (USC) Center for Systems and Software Engineering (CSSE) software and systems engineering cost models can be used in the analysis of this tradespace to show the range of options and the resulting consequences.

Introduction

The ultimate goal of today's systems and systems of systems (SoS) is to provide capabilities to the stakeholders and users of the systems. These capabilities range from "must-haves" to "nice-to-haves", often with disagreements among the stakeholders and users as to where each capability lays in this spectrum. There are many choices in developing and evolving systems to provide these desired capabilities, whether it is in the commercial space, Department of Defense (DoD) space, or other government space. These choices are typically related to development processes and product architecture decisions. Initial choices are often driven by business needs such as time to market, the desired level of performance of the capabilities, and available resources such as engineering expertise and funding. In addition, these choices often result in longer-term consequences that range from good (e.g., market share or future opportunities) to bad (e.g., missed market share, technical debt, or a failure to provide the desired capability). Other times, no particular attention is paid to these choices—they happen without much forethought, but still with the resulting longer-term consequences. Finally, there is often not an optimal set of choices, but rather the engineering team needs to evaluate the stakeholder needs and make trade decisions that sufficiently balance competing needs. This paper looks at the capability affordability tradespace of expediting systems engineering to reduce schedule and cost, encouraging flexibility in architecture decisions to support future evolution of the system, and technical debt that either results in later rework or adversely impacts future options. In addition, this paper shows how the University of Southern California (USC) Center for Systems and Software Engineering (CSSE) software and systems engineering cost models can be used in the analysis of this tradespace to show the range of options and the resulting consequences.

Background

The following discusses and characterizes each of the tradespace aspects considered in this paper.

Expedited Engineering Overview

Even though there is considerable evidence to the contrary, system sponsors and stakeholders continue to encourage developers to take shortcuts early in the development process in order to get system capabilities deployed quickly. These shortcuts are often done under the guise of agile processes with the thought that any resulting problems can be fixed later. However, the real goal is to get quality capabilities deployed quickly that do not overly constrain future evolution of the

system. So the ultimate goal is to expedite "system development" which includes the upfront engineering to design a system or system capability that meets the users need and well as build, test, and deploy that system or system capability in the shortest time possible. Ways to expedite development include:

- Minimal engineering/quick solutions with small, expert teams but that often result in increased technical debt
- Lean approaches that eliminate non-value adding activities, reduce wait times, and a "go-slow approach" at the start to establish a good foundation or architecture, well-defined interfaces, and relatively low complexity, then go fast during the build and test phases.

The following lists some of the key approaches for expediting engineering:

- Commercial-Off-the-Shelf (COTS) Products
- Investment in product-line architectures
- Reuse of existing systems/components
- Repurposing existing systems/components
- Value-stream focus (lean)
- Going fast in general (crisis response)
- Single purpose architecture

But all of these approaches are contingent upon *using the right people*!

Flexibility Overview

The goal of "flexibility" is to focus on developing a robust foundation for the system that will provide the ability to easily modify existing system capabilities as well as expand system capabilities to meet future needs or allow the system to easily interconnect with other systems to support cross cutting capabilities in systems of systems. The challenge in pursuing system flexibility is to balance flexibility and complexity. For example, performance issues may result if system tries to be "everything for everyone".

Technical Debt Overview

Technical debt is a term coined by Ward Cunningham to describe delayed technical work or rework that is incurred when shortcuts are taken. It is often the rework or unfulfilled outcomes/capabilities that result from insufficiently (or poorly) engineered or implemented solutions. In the case of expedited engineering, it can increase as shortcuts are employed to reduce schedule. Examples include insufficiently engineered architectures and the lack of focus on quality checks and reviews. In the case of flexibility, it may be related to unrecognized performance, security limitations, or excessive complexity that must be later addressed.

Initial Tradespace Analyses

Considerable research is currently being conducted in each of these areas (expedited engineering, flexibility, and technical debt). However, most of the research identified to date is looking at each of these areas in a stove-piped fashion, not as a set of features or outcomes to be balanced. The following describes recent research in each of these areas, some of which is currently being done through Stevens-USC Systems Engineering Research Center (SERC) tasks.

Expedited Engineering Analyses.

An important aspect of the system acquisition process is understanding how long it will take to provide the new needed system or system capability. Considerable work has been done in this area to better understand system/capability development schedules, with considerable contributions from parametric cost models for software-intensive systems that are based on historical data. However, many government organizations, including the Government Accountability Office (GAO), have pointed out how much money is wasted on unsuccessful system development programs that often fail to provide the necessary capabilities or if they do provide the capabilities, are late and cost much more than expected (U.S. Government Accountability Office 2011). As a result, the Department of Defense has realized that it is often no longer cost-effective to develop new systems (or system capabilities) using traditional processes and is encouraging researchers to find ways to expedite systems engineering and development. Currently, the engineering community is embracing agile and lean processes as well as system of systems engineering to provide new capabilities through the integration and enhancement of existing systems as ways to rapidly respond to immediate needs. This, though, can lead to single-point solutions that are not flexible enough to meet the next set of needs or incur technical debt due to extensive rework and maintenance costs.

Others are beginning to employ Kanban techniques to better manage work flows. To date, this has been primarily in the area of software development (Anderson 2010), but current research is investigating ways to employ Kanban for capability engineering work using a hierarchy of Kanbans for both systems and software engineering to improve work flow and minimize wait times once the proper foundations are in place for developing and evolving system or SoS capabilities.

A few of the larger, more success system development organizations are investing in infrastructure and product lines (Engineering Design Forum 2012; Royce 2004) so that they can respond quickly when new opportunities or world events occur that require immediate solutions. These suppliers realize a return on their investments when they have the needed flexibility built into their system infrastructure and have minimal technical debt due to the quality of the infrastructure core built in from the start.

To better understand opportunities to expedite systems engineering and software development, one can look at the cost factors in the associated USC CSSE cost models and their influence on productivity. Figures 1 and 2 illustrate these factors for the systems engineering cost model, COSYSMO, and the software development cost model, COCOMO, respectively.

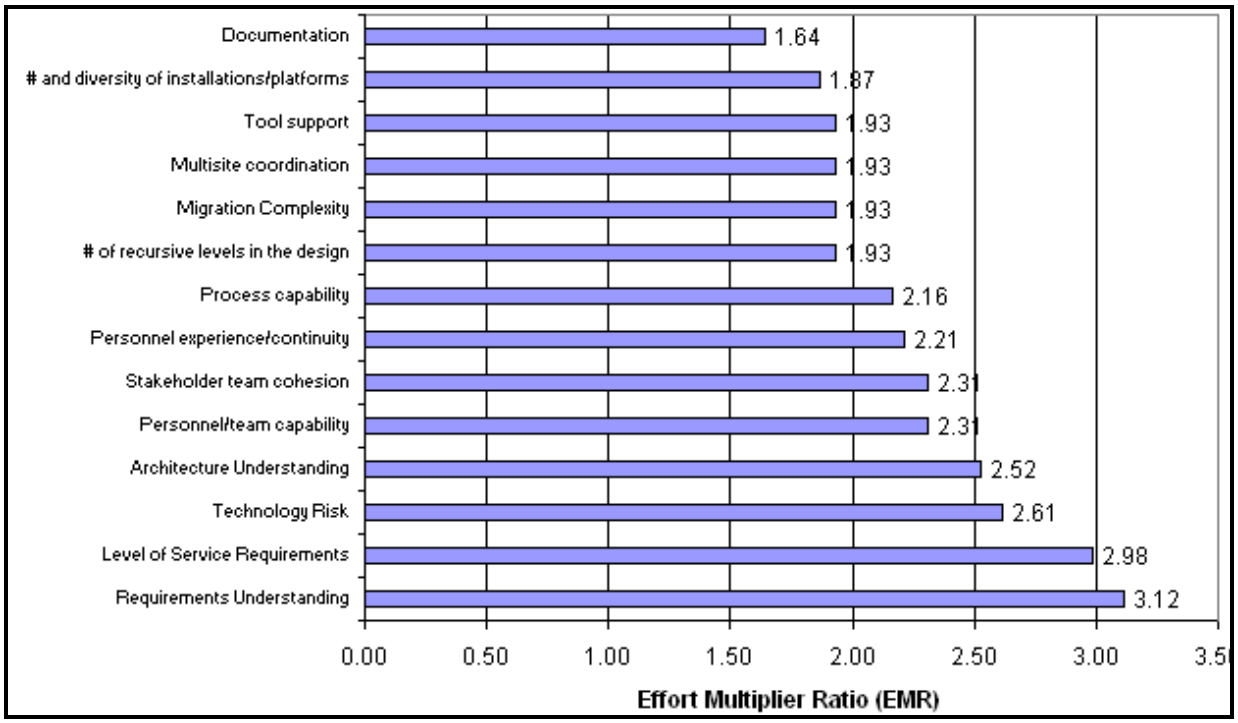

Figure 1. COSYSMO Effort Multiplier Ratio (Valerdi 2005)

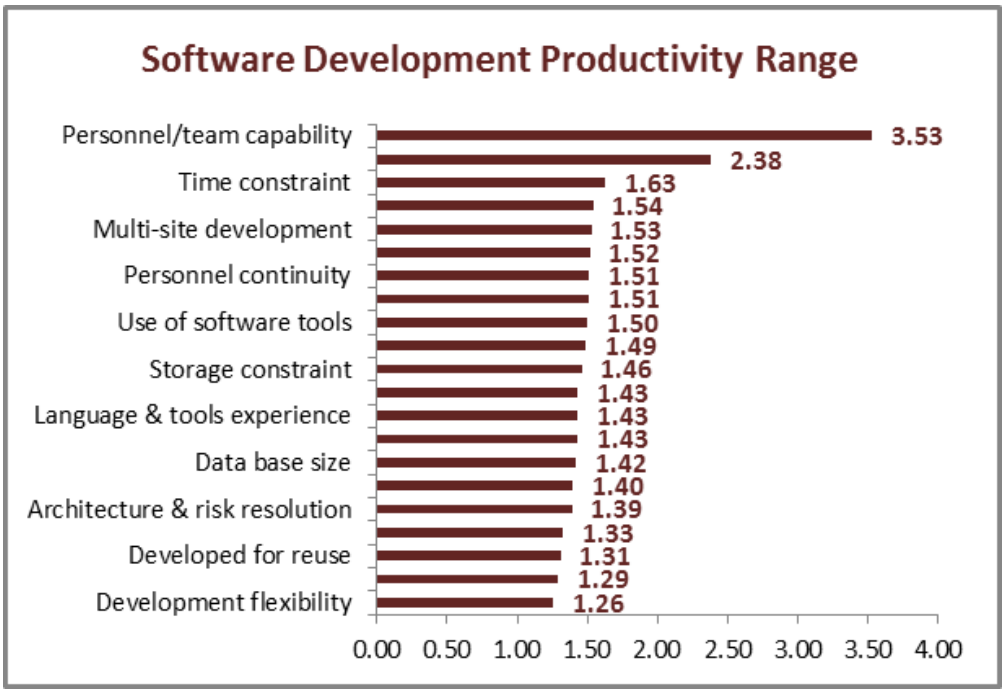

Figure 2. COCOMO software productivity ranges (Boehm et al. 2000)

The greatest gains in effort savings and schedule can be made by focusing engineering improvements in the areas of the high-influence cost factors.

System Flexibility Analyses

Without a certain level of flexibility, systems can quickly become obsolete as technology and stakeholder/mission needs change. However, if too much effort is spend on developing the most flexible systems, it may be at the expense of delivery expediency and simplicity of the system core, resulting in longer development schedules for the first incarnation of the system and poorer operational performance due to the need to sacrifice single capability performance for the ability to perform multiple capabilities (or perform a single capability in multiple environments). The following are some key approaches for enabling system flexibility:

- Employ open architectures
- Design for reuse
- Develop/use product lines
- Standard interfaces, protocols, services, data.

Current SERC research in this area is focusing on evaluation of total ownership costs and options analysis. Preliminary total ownership cost analysis, illustrated in figure 3, has shown potential cost savings when upfront investments are made in architecture.

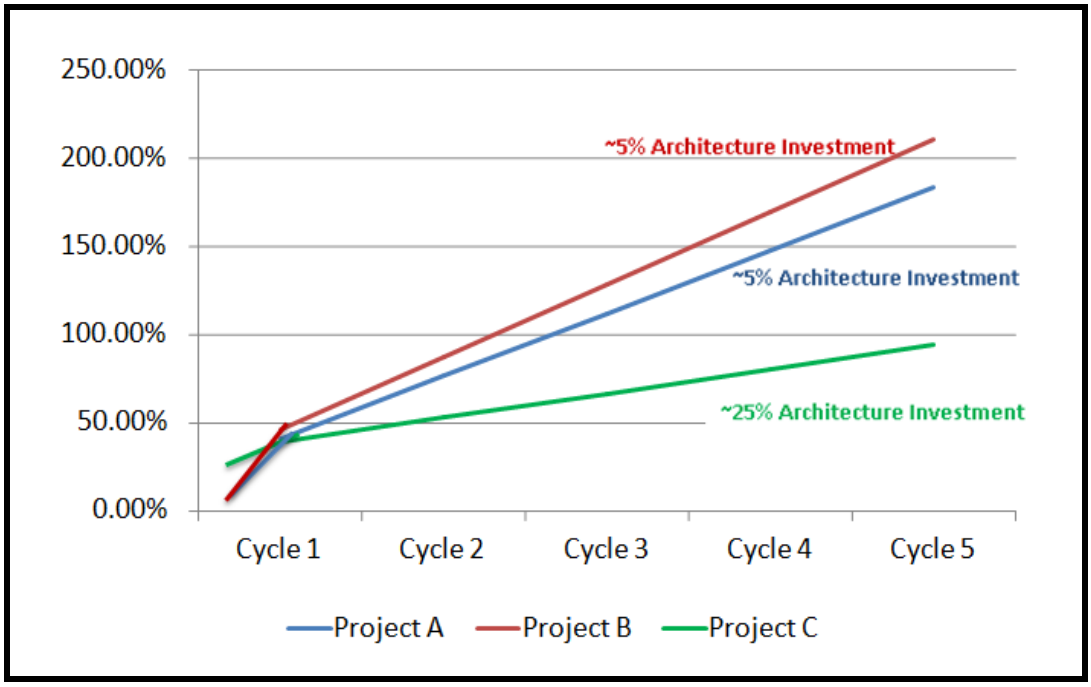

Figure 3. TOC's for 3 Projects Relative to Baseline Costs (Boehm et al. 2011)

However, as shown in table 1, many of the flexibility architecture strategies may present conflicts with other desired system characteristics such as performance, human controllability, increased development schedules and costs, and security, to name a few.

Table 1: Architecture-Based Attribute Trades with Respect to Flexibility

Architecture Strategy	Synergies	Conflicts
High module cohesion Low module coupling	• Interoperability • Reliability	• Impacts high performance achieved via tight coupling
Service-oriented architecture	• Composability • Usability • Testability	• Impacts high performance achieved via tight coupling • Requires additional infrastructure resulting in added costs/schedule
Autonomous adaptive systems	• Affordability via task automation • Response time	• Excess autonomy reduces human controllability • Requires additional analysis and testing to identify emergent behaviors
Modularization around sources of change	• Interoperability • Usability • Reliability • Availability	• Extra time on critical path of rapid fielding
Multi-layered architecture	• Reliability • Availability	• Lower performance due to layer traversal overhead • Potential integration issues with other architecture styles
Many built-in options, entry points	• Functionality • Accessibility	• Reduced usability via options proliferation • Harder to secure
User programmability	• Usability • Mission effectiveness	• Full programmability causes reliability, compatibility, interoperability, safety, security risks
Spare/expandable capacity	• Performance • Reliability	• Added cost • Usefulness may be limited by rapidly changing technologies
Product line architecture Reusable components	• Cost • Schedule • Reliability	• Some loss of performance vs. optimized stovepipes

At the SoS level, there has been a considerable focus on how enable systems to quickly connect to perform desired cross-cutting mission capabilities, then return to their normal single-system missions. This has become more important over time as systems come together at many levels (e.g., joint services and international coalitions) and as these systems participate in multiple SoSs. Some (USAF SAB 2005) advocate for the migration to a set of standard convergent protocols to enable the needed interconnectivity and others (Lane 2009) are developing techniques to evaluate and improve interoperability across a set of systems that must interoperate as an SoS.

Managing Technical Debt Analyses

Recent research by Steven McConnell (McConnell 2008), Practical Systems and Software Measurement (http://www.psmsc.com/) affiliates, and others are focusing on a concept referred to as "technical debt" and ways to better manage it in the development of systems and software. Ward Cunningham (Cunningham 1992) first coined the term and further explained it in his YouTube video (Cunningham 2009). Technical debt refers to delayed technical work or rework that is incurred when shortcuts are taken. The following are some common causes of technical debt:

- Pressures to compress schedule
- Lack of requirements understanding
- Lack of system understanding
- Inflexible architectures/software
- Overly complex design/implementation
- Delayed defect resolution
- Inadequate testing
- Lack of current documentation
- Parallel development in isolation
- Delayed refactoring

Some technical debt is reasonable given time-to-market or other urgent constraints. But over the long term, if the system is to be sustained, the technical debt must be paid back, often with interest (i.e., it is more expensive to fix than it is to do it right the first time). Deferred long-term technical debt can result in fragile, error-prone systems that take excessive time to modify, with more time spent on maintaining existing capabilities than adding or improving capabilities.

Expedited Engineering vs. Valuing Flexibility

The focus of this research effort used the COCOMO suite of cost models illustrated in Figure 4 to compute estimated effort and schedule for a given project with different project drivers such as expedited engineering and valuing flexibility.

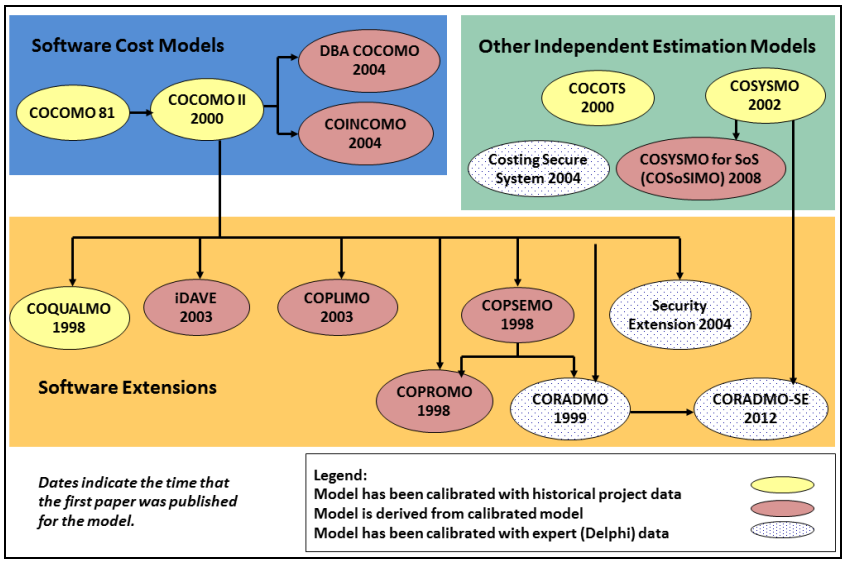

Figure 4: Overview of USC CSSE Cost Models.

For the first case, the project drivers characterized expedited engineering. For the second case, the project driver characterized the development of a flexible product. This comparison illustrates how the USC CSSE cost models can be used to calculate the associated engineering effort and schedule for each of these cases. Key to this analysis is the most recent addition to the COCOMO suite of cost models, CORADMO-SE , that can be used to calculate the savings in schedule for systems engineering. Table 2 shows the CORADMO-SE factors and associated weights. This model calculates a "rapid" factor that can be applied to the Constructive Systems Engineering Cost Model (COSYSMO) estimated schedule.

To use this model, one computes the expected effort and schedule using COSYSMO, then uses CORADMO-SE to determine the expedited factor which is then applied by multiplying the COSYSMO schedule (a cube-root function of effort) by the CORADMO-SE factor. A value of 1 does not change the COSYSMO estimated schedule, a value less than 1 decreases the estimated schedule, and a value greater than 1 increases the estimated schedule.

Table 2: CORADMO-SE Cost and Schedule Factors.

Accelerators/Ratings	Very Low	Low	Nominal	High	Very High	Extra High
Product Factor: Multipliers	**1.09**	**1.05**	**1.0**	**0.96**	**0.92**	**0.87**
Simplicity	Extremely complex	Highly complex	Mod. complex	Moderately simple	Highly simple	Extremely simple
Element Reuse	None (0%)	Minimal (15%)	Some (30%)	Moderate (50%)	Considerate (70%)	Extensive (90%)
Low-Priority Deferrals	Never	Rarely	Sometimes	Often	Usually	Anytime
Models vs Documents	None (0%)	Minimal (15%)	Some (30%)	Moderate (50%)	Considerate (70%)	Extensive (90%)
Key Technology Maturity	>0 TRL 1,2 or >1 TRL 3	1 TRL 3 or > 1 TRL 4	1 TRL 4 or > 2 TRL 5	1-2 TRL 5 or >2 TRL 6	1-2 TRL 6	All > TRL 7
Process Factor: Multipliers	**1.09**	**1.05**	**1.0**	**0.96**	**0.92**	**0.87**
Concurrent Operational Concept, Requirements, Architecture, V&V	Highly sequential	Mostly sequential	2 artifacts mostly concurrent	3 artifacts mostly concurrent	All artifacts mostly concurrent	Fully concurrent
Process Streamlining	Heavily bureaucratic	Largely bureaucratic	Conservative bureaucratic	Moderate streamline	Mostly streamlined	Fully streamlined
General SE tool support CIM (Coverage, Integration, Maturity)	Simple tools, weak integration	Minimal CIM	Some CIM	Moderate CIM	Considerable CIM	Extensive CIM
Project Factors: Multipliers	**1.08**	**1.04**	**1.0**	**0.96**	**0.93**	**0.9**
Project size (peak # of personnel)	Over 300	Over 100	Over 30	Over 10	Over 3	≤ 3
Collaboration support	Globally distributed weak comm. , data sharing	Nationally distributed, some sharing	Regionally distributed, moderate sharing	Metro-area distributed, good sharing	Simple campus, strong sharing	Largely collocated, Very strong sharing
Single-domain MMPTs (Models, Methods, Processes, Tools)	Simple MMPTS, weak integration	Minimal CIM	Some CIM	Moderate CIM	Considerable CIM	Extensive CIM
Multi-domain MMPTs	Simple; weak integration	Minimal CIM	Some CIM or not needed	Moderate CIM	Considerable CIM	Extensive CIM
People Factors: Multipliers	**1.13**	**1.06**	**1.0**	**0.94**	**0.89**	**0.84**
General SE KSAs (Knowledge, Skills, Agility)	Weak KSAs	Some KSAs	Moderate KSAs	Good KSAs	Strong KSAs	Very strong KSAs
Single-Domain KSAs	Weak	Some	Moderate	Good	Strong	Very strong
Multi-Domain KSAs	Weak	Some	Moderate or not needed	Good	Strong	Very strong
Team Compatibility	Very difficult interactions	Some difficult interactions	Basically cooperative interactions	Largely cooperative	Highly cooperative	Seamless interactions
Risk Acceptance Factor: Multipliers	**1.13**	**1.06**	**1.0**	**0.94**	**0.89**	**0.84**
	Highly risk-averse	Partly risk-averse	Balanced risk aversion, accept	Moderately risk-accepting	Considerably risk-accepting	Strongly risk-accepting

The example we use to illustrate the comparison of expedited systems engineering and valuing flexibility is an engineering division in a diversified company that focuses on defense applications. Assume that for the project of interest, there is a team of 20 systems engineers that are doing the up-front engineering for a new system using their standard sequential processes that are based on the Vee model. Their current tasks are to refine the operational concepts and requirements for the system, then develop a system architecture that satisfies the requirements. In addition, the defense customer has requested that they use more rapid processes in order to expedite the delivery of the system. Figure 5 highlights their current process assessment using the CORADMO-SE factors.

Accelerators/Ratings	Very Low	Low	Nominal	High	Very High	Extra High
Product Factor: Multipliers	1.09	1.05	1.0	0.96	0.92	0.87
Simplicity	Extremely complex	Highly complex	Mod. complex	Moderately simple	Highly simple	Extremely simple
Element Reuse	None (0%)	Minimal (15%)	Some (30%)	Moderate (50%)	Considerate (70%)	Extensive (90%)
Low-Priority Deferrals	Never	Rarely	Sometimes	Often	Usually	Anytime
Models vs Documents	None (0%)	Minimal (15%)	Some (30%)	Moderate (50%)	Considerate (70%)	Extensive (90%)
Key Technology Maturity	>2 TRL 1-2 or >1 TRL 3	1 TRL 3 or >1 TRL 4	1 TRL 4 or >1 TRL 5	1-2 TRL 5 or >1 TRL 6	1-2 TRL 6	All > TRL 7
Process Factor: Multipliers	1.09	1.05	1.0	0.96	0.92	0.87
Concurrent Operational Concept, Requirements, Architecture, V&V	Highly sequential	Mostly sequential	2 artifacts mostly concurrent	3 artifacts mostly concurrent	All artifacts mostly concurrent	Fully concurrent
Process Streamlining	Heavily bureaucratic	Largely bureaucratic	Conservative bureaucratic	Moderate streamline	Mostly streamlined	Fully streamlined
General SE tool support CIM (Coverage, Integration, Maturity)	Simple tools, weak integration	Minimal CIM	Some CIM	Moderate CIM	Considerable CIM	Extensive CIM
Project Factors: Multipliers	1.08	1.04	1.0	0.96	0.93	0.9
Project size (peak # of personnel)	Over 300	Over 100	Over 30	Over 10	Over 3	≤ 3
Collaboration support	Globally distributed weak comm. , data sharing	Nationally distributed, some sharing	Regionally distributed, moderate sharing	Metro-area distributed, good sharing	Simple campus, strong sharing	Largely collocated, Very strong sharing
Single-domain MMPTs (Models, Methods, Processes, Tools)	Simple MMPTS, weak integration	Minimal CIM	Some CIM	Moderate CIM	Considerable CIM	Extensive CIM
Multi-domain MMPTs	Simple, weak integration	Minimal CIM	Some CIM or not needed	Moderate CIM	Considerable CIM	Extensive CIM
People Factors: Multipliers	1.13	1.06	1.0	0.94	0.89	0.84
General SE KSAs (Knowledge, Skills, Agility)	Weak KSAs	Some KSAs	Moderate KSAs	Good KSAs	Strong KSAs	Very strong KSAs
Single-Domain KSAs	Weak	Some	Moderate	Good	Strong	Very strong
Multi-Domain KSAs	Weak	Some	Moderate or not needed	Good	Strong	Very strong
Team Compatibility	Very difficult interactions	Some difficult interactions	Basically cooperative interactions	Largely cooperative	Highly cooperative	Seamless interactions
Risk Acceptance Factor: Multipliers	1.13	1.06	1.0	0.94	0.89	0.84
	Highly risk-averse	Partly risk-averse	Balanced risk aversion, accept	Moderately risk-accepting	Considerably risk-accepting	Strongly risk-accepting

Figure 5: Case study CORADMO assessment of current process.

To determine the "rapid" factor to apply to the COSYSMO, one first determines the product, process, project, people, and risk acceptance factors. For each factor, an organization highlights their sub-factor assessments in the CORADMO-SE table, as shown in Figure 5. Then, for each factor, the evaluator "averages" the values. This can be an "average" calculation or it can be subjectively adjusted by the evaluator. To adjust the average value, the evaluator weights the sub-factors based on his/her assessment of their importance to the organization and project. For the example presented here, the following factors are suggested by the CORADMO-SE table:

Product:	1.05
Process:	1.05
Project:	0.95
People:	0.97
Risk acceptance:	1.00

The next step is to multiply these factors together, resulting in a CORADMO-SE factor of 1.02, which is then multiplied with the calculated COSYSMO schedule. This is the current organization baseline expedited factor upon which the project would like to improve, i.e., reduce to a value below 1.0, the nominal expedited factor.

Approach 1: Expedite SE through Concurrent Engineering

To reduce schedule, project management decides to implement concurrent engineering of systems engineering work products. By itself, this process change might reasonably change the process factor from 1.05 to 0.87, resulting in a composite expedited factor of 0.96. However, making this single process change can impact some of the other CORADMO-SE factors. For example, introducing some new tools to support concurrent engineering while continuing to use other more traditional SE tools will have an "slow-down" effect: with the new tools, SE toolset is not as well-integrated as it was, there is additional time required to set up the new tools and train the users on their use. There is also a learning curve for the SE engineers with respect to the concurrent engineering. And finally, team compatibility can take a hit if management continues to use their more traditional processes with the concurrent engineering SE processes. So, making these adjustments to the CORADMO-SE parameters (General Tool Support H→N, General SE KSAs H→L, and Team Compatibility H→L), results in a CORADMO-SE factor of 1.05 (as compared to the initial baseline of 1.02). The CORADMO-SE model shows that changing the process can initially slow down the project until the new tools and processes are integrated and the team becomes familiar with them.

Approach 2: Value Flexibility

If the engineering project decides to value flexibility over "expedited engineering", we get some different results. With this approach, instead of transitioning quickly from a highly sequential process to a fully concurrent engineering process, the project decides to focus on valuing product flexibility and streamlining their largely bureaucratic processes with some concurrency. So, in the CORADMO-SE framework the following changes are made with respect to the baseline:

Element reuse:	None→Moderate
Low-priority deferrals:	Never→Usual
Models vs. documents:	Minimal→Moderate
Concurrency:	Highly sequential→Nominal
Process streamlining:	Largely bureaucratic→Moderate streamlining

The resulting CORADMO-SE factor is 0.86.

Analysis of Two Approaches

The resulting "valuing product flexibility" CORADMO-SE factor (0.86) is much better than the 1.05 value that obtained when a drastic change from a highly sequential process to a fully concurrently engineered process was proposed, requiring re-tooling and a steeper learning curve. Over time, as the engineering team becomes more familiar with concurrent engineering processes, this will change and additional reductions in schedule will be realized. However, CORADMO-SE shows that by moving a little more slowly with process changes (e.g., some concurrency instead of full concurrency, some process streamlining, and focusing more on engineering models rather than documentation) initial results can be much more positive.

This current analysis has not yet addressed the issue of technical debt that sometimes occurs when teams overly focus on expedited engineering, taking shortcuts that impact longer term maintenance, rework, and evolvability of the system. Preliminary analysis using the COQUALMO cost model for software development shows that by valuing flexibility and not shortcutting reviews, schedule can be further reduced and the overall remaining defects are considerably smaller. One example comparing expedited vs. valuing flexibility shows that for 250,000 lines of code, when valuing flexibility fewer defects are introduced during

development and only a small percentage remain at the end of development.

Conclusions and Future Work

To summarize, with the increasing demands for affordable system capabilities that can be provided quickly to the user community, developers must explore a variety of options for identifying "satisficing" solutions. The system capability affordability tradespace must balance expedited systems engineering to reduce schedule and cost, encourage flexibility in architecture decisions to support future evolution of the system, and minimize technical debt that either results in later rework or adversely impacts future options. We showed how the USC CSSE software and systems engineering cost models can be used in the analysis of this tradespace to show the range of options and the resulting consequences. Preliminary analysis also shows that valuing flexibility and investing in system and software architectures can often provide more expedient and flexible solutions than by just focusing on expedited processes.

The next steps are to continue to investigate ways to expedite systems engineering through lean Kanban and other approaches and to further explore tradespace options that include technical debt through additional case studies. As part of this future work, additional analysis using the USC CSSE cost models will be conducted and models refined to better support tradespace analyses as well as predict cost, schedule, and technical debt.

References

Anderson, D. 2010. Kanban: Successful Evolutionary Change for Your Technology Business. Blue Hole Press, Sequim, WA.

Boehm, B., C. Abts, A. Brown, S. Chulani, B. Clark, E. Horowitz, R. Madachy, D. Reifer, and B. Steece. 2000. Software cost estimation with COCOMO II. Upper Saddle River: Prentice Hall

B. Boehm, J. Lane, and R. Madachy. 2011. Total Ownership Cost Models for Valuing System Flexibility, Proceedings of the Conference on Systems Engineering Research, 14-16 April, Los Angeles, CA.

Boehm, B., J. Lane, S. Koolmanojwong, and R. Turner. 2010. "Architected Agile Solutions for Software-Reliant Systems" in Agile Software Development Current Research and Future Directions. Springer; 1st Edition.

Boehm, B., J. Lane, S. Koolmanojwong, R. Turner. 2010. Architected Agile Solutions for Software-Reliant Systems," Proceedings of the International Council on Systems Engineering Symposium, 12-15 July, Chicago, IL.

Boehm, B., W. Brown, R. Madachy, Y. Yang. 2004. "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.

Cunningham, W. 1992. The WyCash portfolio management system, Experience Report. Oopsla'92.

Cunningham, W. 2009. Debt metaphor. http://www.youtube.com/watch?v=pqeJFYwnkjE.

Engineering Design Forum. 2012. http://www.phoenix-int.com/forum_2012/index.html.

Highsmith, J. 2000. Adaptive software development: A collaborative approach to managing complex systems, New York: Dorset House Publishing.

Ingold, D., Boehm, B., Koolmanojwong, S., and Lane, J. A Model for Estimating Agile Project Acceleration, USC CSSE Technical Report, 2013.

Koolmanojwong, S. and B. Boehm. 2010. "The Incremental Commitment Model Process Patterns for Rapid-Fielding Projects," in Proceedings of International Conference on Software Process 2010, July in Paderborn, Germany.

Lane, J. 2009. Cost Model Extensions to Support Systems Engineering Cost Estimation for Complex Systems and Systems of Systems, Proceedings of the Seventh Conference on Systems Engineering Research.

Lane, J. and R. Valerdi. 2011. System Interoperability Influence of System on Systems Engineering Effort, Proceedings of the Conference on Systems Engineering Research, 14-16 April, Los Angeles, CA.

McConnell, S. 2008. Best Practices White Paper: Managing Technical Debt. http://www.construx.com/Page.aspx?cid=2801.

Royce, W. 2004. Software Project Management: A Unified Framework. Addison-Wesley.

United States Air Force (USAF) Scientific Advisory Board (SAB). 2005. Report on system-of-systems engineering for Air Force capability development; Public Release SAB-TR-05-04.

U.S. Government Accountability Office. 2011. Report - WEAPONS ACQUISITION REFORM: Actions Needed to Address Systems Engineering and Developmental Testing Challenges, GAO-11-806, Sep 19, 2011.

Valerdi, R. 2005. Constructive systems engineering cost model. PhD. Dissertation, University of Southern California.

Bibliography

Dr. Jo Ann Lane is Co-Director of the Center for Systems and Software Engineering as well as a Research Assistant Professor in the Daniel J. Epstein Department of Industrial and Systems Engineering at The University of Southern California. She was awarded the International Council on Systems Engineering (INCOSE) Foundation/Stevens Doctoral Award for promising research in Systems Engineering and Integration. Prior to her current work, she was a key technical member of Science Applications International Corporation's Software and Systems Integration Group for over 20 years, responsible for the development and integration of software-intensive systems and systems of systems.

Dr. Supannika Koolmanojwong is a full-time lecturer and a researcher at the University of Southern California Center for Systems and Software Engineering. Her primary research area focuses on Systems and Software Process Modeling, Software Process Improvement, Software Process Quality Assurance, Software Metrics and Measurement, Agile and Lean Software Development and Expediting Systems Engineering.

Dr. Barry Boehm is the TRW Professor in the USC Computer Sciences and Industrial and Systems Engineering Departments. He is also the Director of Research of the DoD-Stevens-USC Systems Engineering Research Center, and the founding Director Emeritus of the USC Center for Systems and Software Engineering. He was director of DARPA-ISTO 1989-92, at TRW 1973-89, at Rand Corporation 1959-73, and at General Dynamics 1955-59. He is a Fellow of the primary professional societies in computing (ACM), aerospace (AIAA), electronics (IEEE), and systems engineering (INCOSE), a member of the U.S. National Academy of Engineering, and the 2010 recipient of the IEEE Simon Ramo Medal for exceptional achievement in systems engineering and systems science.