Software and IT-CAST
Proceedings
22-24 August 2017

Lockheed Martin Global Vision Center
2121 Crystal Drive
Arlington, VA 22202
<table>
<thead>
<tr>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Title</td>
<td>1</td>
</tr>
<tr>
<td>Table of Contents</td>
<td>2</td>
</tr>
<tr>
<td>Overview</td>
<td>3</td>
</tr>
<tr>
<td>Agenda</td>
<td>4</td>
</tr>
<tr>
<td>Keynote Biography</td>
<td>6</td>
</tr>
<tr>
<td>Abstracts</td>
<td>7</td>
</tr>
<tr>
<td>Keynote Address</td>
<td>14</td>
</tr>
</tbody>
</table>

**Presentations**

1) Agile and GAO Cost Estimating Best Practices, Karen Richey (GAO) 22

2) How Should We Estimate Agile Projects and Measure Progress to Plan, Thomas Coonce (IDA) & Glen Alleman (Niwot Ridge) 45

3) Software Size Growth, Marc Russo (NCCA) 94

4) Adapting a classic Independent Cost Estimation [Process] for Agile and DevOPS, David Seaver (NSA) 128

5) Assessing ERP Cost, Schedule and Size Growth, Haset Gebre-Mariam NCCA) & Rob Williams (Herren Associates) 151

6) Objective SLOC: An Alternative Method to Sizing Software Development Efforts, Andrew Kicinski (Integrity Applications Incorporated - NRO) 194

7) Software Cost Estimation Meets Software Diversity, Barry Boehm (USC) 223


9) Apples and Oranges: a Presentation and Analysis of Results of Cloud Cost Calculators and Rate Cards, Daniel J Harper (MITRE) 299

10) Rosetta Stone for Software Sizing, Victor Fuster (QSM) & Taylor Putnam-Majarian (QSM) 321

11) SRDR Unified Review Function (SURF): Deeper Focus on Software Data Quality, Nick Lanham and Marc Russo (NCCA) 350

12) Expanding the Horizons of Software Cost Estimation, Jairus M Hihn (NASA JPL) 372

13) Why Does Software Cost So Much? Towards a Causal Model, Bob Stoddard and Mike Konrad (Software Engineering Institute) 392

14) Reliable Non-Design, Code, Test, and Integration Cost Relationships, Brittany Staley and Jeremy Goucher (Herren Associates) 408

15) Introduction to Software Obsolescence Cost Analysis Framework, Sanathanan Rajagopal (QinetiQ, United Kingdom) 429

16) Cost Assessment Data Enterprise (CADE) – Overview & Software Initiatives, Daron D Fullwood (OSD CAPE) Ranae Woods, SES (AFCAA) 470

17) COCOMO III Workshop: Implementing a New Driver for Software Security, Barry Boehm and Brad Clark (USC) 493
OVERVIEW

The Naval Center for Cost Analysis (NCCA) and the National Geospatial-Intelligence Agency Corporate Assessment and Program Evaluation (NGA CAPE) present the Software and Information Technology Cost Analysis Solutions Team (Software and IT-CAST) meeting from August 22-24, 2017 at the Lockheed Martin Global Vision Center in Crystal City, Virginia. This meeting is organized with the support of US Army ARDEC, Lockheed Martin, and DOD cost agencies.

The Software and IT-CAST meeting is a venue to build coalitions with government and industry, to exchange cost data, share lessons learned, and establish best practices concerning software and information technology cost estimation. Topics include

- Software and Information Technology Cost Estimation
- Software Cost Data Collection and Analysis Best Practices
- Project Cost and Schedule Growth
- Measurements for Agile Software Development
- Measurements for Software Maintenance
- Measurements for Cloud Computing and Cyber Security

The program includes presentations, workshops, and contractor one-on-one discussions. Presentations and workshops are open to all attendees. Contractor one-on-one discussions are restricted to federal employees who have registered.

COMMITTEE

General Chair:
Vjosa Dreshaj (NGA CAPE)
Wilson Rosa (NCCA)
Haset Gebre-Mariam (NCCA)

Program Co-Chairs:
Corinne Wallshein (NCCA)
Corey Boone (NCCA)
Lyle Patashnick (NGA CAPE)

Venue Co-Chair:
Gregory Niemann (Lockheed Martin)

Portal Design Co-Chair:
Don Clarke (NCCA)

ATTENDANCE

General sessions (presentations and workshops) are open to all attendees.
Contractor discussions are restricted to federal government employees who have registered.
## Software and IT-CAST Agenda
### 22-24 August 2017
Lockheed Martin Global Vision Center
2121 Crystal Drive, Crystal City, Arlington, VA 22202

<table>
<thead>
<tr>
<th>Time</th>
<th>Session</th>
<th>Speaker &amp; Details</th>
<th>Location</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Tuesday, August 22, 2017 – General Session (Open to All)</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0730 – 0800</td>
<td>Registration</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0800 – 0810</td>
<td>Opening Remarks</td>
<td>Jennifer Rose (NGA CAPE)</td>
<td>Auditorium</td>
</tr>
<tr>
<td>0810 – 0840</td>
<td>Keynote Address</td>
<td>John Zangardi (Acting DoD CIO)</td>
<td>Auditorium</td>
</tr>
<tr>
<td>0920 – 0950</td>
<td>How Should We Estimate Agile Projects and Measure Progress to Plan</td>
<td>Thomas Coonce (IDA) Glen Alleman (Niwot Ridge)</td>
<td>Auditorium</td>
</tr>
<tr>
<td>0950 – 1005</td>
<td>Break</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1005 – 1035</td>
<td>Software Size Growth</td>
<td>Marc Russo (NCCA)</td>
<td>Auditorium</td>
</tr>
<tr>
<td>1115 – 1145</td>
<td>Assessing ERP Cost, Schedule and Size Growth</td>
<td>Haset Gebre-Mariam (NCCA) Rob Williams (Herren Associates)</td>
<td>Auditorium</td>
</tr>
<tr>
<td>1145 – 1300</td>
<td>Lunch</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1300 – 1330</td>
<td>Objective SLOC: An Alternative Method to Sizing Software Development Efforts</td>
<td>Andrew Kicinski (Integrity Applications Incorporated - NRO)</td>
<td>Auditorium</td>
</tr>
<tr>
<td>1405 – 1415</td>
<td>Break</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1415 – 1700</td>
<td>COCOMO III Workshop: Implementing a New Driver for Software Security</td>
<td>Barry Boehm and Brad Clark (USC)</td>
<td>2nd Floor, GVC-A</td>
</tr>
<tr>
<td><strong>Tuesday, August 22, 2017 – Contractor Discussions (Restricted)</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1415 – 1600</td>
<td>Northrop Grumman One-on-One</td>
<td>John Sautter (Northrop Grumman)</td>
<td>Auditorium</td>
</tr>
</tbody>
</table>
### Wednesday, August 23, 2017 – General Session (Open to All)

<table>
<thead>
<tr>
<th>Time</th>
<th>Event</th>
<th>Speaker(s)</th>
<th>Location</th>
</tr>
</thead>
<tbody>
<tr>
<td>0730 – 0800</td>
<td>Registration</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0800 – 0810</td>
<td>Opening Remarks</td>
<td>David Cashin (NCCA)</td>
<td>Auditorium</td>
</tr>
<tr>
<td>0845 – 0915</td>
<td>Apples and Oranges: a Presentation and Analysis of Results of Cloud Cost Calculators and Rate Cards</td>
<td>Daniel J Harper (MITRE)</td>
<td>Auditorium</td>
</tr>
<tr>
<td>0950 – 1000</td>
<td>Break</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1000 – 1030</td>
<td>SRDR Unified Review Function (SURF): Deeper Focus on Software Data Quality</td>
<td>Nick Lanham (NCCA) Marc Russo (NCCA)</td>
<td>Auditorium</td>
</tr>
<tr>
<td>1035 – 1105</td>
<td>Expanding the Horizons of Software Cost Estimation</td>
<td>Jairus M Hihn (NASA JPL)</td>
<td>Auditorium</td>
</tr>
<tr>
<td>1110 – 1140</td>
<td>Why Does Software Cost So Much? Towards a Causal Model</td>
<td>Bob Stoddard and Mike Konrad (Software Engineering Institute)</td>
<td>Auditorium</td>
</tr>
<tr>
<td>1140 – 1245</td>
<td>Lunch</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1245 – 1315</td>
<td>Reliable Non-Design, Code, Test, and Integration Cost Relationships</td>
<td>Brittany Staley and Jeremy Goucher (Herren Associates)</td>
<td>Auditorium</td>
</tr>
<tr>
<td>1320 – 1355</td>
<td>Introduction to Software Obsolescence Cost Analysis Framework</td>
<td>Sanathanan Rajagopal (QinetiQ, United Kingdom)</td>
<td>Auditorium</td>
</tr>
<tr>
<td>1355 – 1425</td>
<td>Cost Assessment Data Enterprise (CADE) – Overview &amp; Software Initiatives</td>
<td>Daron D Fullwood (OSD CAPE) Ranae Woods, SES (AFCAA)</td>
<td>Auditorium</td>
</tr>
<tr>
<td>1425 – 1435</td>
<td>Break</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1435 – 1700</td>
<td>COSYSMO 3 Workshop</td>
<td>Jim Alstad (USC)</td>
<td>2&lt;sup&gt;nd&lt;/sup&gt; Floor, GVC-A</td>
</tr>
</tbody>
</table>

### Wednesday, August 23, 2017 – Contractor Discussions (Restricted)

<table>
<thead>
<tr>
<th>Time</th>
<th>Event</th>
<th>Speaker(s)</th>
<th>Location</th>
</tr>
</thead>
<tbody>
<tr>
<td>1435 – 1600</td>
<td>NGA PMO Analytic Services One-on-One</td>
<td>Brian Cali (IAI) Patrisha Knight (NGA)</td>
<td>Auditorium</td>
</tr>
</tbody>
</table>

### Thursday, August 24, 2017 – General Session (Open to All)

<table>
<thead>
<tr>
<th>Time</th>
<th>Event</th>
<th>Speaker(s)</th>
<th>Location</th>
</tr>
</thead>
<tbody>
<tr>
<td>0800 – 0830</td>
<td>Registration</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0830 – 1230</td>
<td>CADE Training Session</td>
<td>Torri Preston and Marc Stephenson (OSD CAPE)</td>
<td>Auditorium</td>
</tr>
</tbody>
</table>

### Thursday, August 24, 2017 – Contractor Discussions (Restricted)

<table>
<thead>
<tr>
<th>Time</th>
<th>Event</th>
<th>Speaker(s)</th>
<th>Location</th>
</tr>
</thead>
<tbody>
<tr>
<td>0830 – 1000</td>
<td>VMWare One-on-One</td>
<td>Carol Traynor and Don B (VMware)</td>
<td>2&lt;sup&gt;nd&lt;/sup&gt; Floor, GVC-A</td>
</tr>
<tr>
<td>1010 – 1140</td>
<td>Amazon One-on-One</td>
<td>Seabreeze Osburn (Amazon)</td>
<td>2&lt;sup&gt;nd&lt;/sup&gt; Floor, GVC-A</td>
</tr>
</tbody>
</table>
Dr. John Zangardi became the Principal Deputy Department of Defense Chief Information Officer on October 2, 2016, and is currently serving as the Acting DoD CIO. As the Acting DoD CIO, Dr. Zangardi assists as the primary advisor to the Secretary of Defense for Information Management / Information Technology and Information Assurance as well as non-intelligence space systems; critical satellite communications, navigation, and timing programs; spectrum; and telecommunications.

Dr. Zangardi's background includes acquisition, policy, legislative affairs, resourcing, and operations. In his most recent assignment as the Deputy Assistant Secretary of the Navy for Command, Control, Communications, Computers, Intelligence, Information Operations, and Space (DASN C4I, IO, and Space), he was responsible for providing acquisition oversight for C4I, cyber, space, business enterprise, and information technology programs. In 2014 and 2015, he additionally served as the acting Department of the Navy Chief Information Office (DON CIO).

Dr. Zangardi is a retired Naval Flight Officer and served in a variety of command and staff assignments. After retiring from the Navy, Dr. Zangardi was selected for appointment to the Senior Executive Service (SES) and assigned as the Deputy Director Warfare Integration Programs (N6FB) within the Deputy Chief of Naval Operations Communications Networks (N6) Directorate. With the stand-up of the Deputy Chief of Naval Operations Information Dominance (N2/N6), he was assigned as the Director for Program Integration and as Deputy to the Director for Concepts, Strategy, and Integration.

He is a native of Scranton, Pennsylvania and a graduate of the University of Scranton. Dr. Zangardi was awarded a Master of Science degree from the Naval Postgraduate School and a Doctor of Philosophy degree from George Mason University.
0845 - 0915: Agile and GAO Cost Estimating Best Practices
Karen Richey, Government Accountability Office

Abstract
This paper will examine how GAO’s cost estimating process can be applied to programs that are using an Agile framework. First, it will provide a brief overview of Agile processes and methods. Second, it will examine each of the 12 steps in the GAO cost estimating process and how those steps relate to an Agile framework. Finally, it will discuss how Agile artifacts can be leveraged to fulfill cost estimating documentation needs.

0920 - 0950: How Should We Estimate Agile Projects and Measure Progress to Plan?
Thomas J. Coonce, Institute for Defense Analyses
Glen B. Alleman, Niwot Ridge, LLC

Abstract

1005 - 1035: Software Size Growth
Marc Russo and Corinne Wallshein, Naval Center for Cost Analysis

Abstract
Software cost estimating relationships often rely on software size growth percentages. Actual delivered source lines of code (SLOC) may be predicted with categories of early code estimates such as new, modified, reuse, and auto-generated SLOC. Uncertainty distributions will be presented to represent growth by code category for use in cost modeling.

David Seaver, National Security Agency

Abstract
The Business Intelligence and Analysis organization (B4) develops independent cost estimates for the National Security Agency (NSA). For software intensive systems B4 creates independent software size estimate with functional size estimation techniques. The functional size is converted to source lines of code (where relevant) using B4 historical data from prior completed programs. B4 uses a streamlined functional size technique called Simple Function Points (SFP) to develop the functional size estimate. To count and analyze the SLOC B4 uses USC UCC with some custom tools wrapped around UCC.

The first part of this presentation will provide a brief overview of this process, items to be discussed include: Agile and DevOPS defined; What’s different from classic waterfall projects; What business processes (for estimation) need to be changed; What data collection processes (for estimation) have to be changed.

The second part of the presentation will discuss how this process modification has been applied or will be applied to estimate and measure: Business Systems; Analytic Development; Infrastructure Programs.

1115 - 1145: Assessing ERP Cost, Schedule and Size Growth

Haset Gebre-Mariam, Naval Center for Cost Analysis

Rob Williams, Herren Associates

Abstract
This study will examine percentage changes in cost, schedule, and size across Milestones A, B, C, and full deployment for DoD Enterprise Resource Planning (ERP) programs. The analysis is based on nine fielded systems collected from DoD authoritative data sources. Cost contributors, drivers, and factors by major cost elements will also be examined. Results may be used for crosschecking cost estimates or business case analyses at an early phase to inform funding decisions.
1300 - 1330: Objective SLOC: An Alternative Method to Sizing Software Development Efforts

Andrew Kicinski, Integrity Applications Incorporated

Abstract
Equivalent Source Lines of Code (ESLOC) is the basis of methodology used by many organizations for collecting and estimating software development costs. Selecting ESLOC parameters requires insight into the software reuse. Too often data collectors are unable to verify the appropriateness of the assigned ESLOC parameters and validate their implementation. This paper examines the drawbacks of ESLOC, and presents an alternative and more objective method to estimating software development effort.

1335 - 1405: Software Cost Estimation Meets Software Diversity

Barry Boehm, University of Southern California

Abstract
The previous goal of having a one-size-fits-all software cost (and schedule) estimation model is no longer achievable. Sources of wide variation in the nature of software development and evolution processes, products, properties, and personnel (PPPPs) require a variety of estimation models and methods best fitting their situations. This talk will provide a short history of pattern-breaking changes in software estimation methods; a summary of the sources of variation in software PPPPs and their estimation implications; a summary of the types of estimation methods being widely used or emerging; a summary of the best estimation-types for the various PPPP-types; and a process for guiding an organization's choices of estimation methods as their PPPP-types evolve.

1415 - 1700: COCOMO III Workshop: Implementing a New Driver for Software Security

Brad Clark and Barry Boehm, University of Southern California

Abstract
COCOMO (CONstructive COst MOdel) is an open-source model that allows analysts to estimate the cost, effort, and schedule when planning a new software development activity. This workshop will begin with a brief overview of the COCOMO III project and the proposed cost estimation model. The focus will then shift to an overview of how to make software applications secure and the associated cost impact.

The main purpose of the workshop and the majority of time will be spent discussing ideas for incorporating software security cost estimation in the COCOMO III model. Participants should come to the workshop prepared to learn about and discuss how to make software secure.
0810 - 0840: Army Software Maintenance Cost Estimating Relationships in a Diverse Execution Environment
Cheryl Jones and John McGarry, U.S. Army ARDEC
James Doswell and Jenna Meyers, U.S. Army DASA-CE

Abstract
For the past four years, the Army, under the leadership of DASA-CE, has been collecting and analyzing Army system software maintenance cost and technical execution data to support the development of more accurate cost estimation methods. The presentation will present the cost methods and cost estimation relationships developed from the analysis of the initial execution data sets. It will address how the collected software maintenance data was evaluated, characterized and normalized; show cost distributions across the primary functional domains; and present a set of derived software maintenance CERs and benchmarks.

0845 -0915: Apples and Oranges: a Presentation and Analysis of Results of Cloud Cost Calculators and Rate Cards
Daniel J Harper, MITRE Corporation

Abstract
A recent effort for an Army customer examined over a dozen calculators and rate cards for estimating storage and hosting costs for cloud applications. This presentation will provide an overview of several calculators and tools, guidance for cost estimators on interpreting IT-centric inputs, and a discussion of similarities and variation in results. We will also present a cloud complexity plotter which provides a visual tool for explaining cloud cost and complexity drivers.

0920 -0950: Rosetta Stone for Software Sizing
Victor Fuster and Taylor Putnam-Majarian, QSM Inc.

Abstract
Wouldn’t it be nice if some sort of software sizing “translator” existed, such as the Rosetta Stone for languages? The original Rosetta Stone listed the same text in three languages (Ancient Greek, Demotic script, and Ancient Egyptian hieroglyphics), serving as a "decoder" that helped give meaningful interpretation to the previously mysterious hieroglyphics. The Rosetta Stone for Software Sizing works to accomplish the same result for software sizing by translating units of need into units of work using gearing factors. This allows one to size the same project using multiple methods (requirements, function points, RICE counts, SLOC, etc.). We present our methodology and show how this technique can provide valuable insights and analysis for oversight, management, and development estimation. Additionally, we discuss at least two examples of the methodology’s recent implementation to Enterprise Resource Planning (ERP) project estimation in the DoD and commercial environments.
1000 - 1030: SRDR Unified Review Function (SURF): A Deeper Focus on Software Data Quality

Nicholas Lanham and Marc Russo, Naval Center for Cost Analysis

Abstract
From December 2015 to December 2016, the SURF team completed the development of a standardized V&V question template that was used to develop over 1,282 additional data quality comments. Throughout the review process and as SURF members' generated V&V comments, each one was "tagged" to a specific section of the SRDR V&V guide to identify specific SRDR variables that generate the most data-quality concerns. This presentation summarizes the V&V comment trends generated by the SURF team's 1,282 V&V comments. In addition, this paper helps to raise attention to specific SRDR variables and illustrates tangible data quality improvements to highly critical DoD software data. It also provides detailed metrics to demonstrate how SURF is working and the significant-positive impact the V&V guide + SURF team + new SRDR review process is making on the Government's data-quality.

1035 - 1105: Expanding the Horizons of Software Cost Estimation

Jairus M Hihn, NASA Jet Propulsion Laboratory

Abstract
This presentation summarizes the results of ten years of research in using data mining and machine learning methods to develop analogy estimation models. These results are based on the analysis of NASA robotic spacecraft flight software data obtained from the NASA CADRe and other data sources that have been collected for over thirty years. The results of the research indicate that cluster based algorithms are an in important supplement to parametric models especially early in the lifecycle when information is limited and uncertain.

1110 - 1140: Why Does Software Cost So Much? Towards a Causal Model

Bob Stoddard and Mike Konrad, Software Engineering Institute

Abstract
How can we control the cost of software intensive systems? Software costs continue to escalate as software continues to become an increasing portion of DoD systems. To contain costs we need to better understand the factors that drive costs and which factors we can control. Although we know relationships, we do not yet separate the causal influences from non-causal spurious correlations. By applying a new set of recently developed causal discovery and modeling tools to the research data, causality can be identified, measured, and tested. Existing literature on software cost contains primarily case studies and correlational studies from project data that continue to suffer from limited, public data and overreliance on correlational techniques. Correlation does not logically imply causation, hence correlational results are not necessarily useful for driving reductions in cost. In this talk, we will share early research results that will differentiate true causal factors from those spuriously correlated with cost.
1245 - 1315: Reliable Non-Design, Code, Test and Integration Cost Relationships

Brittany Staley and Jeremy Goucher, Herren Associates

Abstract
Software cost estimates require ratios derived from historic cost reports for non-design, code, test, and integration (NDCTI) cost elements. Since NDCTI accounts for as much as 50% of the estimate, a comprehensive historical data set is critical to ensuring an accurate estimate. The authors have recently analyzed over ten years of actual cost data from DoD command and control systems to develop a new set of NDCTI ratios. The results also bring new insight into “fixed” versus “variable” cost.

1320 - 1350: Introduction to Software Obsolescence Cost Analysis Framework

Sanathanan Rajagopal, QinetiQ, United Kingdom

Abstract
Software plays an important role in defence. Almost every project in defence has software elements with various degrees of complexity and dependencies. This has brought its own challenges to the availability-based contracts. The challenges to both the contractors and the suppliers is that they have to have a good understanding of the whole life cost of the product and have confidence in the whole life cost model at the time of negotiation and contract signing. In order to understand and see the bigger picture developers and the customers need to foresee the following issues that drive the whole life cost and should be in a position to develop innovative means to mitigate these issues by

- Anticipation of the Software Obsolescence at a very early stage of projects.
- Understanding the technology insertion, technology update requirement.
- Understanding the relationship between Software Maintenance and Software Obsolescence.
- Anticipation of future capability integration to the existing platforms
- Formulation and evaluation of alternative architectural framework to inform the software designers that recognizes the key market and cost drivers.

Software Obsolescence Cost Analysis Framework will help in managing software obsolescence proactively and help to estimate the cost of Software Obsolescence Resolution. This framework is at very early stages of its development and intended to release it once the validation is complete.

1355 - 1425: Cost Assessment Data Enterprise Overview and Software Initiatives

Daron D Fullwood, Cost Assessment and Program Evaluation

Ranae Woods, SES, Air Force Cost Analysis Agency

Abstract
Learn about the future of cost data collection from the CAPE perspective. Will provide an update on CADE and ensure the community is aware of ongoing efforts. This session will focus on CADE’s data initiatives along with an update on the Software Resource Data Reports.
**Wednesday, August 23, 2017**

**1430 - 1700: COSYSMO 3.0 Workshop: Updating Cost Estimation of Systems Engineering to Support Affordability**

Barry Boehm and Jim Alstad, University of Southern California

**Abstract**

The purpose of the COSYSMO (Constructive Systems Engineering Cost Model) model is to estimate the Systems Engineering effort for large-scale systems (both software and hardware). COSYSMO supports the ANSI/EIA 632 standard as a guide for identifying the Systems Engineering tasks and ISO/IEC 15288 standard for identifying system life cycle phases.

This presentation will cover a mature draft of the COSYSMO 3.0 model, explaining both the new features and the unchanged features. The presentation is recommended for those with experience in systems engineering, especially as project leads or cost estimators.

---

**Thursday, August 24, 2017**

**0830 - 1230: Cost Assessment Data Enterprise (CADE) Training**

Torri Preston and Marc Stephenson, OSD Cost Assessment and Program Evaluation (CAPE)

**Abstract**

The OSD CAPE mission is to provide high quality, independent program analyses and insights as requested by the Under Secretary of Defense for Acquisition, Technology and Logistics (USD(AT&L)) and Congress, in addition to the review of programs that may be, or already are, struggling in the acquisition process. CAPE initiated development of CADE, the Department’s initiative to identify and integrate data from disparate databases and systems for better decision-making, management of, and oversight of the Department’s acquisition portfolio. The CADE primary function is to house authoritative data sources that are seamlessly integrated, and easily searchable and retrievable to support analytics.

The CADE training session offers better insight into contract cost reporting and how to follow specific regulations outlined by the DCARC. Major Defense Acquisition Program (MDAP) and Major Automated Information Systems (MAIS) program personnel, government and industry, who are interested and involved in Cost and Software Date Reporting (CSDR) contracting and reporting, are encouraged to attend the event.
Software and Information Technology Cost Analysis Solution Team

Dr. John Zangardi
Acting Department of Defense Chief Information Officer
August 22, 2017
IT requires fast movers.
Enterprise Information Technology
Achieving Effectiveness in Change Management

Change Effectiveness Equation

Q x A = E

Q = Quality of Technical Strategy
A = Acceptance of that Strategy
E = Effectiveness

The People Side of the Equation is just as important as the Technical Side of the Equation

Equation derived from GE’s Change Acceleration Process (CAP)™
Enterprise Information Technology
Achieving Effectiveness in Change Management

Change Effectiveness Equation

\[ Q \times A = E \]

If Acceptance \((A) = 0\), then Effectiveness \((E)\) always \(= 0\), regardless of the strength of your Technical Strategy \((Q)\)
Change Process for Enterprise IT

Blocking and Tackling Fundamentals for Success

“As Is” State ➔ Transition State ➔ “To Be” State

Fundamentals for Success in Enterprise IT:

- Shared need, shared vision, shared commitment
- Upfront business process and change management
- Communication and collaboration among stakeholders
- Leadership engagement and initial buy-in

Getting the fundamentals wrong drives up costs
Defense Travel System Modernization

Driving Change with Many Stakeholders

**Problem Statement:** DoD travelers are dissatisfied with the current complex, costly, and cumbersome travel solution.

**Stakeholders:** DoD CIO, USD(AT&L), DCMO, USD(P&R), CAPE, OSD(C)
DoD Enterprise IT Initiatives
_Maximizing Effectiveness and Efficiency_

- **Cloud Security** • Innovating delivery and security approaches to move more data into the commercial cloud

- **Defense Enterprise Office Solutions** • Connecting the workforce through commercial, enterprise office solutions for collaboration and productivity

- **Windows 10 Transition** • Reducing and protecting the Department’s attack surface through a common DoD-wide operating system

- **Joint Regional Security Stacks (JRSS) Rollout** • Making progress on the path to migrating DoD Components to JRSS

- **Data Center Optimization** • Driving efficiency by optimizing DoD data storage solutions and moving to a data center scorecard

_Support The Warfighter_

_Speed to Capability • Balancing Security and Cost • Instilling a Culture of Risk Awareness_
Agile Development and GAO Cost Estimating Best Practices

Karen Richey
August 22, 2017
Outline

• Introduction
• Agile Background
• GAO Cost Estimating 12-Step Process
• Mapping Cost Estimating Best Practices to Agile Methods
• Conclusion
• Next Steps
Introduction Problem

- Federal agencies depend on Information Technology (IT) to support their missions.
  - The government spends more than $80 billion annually on information technology systems
- Congress has expressed interest in monitoring and improving IT investments through hearings and other reviews over the past two decades.
- In 2010, the Office of Management and Budget (OMB) expressed concern about federal IT projects that have taken years but have failed to produce results.
- Common pitfalls are that Agile is often used as an excuse not to
  - Document,
  - Plan for the software development process, and
  - Provide traditional program management tools (e.g. cost estimates, schedule estimates, etc.)

While federal IT investments can improve operational performance and increase public interaction with government, too often they have become risky, costly, and unproductive mistakes

*Government Accountability Office, 2012*
Introduction – Solution?

• One solution to reduce risks associated with broadly scoped, multiyear projects is to use shorter software delivery times
  • Incremental Development
    • One approach to improving federal government IT investments and encouraged by both OMB and GAO
    • Involves planning and delivering new or modified technical functionality or services to users at least every six months
    • The Federal IT Acquisition Reform Act (FITARA), enacted in December 2014, calls for the Chief Information Officer of each covered agency to annually certify that IT investments are adequately implementing incremental development.

• Agile software development supports the practice of
  • Continuous software delivery
  • Developing solutions that include distinct features, some of which may be discovered along the way rather than planned up front
Agile Background

- Agile practices integrate planning, design, development, and testing into an iterative life-cycle to deliver software at frequent intervals

- Short iterations are used to
  - Effectively measure progress,
  - Reduce technical and programmatic risks, and
  - Respond to feedback from stakeholders faster than traditional methods

- More a philosophy than a methodology, the Agile Manifesto articulated four principle values that prefer
  - Individuals and interactions over processes and tools,
  - Working software over comprehensive documentation,
  - Customer collaboration over contract negotiation, and
  - Responding to change over following a plan

While there is value in both parts of each principle, the first part is seen as most important
Agile Alliance 12 Guiding Principles

1) Our highest priority is to satisfy the customer through early and continuous delivery of valuable software.

2) Welcome changing requirements, even late in development. Agile processes harness change for the customer’s competitive advantage.

3) Deliver working software frequently, from a couple of weeks to a couple of months, with a preference to the shorter timescale.

4) Business people and developers must work together daily throughout the project.

5) Build projects around motivated individuals. Give them the environment and support they need, and trust them to get the job done.

6) The most efficient and effective method of conveying information to and within a development team is face-to-face conversation.

7) Working software is the primary measure of progress.

8) Agile processes promote sustainable development. The sponsors, developers, and user should be able to maintain a constant pace indefinitely.

9) Continuous attention to technical excellence and good design enhances agility.

10) Simplicity, the art of maximizing the amount of work not done, is essential.

11) The best architectures, requirements, and designs emerge from self-organizing teams.

12) At regular intervals, the team reflects on how to become more effective, then tunes and adjust its behavior accordingly.
Agile Background
Five Levels Commonly Followed with Agile Planning

- Vision
- Roadmap
- Release
- Iteration
- Daily Work
Agile Background
Traditional vs. Agile Development

**Traditional Development**
- Fixed Requirements
- Linear Development Approach
- Single delivery of end product

**Agile Development**
- Flexible Requirements
- Iterative Development Approach
- Multiple deliveries over time
Traditional Development

- Plan (Requirements Development)
- Analysis
- Design
- Coding
- Test and Integration
- Operations

Documentation
Code delivered but not complete
Code released to the user and ready to go!
Agile Development

Release 1

Iteration 1

Design → Test → Build → Design

Iteration 2

Design → Test → Build → Design

Release 2

Iteration 3

Design → Test → Build → Design

Iteration 4

Design → Test → Build → Design

...Etc.

Release documentation

Code released to the user and ready to go!
Agile Background
Changes to Program Management Philosophy

Traditional Development
- Scope: Plan Driven
- Cost: Fixed
- Schedule: Flexible

Agile Development
- Scope: Value Driven
- Cost: Flexible
- Schedule: Fixed
Agile Background
Benefits to Estimating

- Effort is commonly used as a proxy for cost
  - Without estimating effort, cost cannot be determined for near and long-term deliverables
- Understanding the capacity (e.g. the total amount of work that Agile teams can accomplish in the short-term) helps to prioritize work
- Gaining Agile team commitments to delivering near-term features in upcoming iterations and releases is important because these commitments drive the planning of customer business objectives.
- Estimating is the key to unlocking the ability to commit
  - Agile development focuses on producing incremental deliverables based on team commitments regarding what will be accomplished in the near-term

As in traditional programs, an estimate is not final and should be updated with information as it becomes available
GAO and Cost Estimating
12-Step Cost Estimating Process

Initiation and research
Your audience, what you are estimating, and why you are estimating it are of the utmost importance

Assessment
Cost assessment steps are iterative and can be accomplished in varying order or concurrently

Analysis
The confidence in the point or range of the estimate is crucial to the decision maker

Presentation
Documentation and presentation make or break a cost estimating decision outcome

Analysis, presentation, and updating the estimate steps can lead to repeating previous assessment steps

Define the estimate's purpose
Develop the estimating plan
Define the program
Determine the estimating structure
Identify ground rules and assumptions
Obtain the data
Develop the point estimate and compare it to an independent cost estimate
Conduct sensitivity
Conduct a risk and uncertainty analysis
Document the estimate
Present estimate to management for approval
Update the estimate to reflect actual costs/changes

Source: GAO.
Agile and Cost Estimating

- Many artifacts that help manage Agile development programs can be used to inform the cost estimating process
  - These artifacts provide a clear picture of the planning that the program office did to determine the prioritized features and release / iteration schedule
- New data should be captured at the end of each iteration
  - Agile software cost estimates should be updated at the end of each release (and other important milestones) with actual costs for the specific features that were implemented

Align the cost estimate with the program’s Agile cadence
Mapping Cost Estimating Best Practices to Agile Methods

<table>
<thead>
<tr>
<th>Estimating Process Step</th>
<th>Agile Cadence Example</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Step 1</strong>: Define the estimate’s purpose</td>
<td>During initial and subsequent release planning, determine how any cost estimates will be used.</td>
</tr>
<tr>
<td><strong>Step 2</strong>: Develop the estimating plan</td>
<td>During initial planning, the cost estimating team should be identified along with all technical experts so that Agile team capacity measures can be determined.</td>
</tr>
</tbody>
</table>
Mapping Cost Estimating Best Practices to Agile Methods (Continued)

<table>
<thead>
<tr>
<th>Estimating Process Step</th>
<th>Agile Cadence Example</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Step 3</strong>: Define the program</td>
<td>These steps should occur during initial planning once the Vision and Roadmap have been developed.</td>
</tr>
<tr>
<td><strong>Step 4</strong>: Determine the estimating structure</td>
<td>A prioritized product backlog and product-oriented work breakdown structure (WBS) capture the program requirements that align to the Vision and Roadmap. The assumed number of iterations, releases, and size/cost of the Agile teams provide estimators with the timeframes and loaded labor rates needed to determine the cost to implement features.</td>
</tr>
<tr>
<td><strong>Step 5</strong>: Identify the ground rules and assumptions</td>
<td>After each iteration, specific Agile artifact data can be used to refine the estimate including:</td>
</tr>
<tr>
<td><strong>Step 6</strong>: Obtain data</td>
<td>• burn up/burn down charts,</td>
</tr>
<tr>
<td><strong>Step 7</strong>: Develop the point estimate and compare it to an independent cost estimate</td>
<td>• Velocity metrics, and</td>
</tr>
<tr>
<td></td>
<td>• additional requirements that were discovered and added to the backlog</td>
</tr>
<tr>
<td></td>
<td>Independent cost estimates should be developed to check the reasonableness of the initial cost estimate as well as any new estimates prior to major milestone reviews.</td>
</tr>
</tbody>
</table>
Mapping Cost Estimating Best Practices to Agile Methods (Continued)

<table>
<thead>
<tr>
<th>Estimating Process Step</th>
<th>Agile Cadence Example</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Step 8: Conduct sensitivity</strong></td>
<td>Sensitivity analysis should be conducted on the initial point estimate once the Vision and Roadmap are completed.</td>
</tr>
<tr>
<td></td>
<td>This analysis should be repeated whenever the estimate is updated to understand what drives cost.</td>
</tr>
<tr>
<td><strong>Step 9: Conduct a risk and uncertainty analysis</strong></td>
<td>Risk and uncertainty analysis should occur after the initial point estimate has been developed so that risks affecting the work are known upfront.</td>
</tr>
<tr>
<td></td>
<td>This analysis should be updated along with the point estimate to reflect new Agile artifact data and any technical or schedule program risks.</td>
</tr>
</tbody>
</table>
### Agile and Cost Estimating

#### 12-Step Process (Continued)

<table>
<thead>
<tr>
<th>Estimating Process Step</th>
<th>Agile Cadence Example</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Step 10</strong>: Document the estimate</td>
<td>Documentation of the cost estimate should follow the same cadence that the Agile project has established for updates to the Vision, Roadmap, or other strategic documentation.</td>
</tr>
<tr>
<td><strong>Step 11</strong>: Present the estimate to management for approval</td>
<td>Management should review and sign off on the estimate and its underlying ground rules and assumptions before any major program reviews so that decisions can be based on the most recent information.</td>
</tr>
</tbody>
</table>
| **Step 12**: Update the estimate to reflect actual costs/changes | The estimate should reflect the most current Agile artifact data (i.e. burn up/down charts, velocity, actual vs. planned work, changes in requirements, program risk assessments, etc.) and capture variances so that lessons learned can be applied to future estimates.  
At a minimum, the estimate should be updated before any major milestone decision. Most often, the estimate will be updated at predetermined times that align with the program’s Agile cadence. |
Conclusion

• While the Agile approach is different from traditional software development methods, the need for a high-quality, reliable cost estimate is still applicable for government programs.

• The GAO 12-step cost estimating process and associated best practices still apply to programs using Agile methods.
  • Agile development generates new data after every iteration which can be used to continually update the estimate.
  • Agile methods lower program technical risk by developing software in small segments and continually delivering users desired features to obtain early feedback
    • Analyses such as sensitivity and risk/uncertainty can still be used to inform management decisions as more information becomes known about user needs and business value.
  • While Agile emphasizes working software over comprehensive documentation, information regarding initial assumptions, reasons for variances, and lessons learned should still be captured and used to improve future estimates.
Next Steps – Agile Best Practices Guide in Development

- GAO is establishing an Agile Development and Implementation guide to establish a consistent framework based on best practices that can be used across the federal government for developing, implementing, managing, and evaluating agencies’ IT investments that rely on Agile methods.

- These best practices will be used as a basis for the development of a chapter focusing on Agile and how it relates to cost, schedule, and EVM.
  - All chapters will be thoroughly vetted through GAO’s Agile Expert Group, which meets 3x per year (next meeting will be August 24, 2017).
  - An exposure draft of the entire guide will be published to the GAO website for a year-long open comment period.
  - Those interested in working to develop this guide should contact Jennifer Leotta, leottaj@gao.gov for more information.
Next Steps Agile Guide Draft Outline

- Chapter 1 – Background
- Chapter 2 – Compliance and Past Work
- Chapter 3 – Agile Adoption Best Practices
  - Team activities, Program processes, and Organizational Environment
- Chapter 4 – Agile Implementation Challenges
- Chapter 5 - Agile Metrics
- Chapter 6 – Requirements Decomposition
- Chapter 7 – Agile and the Federal Acquisition Process
  - Agile and Federal Contracting Process / Budget Process
- Chapter 8 - Agile and Program Management Factors
  - Program Planning and Tradeoffs, Team composition
- Chapter 9 – Agile Program Control Best Practices
  - Cost estimating, Scheduling, and Earned Value Management

Appendices:
- Agile Glossary
- Effects of not following best practices
- Agile Methodologies
- Debunking Agile Myths
- Questions for Auditors and Managers
- Case Study Descriptions
Sources

Sources


How Should We Estimate Agile Projects and Measure Progress to Plan

Thomas J. Coonce
Glen B. Alleman
Lockheed Martin Global Vision Center
August 22, 2017

Naval Center for Cost Analysis
Software and IT Cost Analysis Solutions Team (CAST)
“Why do so many big projects overspend and overrun?

They’re managed as if they were merely complicated when in fact they are complex. They’re planned as if everything was known at the start when in fact they involve high levels of uncertainty and risk.”

– Architecting Systems: Concepts, Principles and Practice,
Hillary Sillitto
All these Conditions Exist in the Global Dimension of Modern Acquisition

Pace of Technology

Black Swan Syndrome

Rise of the Commons

Technology Commercialization

Expanding Global Knowledge Base

Economic and S&T Mega-Trends

Information Agility

Mass Collaboration

Source: Dr Reginald Brothers’ chart
Deputy Assistant Secretary of Defense for Research
Why Agility Matters?

- **Agility Reflects Reality** – accepting uncertainty, driving it out, and reprioritizing efforts based on new information is how the world works.

- **Agility Enables Flexibility** – the freedom to make the right decisions at the right time, based on the right amount of information.

- **Agility is Path to the Present** – the expectation of customers, users, and buyers, is that things will be on a path of constant improvement and zero issues, or they’ll jump to the next most available platform.
The Results is Four Immutable Truths of Software Development

1. You can’t gather all the requirements upfront.

2. The requirements you do gather will change.

3. There is always more work than time and money available.

4. Estimates will always be off by some factor, and this factor is likely unknown.
6 Top Level Processes to Increase Probability of Program Success

1. **Pre-Award**
   - Define desired capabilities
   - Assess readiness of technologies.
   - Define war fighter’s use System
   - Define Measures of Effectiveness
   - Create Integrated Master Plan
   - Identify uncertainties
   - Develop risk-adjusted estimates

2. **Issue Request for Proposal**
   - Include Government products
   - Specify award criteria
   - Update IMP and Uncertainties
   - Define key framing assumptions
   - Submit updated cost and schedule estimate to 70% JCL
   - Submit deterministic IMS

3. **Award Based on Criteria**
   - Establish these criteria in the Integrated baseline Review
   - Measures performance and award fee against these criteria
   - Use criteria to produce ETC, EAC, ECD

4. **Awardee Creates Credible PMB**
   - Integrated Master Plan
   - Technical Plan
   - WBS and Dictionary
   - Program Management Plan

5. **Install Credible PMB**
   - Define Measures of Effectiveness, Measures of Performance, technical Performance Measures, or Key Performance Parameters for each deliverable

6. **Monitor Progress to Plan**
   - Ensure technical progress made according to Plan
   - Review cost and schedule progress according to Plan
   - Update risk register
   - Identify which activities require closer monitoring in the future
The PMB Connected to Agile Development Processes

- Horizontal and Vertical traceability for all plans and work, IAW FAR acquisition rules.
- Using Measures of Physical Percent Complete of the planned Features using Measures Effectiveness (MoE) and Performance, (MoP) and Technical Performance Measures (TPM)
10 Steps to Apply Agile on Federal Programs using EVM

- **Initial Estimate**
- **Product Roadmap**
- **Release Plan**
- **IMS with Features in WP**
- **Product Backlog**
- **Sprint Backlog**
- **Task Estimates During Sprint**
- **TO DO Updates Produce Physical Percent Complete**
- **Update Feature in IMS with Physical Percent Complete**
- **Update Physical Percent Complete in EVMS**

Continuous feedback at each step with corrective actions for Root Cause of Performance Variances

Closed Loop Feedback Control for Agile at Scale

EVM is applied to DHS IT Acquisition projects IAW *Capital Planning and Investment Guide*
Start with Decomposing the ConOps into Capabilities, Features, and Stories

ConOps is a document describing the capabilities of a proposed system from the viewpoint of an individual who will use that system. It is used to communicate the quantitative (Measure of Performance) and qualitative (Measure of Effectiveness) characteristics to all stakeholders.

Capabilities needed to accomplish Mission, defined from ConOps defined before Contract Award and placed in RFP, with flowing down Measures of Effectiveness and Measures of Performance

Features from Capabilities, further refined after Contract Award in the Product Backlog. Features decomposed into Stories and Task in the Agile system for development
First Steps to Estimating Agile Software Development

Increasing Maturity of Agile Estimating Using Function Points (FP)

Establish a Repository for project performance information using Function Point (FP) from past projects to build Reference Class Forecasting DB

Start Feature Breakdown Structure (FBS) from existing and past projects. Record time and cost for each Feature in the FBS for current projects.

Decompose current project from past FBS in repository. Add new Features as discovered.

Connect time and cost estimates to past FBS. Add New FBS Feature estimates from Product Backlog and Release Plan.

Update FBS repository with new cost and time data at the of each Release or Feature delivery to increase fidelity of DB.

Migrating from FP to FBS with Time and Cost data in Repository, just like WBS

Like the WBS elements in the CCDR database, a Feature Breakdown Structure can be built from past program and used as Reference Class for future programs. Like FP Enterprise IT system have many common or derivable elements.
Reference Class Step By Step
From Nobel Prize to Project Management, Bent Flyvbjerg

Step I
1. Identify relevant reference class of past similar projects
   Broad enough to generate some sample size of power, but narrow enough to be specific

Step II
2. Establish a probability distribution for that class

Step III
3. Compare specific project to reference class’ distribution
   = Regressing forecaster’s estimates toward the average of the reference class & extending the credible interval
   toward the corresponding for the class

Frame forecasts with all available information

Also simple questions which force estimators to take outside views
Ask experts about previous similar projects (espy regarding scope)
Can not help to predict extreme cases or outliers

Function Points and Agile

- To measure the productivity and evaluate percent of increase and decrease in productivity rate
- Helps end users/clients to quantify the number of requirements emended in software
- Prepare the estimation for software development
- Prepare the cost related metrics for software development
- Used in Decision Analysis and Resolution Techniques (DART)
- Used to prepare the resource pyramid for software development
Start Estimating Using Function Points

- For each Capability, list the business and data transactions for the Features that implement the Capability
- For each Feature, list the business and data transactions for the Stories that implement the Feature
- With the Functional Point count, assess capacity for work in FPs
Measuring Physical Percent Complete at the Sprint Level

<table>
<thead>
<tr>
<th>Feature Hrs</th>
<th>100</th>
<th>125</th>
</tr>
</thead>
<tbody>
<tr>
<td>Updated Feature Forecast</td>
<td>100</td>
<td>125</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Task Est</th>
<th>TO DO</th>
</tr>
</thead>
<tbody>
<tr>
<td>US 1</td>
<td>10</td>
</tr>
<tr>
<td>US 2</td>
<td>10</td>
</tr>
<tr>
<td>US 3</td>
<td>10</td>
</tr>
<tr>
<td>US 4</td>
<td>10</td>
</tr>
<tr>
<td>US 5</td>
<td>10</td>
</tr>
<tr>
<td>US 6</td>
<td>10</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Sprint Est</th>
<th>40 Hrs</th>
<th>30 Hrs</th>
<th>70 Hrs</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sprint 1 % Cmplt</td>
<td>100%</td>
<td>50%</td>
<td>0%</td>
</tr>
<tr>
<td>Sprint 2 % Cmplt</td>
<td>44%</td>
<td>44%</td>
<td>44%</td>
</tr>
<tr>
<td>Sprint 3 % Cmplt</td>
<td>44%</td>
<td>44%</td>
<td>44%</td>
</tr>
</tbody>
</table>

**Original Engineering Estimate**
- Estimate of User Stories in Sprint
- Remaining Work for Story
- 0 Remaining Means Story Done
- 10 of 10 Remaining Means Story Not Stated

**Sprint 1** - 100% Complete
- After Sprint 1 Feature 32% Complete, with 60 Hrs remains

**Sprint 2** - 50% Complete
- At this point in Sprint 2, Features 44% Complete
Forecasting ETC/EAC with Earned Value using Physical Percent Complete

- Program performance in Agile or Traditional is the same at the PMB level.
- Physical Percent Complete is measured at the Feature level from the Agile SW Development System.
  - BCWS is *flat spread*, not an S-Curve
  - BCWP = BCWS × P%C

Although Agile is different and can be challenging, success can be achieved using the proven principles of Earned Value Management...
Summary

- Government requires a credible at the start based on a quality Concept of Operations (ConOps) for the system.
- Features, derived from the needed Capabilities, must be in the ConOps.
- *Ideally* we want to estimate Features, using a Reference Class Database containing a Feature Breakdown Structure – with hours and duration by Feature.
- Until this database is available, Function Points can be used to estimate the Agile ConOps.

Progress must be measured as *Physical Percent Complete*
Contact Information

Thomas J. Coonce  
TJC, LLC  
tom.coonce@verizon.net  
+1 703 362 2568

Glen B. Alleman, MSSM  
Niwot Ridge, LLC  
Glen.Alleman@niwotridge.com  
+1 303 241 9633
Backup
The Purpose of Credible Estimates

- Credible estimates start with Reference Class Database of Function Points models,
- These continue with Engineering Estimate updates from actual performance and emerging risks, and
- End with the application of effective program planning and controls principles.
Forecasting Future Performance is needed to Successfully Manage the project so we can ...

Determine where we are now. And is determined with a simple calculation, that says …

What Value did we plan to Earn? (BCWS)
Where are we now? (Physical Percent Complete)
What Value have we Earned to date? (BCWP)

BCWP = BCWS × Physical Percent Complete
Data Needed for Program Success Using Estimates
Flyvbjerg’s 4-Steps to Reference Class Forecasting

1) Form the reference class, a collection of similar-to projects for which there is both history and reasonable insight to the history so that adjustments for present time can be made.

2) Develop a true distribution of the reference class, and from that distribution calculate the cumulative probability.
   - This probability curve, developed from reference class, the outside view.

3) Develop the inside view.
   - The inside view is a traditional estimate by the project team.

4) Adjust the inside view based on the probability of historical outcome from the outside view.
   - Develop a forecast using the reference class probability confidence curve.
   - Pick a confidence limit, and then adjust the inside view to have a corresponding confidence.
# 10 Steps to a Credible Estimating Process†

<table>
<thead>
<tr>
<th>Step</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.</td>
<td>What are we Planning to Build?</td>
</tr>
<tr>
<td>2.</td>
<td>We Know We have to achieve this when?</td>
</tr>
<tr>
<td></td>
<td>- Who give the final “go live” decision?</td>
</tr>
<tr>
<td>3.</td>
<td>What’s the customer’s deliver date?</td>
</tr>
<tr>
<td>4.</td>
<td>What’s the Cost of Delay for each Feature in the Delivery?</td>
</tr>
<tr>
<td>5.</td>
<td>To Start, we need?</td>
</tr>
<tr>
<td></td>
<td>- The following prior software</td>
</tr>
<tr>
<td></td>
<td>- The following questions answered</td>
</tr>
<tr>
<td></td>
<td>- A minimum dedicated team</td>
</tr>
<tr>
<td></td>
<td>- Dependencies</td>
</tr>
<tr>
<td>6.</td>
<td>What thing might Impede our progress?</td>
</tr>
<tr>
<td></td>
<td>- Other projects?</td>
</tr>
<tr>
<td></td>
<td>- Events?</td>
</tr>
<tr>
<td></td>
<td>- Staff?</td>
</tr>
<tr>
<td></td>
<td>- Technical?</td>
</tr>
<tr>
<td>7.</td>
<td>To deliver, the following must be completed as well</td>
</tr>
<tr>
<td></td>
<td>- Testing</td>
</tr>
<tr>
<td></td>
<td>- Documentation</td>
</tr>
<tr>
<td></td>
<td>- Integration</td>
</tr>
<tr>
<td>8.</td>
<td>What do we need to learn to Deliver?</td>
</tr>
<tr>
<td></td>
<td>- Learn what?</td>
</tr>
<tr>
<td></td>
<td>- Learn How?</td>
</tr>
<tr>
<td>9.</td>
<td>What Skills do we need to Deliver?</td>
</tr>
<tr>
<td></td>
<td>- Skill?</td>
</tr>
<tr>
<td></td>
<td>- Number of People?</td>
</tr>
<tr>
<td>10.</td>
<td>How will we avoid finding quality or issues too late?</td>
</tr>
</tbody>
</table>

† FocusedObjective.Resources/Canvas and Forms/Forecast Assumption Canvas.pdf
Estimating Agile Work in Hours is Required for Federal contracts issued in Dollars

<table>
<thead>
<tr>
<th>Feature Request Title:</th>
<th>Hours</th>
</tr>
</thead>
<tbody>
<tr>
<td>Feature 1: Level of Effort and Administrative Activities</td>
<td>423</td>
</tr>
<tr>
<td>Feature 2: Modify SWS Class Update Process</td>
<td>312</td>
</tr>
<tr>
<td>Feature 3: Modify Contract Mod Launch Process</td>
<td>232</td>
</tr>
</tbody>
</table>

### Feature 1: Level of Effort and Administrative Activities

<table>
<thead>
<tr>
<th>Feature</th>
<th>Hours</th>
</tr>
</thead>
<tbody>
<tr>
<td>Feature 1: Level of Effort and Administrative Activities</td>
<td>423</td>
</tr>
<tr>
<td>Feature 2: Modify SWS Class Update Process</td>
<td>312</td>
</tr>
<tr>
<td>Feature 3: Modify Contract Mod Launch Process</td>
<td>232</td>
</tr>
</tbody>
</table>

#### Assumptions
- Baseline assumptions are not included in the Feature 1 table.
- The feature 2 and 3 are included in the Feature 1 and 3 tables.
- The estimated number of hours for feature 1 is 423, which is determined by the resources required to perform the work.

#### Risks
- Describe any assumptions that impact the estimate (Column B).
- Describe any risks to the estimate (Column C).
- Risk Impact on Hours (Column D).
- Story Points (Optional) (Column E).
- User Stories (Optional) (Column F).

#### Notes
- Basis of Estimate is the explanation and justification of your estimate to do the work. It describes the thought process, approach, and rationale used to arrive at the estimate being proposed.

---

**Business Case Request Title:**

**Feature Summary List:**

- Feature 1: Level of Effort and Administrative Activities
- Feature 2: Modify SWS Class Update Process
- Feature 3: Modify Contract Mod Launch Process
- Feature 4: Feature Name Goes Here
- Feature 5: Feature Name Goes Here
- Feature 6: Feature Name Goes Here
- Feature 7: Feature Name Goes Here
- Feature 8: Feature Name Goes Here
- Feature 9: Feature Name Goes Here
- Feature 10: Feature Name Goes Here
- Feature 11: Feature Name Goes Here
- Feature 12: Feature Name Goes Here
- Feature 13: Feature Name Goes Here
- Feature 14: Feature Name Goes Here
- Feature 15: Feature Name Goes Here
- Feature 16: Feature Name Goes Here
- Feature 17: Feature Name Goes Here
- Feature 18: Feature Name Goes Here
- Feature 19: Feature Name Goes Here
- Feature 20: Feature Name Goes Here

**Total Hours:**

- Total number of Hours for the Engineering Estimate: 956

---

**Features #**

<table>
<thead>
<tr>
<th>Portfolio</th>
<th>App</th>
<th>Rate/Team</th>
<th>Work Included</th>
<th>Company</th>
<th>Labor Category</th>
<th>Hours</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Feature 1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>423</td>
<td></td>
</tr>
<tr>
<td>Feature 2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>312</td>
<td></td>
</tr>
<tr>
<td>Feature 3</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>232</td>
<td></td>
</tr>
</tbody>
</table>

---

**Columns:**

- **Level of Effort and Administrative Activities:**
  - Feature 1: Level of Effort and Administrative Activities
  - Feature 2: Modify SWS Class Update Process
  - Feature 3: Modify Contract Mod Launch Process
  - Feature 4: Feature Name Goes Here
  - Feature 5: Feature Name Goes Here
  - Feature 6: Feature Name Goes Here
  - Feature 7: Feature Name Goes Here
  - Feature 8: Feature Name Goes Here
  - Feature 9: Feature Name Goes Here
  - Feature 10: Feature Name Goes Here
  - Feature 11: Feature Name Goes Here
  - Feature 12: Feature Name Goes Here
  - Feature 13: Feature Name Goes Here
  - Feature 14: Feature Name Goes Here
  - Feature 15: Feature Name Goes Here
  - Feature 16: Feature Name Goes Here
  - Feature 17: Feature Name Goes Here
  - Feature 18: Feature Name Goes Here
  - Feature 19: Feature Name Goes Here
  - Feature 20: Feature Name Goes Here

---

**Notes:**

- Basis of Estimate is the explanation and justification of your estimate to do the work. It describes the thought process, approach, and rationale used to arrive at the estimate being proposed.
## Earned Value + Agile

<table>
<thead>
<tr>
<th>Earned Value</th>
<th>Agile</th>
</tr>
</thead>
<tbody>
<tr>
<td>Decompose needed Capabilities into Features for future Reference Classes of Cost and Schedule data</td>
<td>Develop the Product Roadmap and Release Plan for the needed Capabilities and their Features</td>
</tr>
<tr>
<td>Prioritize Features based on Business Value of estimated effort (cost and time)</td>
<td>Using prioritized Features and place them in Product Backlog</td>
</tr>
<tr>
<td>Determine uncertainties of estimate with Monte Carlo Simulation tool</td>
<td>Define Stories and Tasks for Sprint, execute that work and update estimates to produce a Reference Class Forecasting database of Agile data</td>
</tr>
<tr>
<td>Place this information in the Performance Measurement Baseline Work Packages containing the Features</td>
<td>Use PMB Reference Class database to estimate emerging work as the program proceeds</td>
</tr>
</tbody>
</table>
Some Understanding of Agile Software Development (1)

- Product Roadmap defines what Capabilities are needed.

- The Product Roadmap implements the needed Capabilities found in the Concept of Operations

- Release Plan states when Features are available to fulfill the Capabilities.

- Product Backlog contains Features to be implemented in Sprints.

- Stories define the elements for the Features.

- Tasks define the work to deliver the Story.
Some Understanding of Agile Software Development (2)

- Physical Percent Complete defined by the 100% completion of a Story with its exit criteria.

- BCWS is the flat spread of the Labor for the Sprint.

- BCWP = BCWS × Physical Percent Complete.

- Estimating in Agile answers the question *Can we deliver the Features for the Budget?*

- Estimating in Traditional EVMS answers the question *What is the Cost for the needed Features?*
Executing the Program After Contract Award

- Using reference class data,
- Compare actual performance with planned performance, and
- Identify corrective actions needed to keep the program on plan.
Why Agility Matters.
All Modern Projects Operate in Presence of …

- **Caprice** (Unpredictability): unknowable situations, where …
  - Urgent needs regularly occurs
  - Un-availability of key personnel and/or subcontractors

- **Uncertainty**: randomness with unknowable probabilities, about …
  - Feasibility of solution design
  - Contracting issues, funding gaps, and budget shortfalls

- **Risk**: randomness with knowable probabilities, for …
  - Performance of sub-contractors and suppliers
  - Meeting necessary schedules and performance measures

- **Variations**: knowable variables and variance range, for …
  - Availability of critical test/demo facility/personnel
  - Performance and behavior differences in multiple COTS-sources

- **Evolution**: successive external developments, that …
  - Change in targeted operating environment
  - Change the Availability of superior technology matures
The Challenge of Agile Estimating

- Subjectivity of how to measure size consistently in Story Points across the organization
- A Story Point means different things to different teams
- The meaning of a Story Point changes as the project progresses
- The only *Cardinal* value(s) for a program is Time and Money
- FAR programs report in Time and Money, not Story Points
Agile Estimation Practices

- Benchmarking
- Upfront project estimation and budgeting
- Iteration planning and project re-estimation
- Process improvement monitoring
Agile Estimating Resources

- Estimating Databases
  - COSMIC
  - NESMA
  - COCOMO
  - ISBSG

- Tools
  - QSM
  - SEER
  - Price
Metrics Needed to Successfully Manage A Program

Evidence of Credible Plan at IBR

1. Key Technical Performance Measures plan(s)
2. Deliverables plan
3. Summary level of the Integrated Master Schedule (IMS) and proposed budgeted cost of work scheduled
4. Labor FTE utilization plan
5. Schedule health and performance checks
6. Risk register and mitigation actions
7. Computation of initial Management Reserves (MR)
8. Risk burn down plan
9. Computation of Schedule Margin (SM)
Metrics Needed to Successfully Manage A Program (Continued)

Periodic data Ensures C/S Performance consistent with Technical Progress

10. TPM plan vs estimated actuals vs cost and schedule performance metrics (CPI, SPI)

11. Deliverables plan vs actuals vs CPI, SPI

12. FTE plan vs actuals

13. Cumulative BCWS, BCWP, ACWP against IBR spend plan, earned schedule with percent spent, percent complete, and percent scheduled (Enhanced Gold Card)

14. Risk burn down plan vs actual

15. C/S Performance Informed by Risk Burn Down Actuals
Metrics Needed to Successfully Manage A Program (Continued)

Additional Periodic Data Identifies Current and Likely Future Problem Areas

16. Schedule health and schedule performance related data on the “go-forward” IMS (similar to view # 5)

17. Cumulative BCWS, BCWP, ACWP against IBR spend plan with Earned Schedule and status dates, percent spent, percent complete, and percent scheduled (same as # 13)

18. Tornado (or Galaxy) chart that shows the relative percentage of Budget at Complete to total for any level of WBS

19. Monthly and cumulative charts of CV, SV, CPI, SPI, SPI_t for any level of WBS element or OBS

20. Management Reserve usage and balance

21. Sources and uses of MR and Undistributed Budget

22. Changes to the Baseline (new scope or use of MR)
Metrics Needed to Successfully Manage A Program (Concluded)

Periodic Data That Indicates Current and Likely Future Problem Areas (Concluded)

22. Updated Risk Register (same as metric/view # 6)

23. Forecast of Estimate At Completion (EAC) and Estimated Completion Date (ECD)

24. Confidence level of meeting contractor best case, worst case and most likely EACs and ECDs

25. Schedule and cost crucially indices
Successful Solicitation and Evaluation (1)†

- Use Presentations as Part of the Technical Evaluation. Consider including language in the solicitation that the Government intends to require oral presentations as part of the offeror’s technical portion of its quote or proposal. This will enable the Government to determine whether an offeror truly knows Agile software development. This is not mandatory, but has proven to be effective for some agencies. Of note, oral presentations need to be tightly controlled and recorded to ensure that all offerors are treated equally, that the Government does not inadvertently open discussions, and to create a defendable record of the agency’s actions. If using oral presentations, consider using them after the competitive range is established. The Government should clearly spell out the intended use of oral presentations in the Evaluation Criteria if it chooses to use them.

† TechFAR Handbook for Procuring Digital Services Using Agile Processes
Successful Solicitation and Evaluation (2)

- Integrate Agile into the Technical Factors in the RFQ: For example,
  - Factor 1 – Performance Work Statement (“Offerors shall provide a Performance Work Statement (PWS) in response to the Statement of Objectives and this RFQ. The proposed solution shall include an explanation of how project and contract management, communication/collaboration with the Government, security and privacy requirements, documentation, and reporting will function in conjunction with the proposed Agile methodology.”);
  - Factor 2 – Product Development Roadmap (“Offerors shall propose an Agile product development roadmap which correlates how the stated objective aligns with the timeframe for implementation and the offeror’s proposed Agile methodology. The product development roadmap shall demonstrate where testing, training, security, privacy, and cut over planning, will be included.”);
  - Factor 3 – Notional Performance Control Plan (“Offerors shall describe the QC and Performance Measurement approach, including how proposed performance standards will be monitored, evaluated, and reported. The purpose of the notional QCP is to provide evaluators with an understanding of how measures and metrics will be applied based on the proposed technical solution.”)
Successful Solicitation and Evaluation (1)

- Request Agile software development-Specific Information from Offerors. As part of the technical evaluation, request information from the offerors addressing how they manage Agile implementation, techniques for release planning, plans for engaging end users, methods for capturing and applying lessons learned, testing processes, reasons behind the composition of their Agile teams and the rationale behind the proposed development talent and project oversight (tied to Product Vision), how they will make resources available within schedule and budget constraints, and their approach to configuration management.
Successful Solicitation and Evaluation (1)

- Evaluate Demonstrated Experience with Agile. As part of the past experience evaluation criterion, include demonstrated experience with successfully developing software using an Agile approach.
The Framework for Agile Performance Management using Earned Value

Starting with a Product Roadmap and Cadence Releases, Earned Value Management + Agile Integration is straightforward when progress to plan is measured as **Physical Percent Complete**

Starting with a Product Roadmap and Cadence Releases, Earned Value Management + Agile Integration is straightforward when progress to plan is measured as **Physical Percent Complete**

- **Performance Measurement Baseline**
- **Agile Software Development Lifecycle**
  - **Sprints**
  - **Milestones**
  - **Data Items**
    - Product Roadmap
    - Capabilities in a Release Plan
  - **CA**
    - **WP**
    - **SLPP in IMS**

**Agile Development Control Account**

- **Feature 1, 2, 3**
- **Feature 4, ..., 8**
- **Feature 9, ..., 12**

**BCWS (PV)** is the labor spread at the Sprint level contained in the Work Packages.

BCWP (EV) = BCWS × Physical Percent Complete of Features produced by Sprints.

**Physical Percent Complete** from Planned Stories that implement the Features in each Sprint.
Knowledge is of two kinds. We know a subject ourselves, or we know where we can find information upon it
– Samuel Johnson
Resources

- COSMIC – http://cosmic-sizing.org/
- NESMA – http://nesma.org/
- ISBSG – http://isbsg.org/
- IFPUG – http://www.ifpug.org/
- QSM – http://www.qsm.com/
- Price – http://www.pricesystems.com/
- Galorath – http://galorath.com/
Books (1)

Books (2)


- *Progressive Function Point Analysis: Advanced Estimation Techniques for IT Projects*, Ruben Gerad Mathew and Anna Bandura,
  
  - Excel Spread sheet from Source Forge,  
Papers

- Assessing COTS Integration Risk Using Cost estimation Inputs, Ye Yang, Barry Boehm, and Betsy Clark, ICSE, 2006
- “Estimate and Measure Agile Projects with Function Points,” Radenko Corovic.
Papers

- “Story Points or Function Points or Both?” David Consulting Group, July 2015
- “From Story Points to COSMIC Function Points in Agile Software Development – A Six Sigma perspective,” Thomas Fehlmann and Luca Santillo, MetriKon 2010
Papers


Papers

- “Using NESMA Function Point Analysis in an Agile Context,” Roel van Rijswijck, Radboud Universiteit Nijmegen, August 2013
- “Function Point Estimation Methods: A Comparative Overview,” Roberto Meill and Luca Santillo,
Software Size Growth Study

22 August 2017
Software & IT-CAST

Presenter: Marc Russo,
Corinne Wallshein
Naval Center for Cost Analysis
marc.russo1@navy.mil
corinne.wallshein@navy.mil
Outline

- Abstract
- Study questions
- GAO recommendation on software growth
- Data
- Percent change overview
- Uncertainty overview
- Example problem
- Correlation and subsets
- Conclusion and future research
Abstract

Software cost estimating relationships often rely on software size growth percentages.

Actual delivered source lines of code (SLOC) may be predicted with categories of early code estimates such as new, modified, reuse, and auto-generated SLOC. Uncertainty distributions will be presented to represent growth by code category for use in cost modeling.

Uncertainty distributions will be based on the actual percentage growth for Department of Defense programs’ computer software configuration items in selected data subsets.
Questions Answered by Study

• What is the growth or shrinkage for types of SLOC (New, Modified, Reused, Auto-Generated, and Total), requirements, peak staff, effort hours, and duration?

• What uncertainty should be associated with growth?

• Is requirements growth correlated to SLOC growth?

• What other areas can be explored?
Per 2009 GAO Cost Estimating and Assessment Guide:

“It is extremely important to include the expected growth in software size from requirements growth or underestimation (that is, optimism). Adjusting the software size to reflect expected growth from requirements being refined, changed, or added or initial size estimates being too optimistic and less reuse than expected is a best practice. This growth adjustment should be made before performing an uncertainty analysis [on effort or cost CERs created from actual, final reports]. Understanding software will usually grow, and accounting for it by using historical data, will result in more accurate software sizing estimates.”
Data

- Non-random sample of secondary data

- Projects reported at the CSCI level by Software Resource Data Reports on the OSD/CAPE website called Cost Assessment Data Enterprise

- Content
  - Allows for collection of project context, responsible company or government entity, certified maturity level, requirements count, product size, effort hours, and schedule
Each program submitted:

- SRDR Initial Developer Report (Estimates)
- SRDR Final Developer Report (Actuals)

• Analysis based on a subset of paired initial to final records from 2014 SRDR data set:
  - Requirements between 10 and 1000
  - Total SLOC between 100 and 1 Million
  - Effort Hours below 150,000
Data Analysis Pedigree

2624 Total CSCI Records

911 Completed Program / Build CSCI Records

403 Completed CSCIs with IEEE 12207 break-outs

219 Paired CSCI Records

129 analyzed

Since last ICEAA (2016)
Outliers and records outside analysis scope were excluded
Data Demographics (SLOC)

<table>
<thead>
<tr>
<th>Variable</th>
<th>Quantiles</th>
<th>Moments</th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Max</td>
<td>Median</td>
<td>Min</td>
<td>Mean</td>
<td>Std Dev</td>
<td>SE Mean</td>
<td>N</td>
<td>Skewness</td>
<td>Kurtosis</td>
</tr>
<tr>
<td>Initial New LS</td>
<td>192000</td>
<td>12028</td>
<td>120</td>
<td>25858</td>
<td>36928.37</td>
<td>3251.36</td>
<td>129</td>
<td>2.70</td>
<td>7.81</td>
</tr>
<tr>
<td>Final New LS</td>
<td>268800</td>
<td>18644</td>
<td>500</td>
<td>37370</td>
<td>49402.21</td>
<td>4349.62</td>
<td>129</td>
<td>2.25</td>
<td>5.47</td>
</tr>
<tr>
<td>Initial Modified LS</td>
<td>158718</td>
<td>2000</td>
<td>0</td>
<td>10548</td>
<td>25628.81</td>
<td>2256.49</td>
<td>129</td>
<td>4.33</td>
<td>20.47</td>
</tr>
<tr>
<td>Final Modified LS</td>
<td>196168</td>
<td>640</td>
<td>0</td>
<td>9463</td>
<td>25359.16</td>
<td>2232.75</td>
<td>129</td>
<td>4.99</td>
<td>29.23</td>
</tr>
<tr>
<td>Initial Reused LS</td>
<td>514800</td>
<td>7900</td>
<td>0</td>
<td>44556</td>
<td>94915.04</td>
<td>8356.80</td>
<td>129</td>
<td>3.41</td>
<td>12.18</td>
</tr>
<tr>
<td>Final Reused LS</td>
<td>617008</td>
<td>6000</td>
<td>0</td>
<td>55031</td>
<td>111247.56</td>
<td>9794.80</td>
<td>129</td>
<td>2.89</td>
<td>8.83</td>
</tr>
<tr>
<td>Initial Auto-Generated LS</td>
<td>16490</td>
<td>0</td>
<td>0</td>
<td>293</td>
<td>1940.40</td>
<td>170.84</td>
<td>129</td>
<td>6.94</td>
<td>49.39</td>
</tr>
<tr>
<td>Final Auto-Generated LS</td>
<td>213650</td>
<td>0</td>
<td>0</td>
<td>3247</td>
<td>20735.86</td>
<td>1825.69</td>
<td>129</td>
<td>8.97</td>
<td>86.71</td>
</tr>
<tr>
<td>Initial SLOC LS</td>
<td>614111</td>
<td>48237</td>
<td>904</td>
<td>81256</td>
<td>107902.35</td>
<td>9500.27</td>
<td>129</td>
<td>2.74</td>
<td>8.72</td>
</tr>
<tr>
<td>Final SLOC LS</td>
<td>818071</td>
<td>46200</td>
<td>1169</td>
<td>105111</td>
<td>141337.50</td>
<td>12444.07</td>
<td>129</td>
<td>2.63</td>
<td>8.41</td>
</tr>
</tbody>
</table>

- All data either reported in Logical Statements (LS) count or converted using the following:
  - Logical Statements (LS) = 0.66 x Non-Commented Source Statements (NCSS)
  - LS = 0.33 x Physical Source Lines of Code (SLOC)
Data Demographics (Other Variables)

<table>
<thead>
<tr>
<th>Variable</th>
<th>Quantiles</th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Max</td>
<td>Median</td>
<td>Min</td>
<td>Mean</td>
<td>Std Dev</td>
<td>SE Mean</td>
<td>N</td>
<td>Skewness</td>
<td>Kurtosis</td>
<td>CV</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Initial Effort Hours</td>
<td>133855</td>
<td>18643</td>
<td>575</td>
<td>31122.61</td>
<td>32456.7</td>
<td>2857.66</td>
<td>129</td>
<td>1.58</td>
<td>1.85</td>
<td>104.29</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Final Effort Hours</td>
<td>139786</td>
<td>27265</td>
<td>1486</td>
<td>37799.27</td>
<td>35288.9</td>
<td>3107.02</td>
<td>129</td>
<td>1.27</td>
<td>0.78</td>
<td>93.36</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Initial Requirements</td>
<td>990</td>
<td>184</td>
<td>10</td>
<td>274.19</td>
<td>260.11</td>
<td>22.90</td>
<td>129</td>
<td>1.14</td>
<td>0.30</td>
<td>94.86</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Final Requirements</td>
<td>965</td>
<td>208</td>
<td>11</td>
<td>275.53</td>
<td>246.38</td>
<td>21.69</td>
<td>129</td>
<td>1.18</td>
<td>0.65</td>
<td>89.42</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Initial Duration (Months)</td>
<td>100.11</td>
<td>20.02</td>
<td>0.23</td>
<td>20.59</td>
<td>19.67</td>
<td>1.73</td>
<td>129</td>
<td>1.11</td>
<td>1.64</td>
<td>95.57</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Final Duration (Months)</td>
<td>109.09</td>
<td>21.01</td>
<td>0.36</td>
<td>21.48</td>
<td>20.40</td>
<td>1.80</td>
<td>129</td>
<td>1.39</td>
<td>3.56</td>
<td>94.99</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Initial Peak Staff</td>
<td>71</td>
<td>8</td>
<td>1</td>
<td>11.84</td>
<td>12.57</td>
<td>1.11</td>
<td>129</td>
<td>2.27</td>
<td>5.68</td>
<td>106.19</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Final Peak Staff</td>
<td>69</td>
<td>9</td>
<td>1</td>
<td>12.24</td>
<td>11.70</td>
<td>1.03</td>
<td>129</td>
<td>2.05</td>
<td>5.14</td>
<td>95.61</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- All data either reported in Logical Statements (LS) count or converted using the following:
  - Logical Statements (LS) = 0.66 x Non-Commented Source Statements (NCSS)
  - LS = 0.33 x Physical Source Lines of Code (SLOC)
Process Overview

- From the data set have the ability to calculate percent change from initial to final using this formula:

\[ \text{Percent Change} = \frac{(Final - Initial)}{Initial} \]

- Calculations were performed on all code types, requirement counts, duration in months, effort hours, and peak staff.

- Crystal Ball batch fit capability used to determine best fit for percent change uncertainty.
Percent Change (PC) Summary
SLOC (Logical Statements [LS])

<table>
<thead>
<tr>
<th>Variable</th>
<th>Quantiles</th>
<th>Moments</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Max</td>
<td>Median</td>
</tr>
<tr>
<td>PC for New LS</td>
<td>21.90</td>
<td>0.37</td>
</tr>
<tr>
<td>PC for Modified LS</td>
<td>182.73</td>
<td>0.01</td>
</tr>
<tr>
<td>PC for Reused LS</td>
<td>24.88</td>
<td>-0.11</td>
</tr>
<tr>
<td>PC for Auto-Generated LS</td>
<td>1.01</td>
<td>-0.78</td>
</tr>
<tr>
<td>PC for Total SLOC in LS</td>
<td>18.55</td>
<td>0.05</td>
</tr>
</tbody>
</table>
### Percent Change (PC) Summary

Other Variables

<table>
<thead>
<tr>
<th>Variable</th>
<th>Quantiles</th>
<th>Moments</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Max</td>
<td>Median</td>
</tr>
<tr>
<td>PC in Duration (Months)</td>
<td>32.63</td>
<td>0.01</td>
</tr>
<tr>
<td>PC in Effort Hours</td>
<td>11.20</td>
<td>0.14</td>
</tr>
<tr>
<td>PC in Requirements</td>
<td>9.71</td>
<td>0.00</td>
</tr>
<tr>
<td>PC in Peak Staff</td>
<td>2.67</td>
<td>0.00</td>
</tr>
</tbody>
</table>
Uncertainty Overview
To ensure that uncertainty range does not provide a negative value (for Total SLOC) each distribution needs to be truncated at -1.
Uncertainty Distributions
SLOC Percent Change

- Auto-generated distribution not available due to Crystal Ball Batch Fit requiring 15 data points
  - See Data Demographic chart
Uncertainty Distributions
Other Variables Percent Change

<table>
<thead>
<tr>
<th>Requirements Count</th>
<th>Duration (Months)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Development Hours</td>
<td>Peak Staff</td>
</tr>
</tbody>
</table>

![Lognormal Distribution](image1)

- **Requirements Count**: Location: 0.93, Mean: 0.26, Std. Dev: 0.87
- **Duration**: Location: -1.18, Mean: 0.37, Std. Dev: 0.93
- **Development Hours**: Location: -0.90, Mean: 0.65, Std. Dev: 1.22
- **Peak Staff**: Likelihood: 0.04, Scale: 0.36
Example

- Program is able to provide SLOC, in logical statements, by initial New, Modified, Reuse, and Auto-Generated
- To estimate final data sizes, apply growth factors to initial data sizes
- Program Data:

<table>
<thead>
<tr>
<th>CSCI</th>
<th>New (Initial)</th>
<th>Mod (Initial)</th>
<th>Reuse (Initial)</th>
<th>Auto (Initial)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>200</td>
<td>4,699</td>
<td>31,144</td>
<td>16,490</td>
</tr>
<tr>
<td>2</td>
<td>200</td>
<td>2,236</td>
<td>22,803</td>
<td>340</td>
</tr>
<tr>
<td>3</td>
<td>3,354</td>
<td>1,147</td>
<td>67,083</td>
<td>25,660</td>
</tr>
<tr>
<td>4</td>
<td>10,000</td>
<td>15,000</td>
<td>275,000</td>
<td>1,100</td>
</tr>
</tbody>
</table>
Example cont.

• Apply formula to initial variables
  
  \[
  \text{Final} = \text{Initial} \times (1 + \text{Percent Change})
  \]

<table>
<thead>
<tr>
<th>CSCI</th>
<th>New (Initial)</th>
<th>1+ New PC</th>
<th>Mod (Initial)</th>
<th>1+ Mod PC</th>
<th>Reuse (Initial)</th>
<th>1+ Reuse PC</th>
<th>Auto (Initial)</th>
<th>1 + Auto PC</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>200</td>
<td>1+ 1.26</td>
<td>4,699</td>
<td>1 + 2.65</td>
<td>31,144</td>
<td>1 + .55</td>
<td>16,490</td>
<td>1 - .39</td>
</tr>
<tr>
<td>2</td>
<td>200</td>
<td>1+ 1.26</td>
<td>2,236</td>
<td>1 + 2.65</td>
<td>22,803</td>
<td>1 + .55</td>
<td>340</td>
<td>1 - .39</td>
</tr>
<tr>
<td>3</td>
<td>3,354</td>
<td>1+ 1.26</td>
<td>1,147</td>
<td>1 + 2.65</td>
<td>67,083</td>
<td>1 + .55</td>
<td>25,660</td>
<td>1 - .39</td>
</tr>
<tr>
<td>4</td>
<td>10,000</td>
<td>1+ 1.26</td>
<td>15,000</td>
<td>1 + 2.65</td>
<td>275,000</td>
<td>1 + .55</td>
<td>1,100</td>
<td>1 - .39</td>
</tr>
</tbody>
</table>

• Apply uncertainty (example)
Example

Results

<table>
<thead>
<tr>
<th>CSCI</th>
<th>New (Initial)</th>
<th>I+ New PC</th>
<th>New (Final)</th>
<th>Mod (Initial)</th>
<th>I+ Mod PC</th>
<th>Mod (Final)</th>
<th>Reuse (Initial)</th>
<th>I+ Reuse PC</th>
<th>Reuse (Final)</th>
<th>Auto (Initial)</th>
<th>1 + Auto PC</th>
<th>Auto (Final)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>200</td>
<td>1+ 1.26</td>
<td>451</td>
<td>4,699</td>
<td>1 + 2.65</td>
<td>17,166</td>
<td>31,144</td>
<td>1 + .55</td>
<td>48,284</td>
<td>16,490</td>
<td>1 - .39</td>
<td>10,082</td>
</tr>
<tr>
<td>2</td>
<td>200</td>
<td>1+ 1.26</td>
<td>451</td>
<td>2,236</td>
<td>1 + 2.65</td>
<td>8,168</td>
<td>22,803</td>
<td>1 + .55</td>
<td>48,284</td>
<td>340</td>
<td>1 - .39</td>
<td>208</td>
</tr>
<tr>
<td>3</td>
<td>3,354</td>
<td>1+ 1.26</td>
<td>7,571</td>
<td>1,147</td>
<td>1 + 2.65</td>
<td>4,190</td>
<td>67,083</td>
<td>1 + .55</td>
<td>48,284</td>
<td>25,660</td>
<td>1 - .39</td>
<td>15,689</td>
</tr>
<tr>
<td>4</td>
<td>10,000</td>
<td>1+ 1.26</td>
<td>22,573</td>
<td>15,000</td>
<td>1 + 2.65</td>
<td>54,795</td>
<td>275,000</td>
<td>1 + .55</td>
<td>48,284</td>
<td>1,100</td>
<td>1 - .39</td>
<td>673</td>
</tr>
</tbody>
</table>

Uncertainty

- As an example the uncertainty distribution and analysis is provided for CSCI 1 New

Uncertainty in growth levels should be applied to all CSCI factors
Program Type Break Out
## Program Type Percent Change

### SLOC Total

<table>
<thead>
<tr>
<th>Program Type</th>
<th>Percent Change</th>
<th>Mean</th>
<th>Std Dev</th>
<th>Std Err Mean</th>
<th>Upper 95% Mean</th>
<th>Lower 95% Mean</th>
<th>N</th>
</tr>
</thead>
<tbody>
<tr>
<td>Aircraft - Fixed Wing</td>
<td></td>
<td>0.426</td>
<td>1.429</td>
<td>0.312</td>
<td>1.076</td>
<td>-0.225</td>
<td>21</td>
</tr>
<tr>
<td>Aircraft - Rotary Wing</td>
<td></td>
<td>0.173</td>
<td>0.436</td>
<td>0.121</td>
<td>0.436</td>
<td>-0.090</td>
<td>13</td>
</tr>
<tr>
<td>C2-4I &amp; Other</td>
<td></td>
<td>0.790</td>
<td>1.78</td>
<td>0.215</td>
<td>1.218</td>
<td>0.361</td>
<td>69</td>
</tr>
<tr>
<td>Missiles</td>
<td></td>
<td>0.222</td>
<td>1.158</td>
<td>0.366</td>
<td>3.564</td>
<td>-5.333</td>
<td>10</td>
</tr>
<tr>
<td>Ships</td>
<td></td>
<td>0.924</td>
<td>1.066</td>
<td>0.321</td>
<td>3.564</td>
<td>-5.333</td>
<td>11</td>
</tr>
</tbody>
</table>

Mean Total SLOC percent change for all programs was 0.78
# Program Type Percent Change

## New SLOC

<table>
<thead>
<tr>
<th>Program Type</th>
<th>Percent Change</th>
<th>Mean</th>
<th>Std Dev</th>
<th>Std Err Mean</th>
<th>Upper 95% Mean</th>
<th>Lower 95% Mean</th>
<th>N</th>
</tr>
</thead>
<tbody>
<tr>
<td>ircraft- Fixed Wing</td>
<td></td>
<td>1.695</td>
<td>4.734</td>
<td>1.033</td>
<td>3.850</td>
<td>-0.460</td>
<td>21</td>
</tr>
<tr>
<td>Aircraft- Rotary Wing</td>
<td></td>
<td>1.487</td>
<td>1.607</td>
<td>0.446</td>
<td>2.458</td>
<td>0.516</td>
<td>13</td>
</tr>
<tr>
<td>C2-4I &amp; Other</td>
<td></td>
<td>1.125</td>
<td>3.135</td>
<td>0.377</td>
<td>1.879</td>
<td>-0.759</td>
<td>69</td>
</tr>
<tr>
<td>Missiles</td>
<td></td>
<td>1.625</td>
<td>3.332</td>
<td>1.054</td>
<td>4.009</td>
<td>-0.759</td>
<td>10</td>
</tr>
<tr>
<td>Radar</td>
<td></td>
<td>0.811</td>
<td>1.171</td>
<td>0.524</td>
<td>2.264</td>
<td>-0.643</td>
<td>5</td>
</tr>
<tr>
<td>Ships</td>
<td></td>
<td>0.846</td>
<td>1.180</td>
<td>0.356</td>
<td>1.639</td>
<td>0.054</td>
<td>11</td>
</tr>
</tbody>
</table>

Mean New SLOC percent change for all programs was 1.26
Mean Modified SLOC percent change for all programs was 2.65
Program Type Percent Change
Reuse SLOC

Mean Reuse SLOC percent change for all programs was 0.55
# Program Type Percent Change Duration (Months)

<table>
<thead>
<tr>
<th>Aircraft- Fixed Wing</th>
<th>Aircraft- Rotary Wing</th>
<th>C2-4I &amp; Other</th>
<th>Missiles</th>
<th>Radar</th>
<th>Ships</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mean</td>
<td>0.316</td>
<td>3.581</td>
<td>0.187</td>
<td>0.161</td>
<td>0.167</td>
</tr>
<tr>
<td>Std Dev</td>
<td>0.954</td>
<td>8.814</td>
<td>0.852</td>
<td>0.960</td>
<td>0.320</td>
</tr>
<tr>
<td>Std Err Mean</td>
<td>0.208</td>
<td>2.445</td>
<td>0.103</td>
<td>0.304</td>
<td>0.097</td>
</tr>
<tr>
<td>Upper 95% Mean</td>
<td>0.751</td>
<td>8.907</td>
<td>0.392</td>
<td>0.848</td>
<td>0.163</td>
</tr>
<tr>
<td>Lower 95% Mean</td>
<td>-0.118</td>
<td>-1.746</td>
<td>-0.018</td>
<td>-0.526</td>
<td>-0.270</td>
</tr>
<tr>
<td>N</td>
<td>21</td>
<td>13</td>
<td>69</td>
<td>10</td>
<td>11</td>
</tr>
</tbody>
</table>

Mean Duration percent change for all programs was 0.53
## Program Type Percent Change Effort (Hours)

<table>
<thead>
<tr>
<th>Program Type</th>
<th>Mean Effort</th>
<th>Std Dev</th>
<th>Std Err Mean</th>
<th>Upper 95% Mean</th>
<th>Lower 95% Mean</th>
<th>N</th>
</tr>
</thead>
<tbody>
<tr>
<td>Aircraft- Fixed Wing</td>
<td>0.863</td>
<td>2.505</td>
<td>0.547</td>
<td>2.003</td>
<td>-0.277</td>
<td>21</td>
</tr>
<tr>
<td>Aircraft- Rotary Wing</td>
<td>0.665</td>
<td>0.946</td>
<td>0.262</td>
<td>1.236</td>
<td>0.093</td>
<td>13</td>
</tr>
<tr>
<td>C2-4I &amp; Other</td>
<td>0.864</td>
<td>1.902</td>
<td>0.229</td>
<td>1.321</td>
<td>0.047</td>
<td>69</td>
</tr>
<tr>
<td>Missiles</td>
<td>0.425</td>
<td>0.445</td>
<td>0.261</td>
<td>0.744</td>
<td>-0.299</td>
<td>10</td>
</tr>
<tr>
<td>Radar</td>
<td>0.025</td>
<td>0.242</td>
<td>0.073</td>
<td>0.340</td>
<td>0.014</td>
<td>11</td>
</tr>
<tr>
<td>Ships</td>
<td>0.177</td>
<td>0.242</td>
<td>0.073</td>
<td>0.340</td>
<td>0.014</td>
<td>11</td>
</tr>
</tbody>
</table>

Mean Effort percent change for all programs was 0.72
## Program Type Percent Change Requirements

<table>
<thead>
<tr>
<th>Program Type</th>
<th>Percent Change</th>
<th>Mean</th>
<th>Std Dev</th>
<th>Std Err Mean</th>
<th>Upper 95% Mean</th>
<th>Lower 95% Mean</th>
<th>N</th>
</tr>
</thead>
<tbody>
<tr>
<td>Aircraft- Fixed Wing</td>
<td>-0.277</td>
<td>0.055</td>
<td>0.119</td>
<td>0.033</td>
<td>0.327</td>
<td>-0.499</td>
<td>21</td>
</tr>
<tr>
<td>Aircraft- Rotary Wing</td>
<td>-0.017</td>
<td>0.542</td>
<td>1.942</td>
<td>0.234</td>
<td>2.011</td>
<td>-1.664</td>
<td>13</td>
</tr>
<tr>
<td>C2-4I &amp; Other</td>
<td>0.448</td>
<td>0.721</td>
<td>0.142</td>
<td>0.037</td>
<td>0.998</td>
<td>0.447</td>
<td>69</td>
</tr>
<tr>
<td>Missiles</td>
<td>0.037</td>
<td>0.272</td>
<td>1.473</td>
<td>0.189</td>
<td>0.606</td>
<td>-0.335</td>
<td>5</td>
</tr>
<tr>
<td>Radar</td>
<td>0.588</td>
<td>0.371</td>
<td>0.100</td>
<td>0.030</td>
<td>0.795</td>
<td>-0.384</td>
<td>11</td>
</tr>
</tbody>
</table>

Mean Requirements percent change for all programs was 0.36
Mean Peak Staff percent change for all programs was 0.17
Additional Explorations
Requirements and SLOC

- Are Requirements and SLOC correlated?
- The data set shows no correlation between total SLOC change and requirements change though they both increase
- A second look, removing items with requirements count over 200, shows similar trend

Continued analysis into how requirements growth is related to SLOC should be conducted
New/Upgrade Percent Change ANOVA Analysis

Oneway Anova Summary of Fit

<table>
<thead>
<tr>
<th></th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>R²</td>
<td>0.053</td>
</tr>
<tr>
<td>Adjusted R²</td>
<td>0.045</td>
</tr>
<tr>
<td>Root Mean Square Error</td>
<td>2.149</td>
</tr>
<tr>
<td>Mean of Response</td>
<td>0.782</td>
</tr>
<tr>
<td>Observations (or Sum Wgts)</td>
<td>129</td>
</tr>
</tbody>
</table>

Mean difference for SLOC percent change for New versus Upgrade is pronounced.

Means for Effort Hours percent change for New versus Upgrade are similar.
Conclusion and Future Research

• From this analysis, Percent Change averages and uncertainties are available to estimate growth and cross check software cost estimates

• Based on the 129 data points, requirements growth is not directly correlated to Total SLOC growth
  – Mean percent change for both requirements and Total SLOC grows

• Percent change analysis should be updated and improved as more data becomes available

• Analysis on software size growth will be continued
Questions?
Adapting a classic Independent Cost Estimation (ICE) Cost Shop for Agile and DevOPS estimates

Software & IT CAST
August 2017

David P. Seaver
Senior Technical Analyst
National Security Agency
Outline

- Definitions
- Changes to Process
- Changes to Data Collection
- Problem and Solution
  - Business Systems
  - Analytics
  - Infrastructure Projects
DevOPS

• **DevOps** (development and operations) is an enterprise software development phrase used to mean a type of agile relationship between development and IT operations. The goal of **DevOps** is to change and improve the relationship by advocating better communication and collaboration between these two business units.

• Under a DevOps model, development and operations teams are no longer “siloed.”
  – Sometimes, these two teams are merged into a single team where the engineers work across the entire application lifecycle, from development and test to deployment to operations, and develop a range of skills not limited to a single function.

• These teams use practices to automate processes that historically have been manual and slow.
  – They use a technology stack and tooling which help them operate and evolve applications quickly and reliably.
  – These tools also help engineers independently accomplish tasks (for example, deploying code or provisioning infrastructure) that normally would have required help from other teams, and this further increases a team’s velocity.

• For more information (since I am not defining agile today for you)
DevOPS....

• Agency has been reorganized around a DevOPS perspective
  – Development and Operations merged into same organization
  – Architecture consolidated onto a common, managed cloud platform
  – New oversight/governance based on a new requirements process
    • Strategic goal suitable for board of directors and senior management oversight and approval
    • Initiative
      – Epic
        » Outcome
          • Story
          • Story
          • Story
Elementary Process
(It’s a Function Point Thing)

• Elementary Process: represent the smallest whole unit of work that is meaningful to the user (any person or thing that interacts with the application).
Part 2

CHANGES TO ESTIMATION PROCESS
NSA Process to Estimate Software

1. Identify the boundary of the application
   - What data is maintained by the application
   - What data feeds need to be accommodated
   - External data sources

2. Count elementary processes
   - Create, Update, Delete, Report, Read/Query

3. Count data groups
   - Maintained by elementary process (typically create)
   - Data in other applications that is utilized to support an elementary process

4. Enter information in SFP Toolkit

5. Calculate Software Size, Effort and Schedule

6. Review with Stakeholders and revise as needed

7. Calibration

     Evaluate what you have done before!

8. Start here

   Description of Solution

   - If its red its changed
Changes to Estimation Process

• Description of the Solution:
  – No longer getting Functional Requirements Documents, the descriptions of capabilities we are receiving are not detailed enough to provide reliable cost estimates

• Review with Stakeholders: This issue overlaps with the lack of detailed requirements. It can be difficult to identify all the stakeholders
  – Users
  – Development team
  – Sponsor
Example

- **Initiative:** Toolbox for conducting business
  - Establish a modern relationship management and business intelligence platform for Agency officers featuring integrated capabilities for effectively anticipating and compliantly addressing customer needs. Provide a drastically improved, simplified and streamlined digital experience through an intuitive interface that continuously adapts to the officers evolving needs and preferences
  - **Epic**
    - System enables officer to effectively expand relationship management (manage partnerships, activities and strategies to align with customer needs) by understanding what was is, and could be (schedule and track exchanges, visits agreements etc)
    - **Outcome:** Manage information for a 21st century mission by expanding relationships, coordinating activities, aligning strategies and tracking the outcomes to maximize the value and impact of partnerships while protecting sensitive equities
      - **Story:** I need to know the value of my partner’s resources to my organization in order to identify opportunities for maintaining, optimizing and broadening relationships
        - **Story:**
        - **Story:**
### Requirement Text

**Initiative:** Toolbox for conducting business

- Establish a modern relationship management and business intelligence platform for Agency officers featuring integrated capabilities for effectively anticipating and compliantly addressing customer needs. Provide a drastically improved, simplified and streamlined digital experience through an intuitive interface that continuously adapts to the officers evolving needs and preferences.

**Epic**

- System enables officer to effectively expand relationship management (manage partnerships, activities and strategies to align with customer needs) by understanding what was is, and could be (schedule and track exchanges, visits agreements etc).

- Outcome: Manage information for a 21st century mission by expanding **relationships**, coordinating **activities**, aligning **strategies** and tracking the **outcomes** to maximize the value and impact of **partnerships** while protecting sensitive equities.

>> Story: I need to know the **value of my partner’s resources** to my organization in order to identify **opportunities** for maintaining, optimizing and broadening **relationships**.
Words that define Elementary Processes

Elementary Process: Represent the smallest whole unit of work that is meaningful to the user (any person or thing that interacts with the application). They are transactions that move data, or data that’s at rest.

<table>
<thead>
<tr>
<th>Accept</th>
<th>Import</th>
<th>Interface</th>
<th>Detect</th>
</tr>
</thead>
<tbody>
<tr>
<td>Add</td>
<td>Ingest</td>
<td>Provide</td>
<td>Display</td>
</tr>
<tr>
<td>Adjust</td>
<td>Inputs</td>
<td>Track</td>
<td>Distribution</td>
</tr>
<tr>
<td>Apply</td>
<td>Link</td>
<td>Browse</td>
<td>Export</td>
</tr>
<tr>
<td>Assign</td>
<td>Log</td>
<td>Enquire</td>
<td>Generate</td>
</tr>
<tr>
<td>Associate</td>
<td>Maintain</td>
<td>Extract</td>
<td>Identify</td>
</tr>
<tr>
<td>Change</td>
<td>Make Inactive</td>
<td>Inquire</td>
<td>Inform</td>
</tr>
<tr>
<td>Combine</td>
<td>Manage</td>
<td>List</td>
<td>Knowledge</td>
</tr>
<tr>
<td>Create</td>
<td>Modify</td>
<td>Pick List</td>
<td>Measure</td>
</tr>
<tr>
<td>Data Source</td>
<td>Provenance</td>
<td>View</td>
<td>Outputting</td>
</tr>
<tr>
<td>Delete</td>
<td>Purge</td>
<td>Allocate</td>
<td>Report</td>
</tr>
<tr>
<td>Enrich</td>
<td>Smart Data Tagging</td>
<td>Analyze</td>
<td>Tabulate</td>
</tr>
<tr>
<td>Enter</td>
<td>Store</td>
<td>Correlate</td>
<td></td>
</tr>
</tbody>
</table>
Part 3

CHANGES TO DATA COLLECTION
## Data Averages

<table>
<thead>
<tr>
<th>Metric</th>
<th>Average Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Function Points/ Requirement</td>
<td>24</td>
</tr>
<tr>
<td>Hours/Function Point</td>
<td>8</td>
</tr>
<tr>
<td>Hours/SLOC</td>
<td>0</td>
</tr>
<tr>
<td>Function Points</td>
<td>1,628</td>
</tr>
<tr>
<td>Requirements</td>
<td>112</td>
</tr>
<tr>
<td>Hours</td>
<td>43,901</td>
</tr>
<tr>
<td>Person Months</td>
<td>283</td>
</tr>
<tr>
<td>FTE (for a year)</td>
<td>24</td>
</tr>
<tr>
<td>SLOC</td>
<td>79,142</td>
</tr>
<tr>
<td>Function Point/ PM</td>
<td>6</td>
</tr>
<tr>
<td>SLOC/ Hour</td>
<td>2</td>
</tr>
</tbody>
</table>
## Programming Languages

<table>
<thead>
<tr>
<th>Language</th>
<th>%</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bash</td>
<td>2.5%</td>
</tr>
<tr>
<td>C_CPP</td>
<td>13.3%</td>
</tr>
<tr>
<td>CSS</td>
<td>3.5%</td>
</tr>
<tr>
<td>HTML</td>
<td>14.0%</td>
</tr>
<tr>
<td>Java</td>
<td>41.0%</td>
</tr>
<tr>
<td>JavaScript</td>
<td>10.2%</td>
</tr>
<tr>
<td>JSP</td>
<td>0.4%</td>
</tr>
<tr>
<td>Python</td>
<td>2.3%</td>
</tr>
<tr>
<td>XML</td>
<td>9.6%</td>
</tr>
<tr>
<td>Ruby</td>
<td>0.1%</td>
</tr>
<tr>
<td>SQL</td>
<td>1.8%</td>
</tr>
<tr>
<td>Perl</td>
<td>1.2%</td>
</tr>
</tbody>
</table>
## Code Data

<table>
<thead>
<tr>
<th>Category</th>
<th>Developed</th>
<th>Duplicate</th>
<th>GOTS/COTS/FOSS</th>
<th>Test</th>
<th>AutoGen</th>
<th>Totals</th>
</tr>
</thead>
<tbody>
<tr>
<td>SLOC</td>
<td>4,061,083</td>
<td>6,407,518</td>
<td>4,907,975</td>
<td>2,170,031</td>
<td>120,297</td>
<td>17,666,904</td>
</tr>
<tr>
<td>%</td>
<td>23%</td>
<td>36%</td>
<td>28%</td>
<td>12%</td>
<td>1%</td>
<td></td>
</tr>
<tr>
<td>Median</td>
<td>126,594</td>
<td>1,108</td>
<td>17,616</td>
<td>19,523</td>
<td>33,201</td>
<td></td>
</tr>
<tr>
<td>Min</td>
<td>468</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>111</td>
<td>-</td>
</tr>
<tr>
<td>Max</td>
<td>743,846</td>
<td>2,664,416</td>
<td>2,059,046</td>
<td>794,658</td>
<td>-</td>
<td></td>
</tr>
</tbody>
</table>
Part 4

PROBLEM AND SOLUTION
The dilemma

- We are getting capability statements/mission needs and proposed head counts
- Senior management wants to manage the portfolio at the missions needs statement level
- We could predict how much code we can create using historical data
  - But that’s not really a satisfying solution
  - I cannot really relate that back to the mission need
  - And its difficult to measure progress using source code
Proposed Solution

• We have on average ~ 24 Function Points per Requirement.
• From our historical data we know on average that:
  – 20% of the requirements are for Data entities
  – 80% of the requirements are for Transactions
• So each requirement can account for roughly
  – 5 transactions; or
  – 4 transactions and 1 Data entity
• The development team provides us with high level mission needs statements and proposed FTE to implement the capability
Proposed Solution version 1

<table>
<thead>
<tr>
<th>Requirement</th>
<th>Tcount</th>
<th>Dcount</th>
<th>Proposed FTE</th>
<th>Hours (1880)</th>
<th>Function Points ((\text{Hours} / 8))</th>
<th>Revised Tcount *</th>
<th>Revised Dcount **</th>
<th>Estimated Requirements ((\text{FP}/24))</th>
</tr>
</thead>
<tbody>
<tr>
<td>–Outcome: Manage information for a 21st century mission by expanding relationships, coordinating activities, aligning strategies and tracking the outcomes to maximize the value and impact of partnerships while protecting sensitive equities</td>
<td>20</td>
<td>4</td>
<td>8</td>
<td>15,040</td>
<td>1,880</td>
<td>301</td>
<td>54</td>
<td>78</td>
</tr>
<tr>
<td>»Story: I need to know the value of my partner’s resources to my organization in order to identify opportunities for maintaining, optimizing and broadening relationships</td>
<td>15</td>
<td>3</td>
<td>10</td>
<td>18,800</td>
<td>2,350</td>
<td>376</td>
<td>67</td>
<td>98</td>
</tr>
</tbody>
</table>

- Version 1 is a generic approach
- We are working on a more sophisticated model based on different metrics for business systems, analytics and infrastructure projects
  - Our preliminary analysis indicates that the ration of transactions to data entities changes for each of those types of projects
## Proposed Solution version 1-2

<table>
<thead>
<tr>
<th>Requirement</th>
<th>Tcount</th>
<th>Dcount</th>
<th>Propsed FTE</th>
<th>Hours (1880)</th>
<th>Function Points (Hours /8)</th>
<th>Revised Tcount</th>
<th>Revised Dcount</th>
<th>Estimated Requirements (FP/24)</th>
</tr>
</thead>
<tbody>
<tr>
<td>–Outcome: Manage information for a 21st century mission by expanding relationships, coordinating activities, aligning strategies and tracking the outcomes to maximize the value and impact of partnerships while protecting sensitive equities</td>
<td>20</td>
<td>4</td>
<td>8</td>
<td>15,040</td>
<td>1,880</td>
<td>301</td>
<td>54</td>
<td>78</td>
</tr>
<tr>
<td>»Story: I need to know the value of my partner’s resources to my organization in order to identify opportunities for maintaining, optimizing and broadening relationships</td>
<td>15</td>
<td>3</td>
<td>10</td>
<td>18,800</td>
<td>2,350</td>
<td>376</td>
<td>67</td>
<td>98</td>
</tr>
</tbody>
</table>

- * Transactions are typically 80% of our Function point size, to estimate transactions we take 80% of total Function Points, then divide by 5 (multiplier for transactions)
- ** Data is typically 20% of our Function point size. To estimate data entities we take 20% of total Function Points and divide by 7 (multiplier for transactions)
Next Steps

• My group has approximately 200 projects to evaluate over the next year
• We will be applying this initial model and revising it as we get more data
• We are completing a pilot with the CAST AFP tool and will be adding that to the data collection to add delivered function points and enhancement function points to the database
Questions
Assessing Enterprise Resource Planning (ERP) Cost, Schedule and Size Growth

22 August 2017
Software & Information Technology-Cost Analysis Solutions Team (Software & IT-CAST)

Authors:
Haset Gebre-Mariam
Abishek Krupanand
Robert Williams
Outline

• Introduction
• ERP Overview
• Data Analysis Approach
• Data Demographics
• Cost and Schedule Growth
• Benchmarks
• Conclusion
Program Office estimates of Enterprise Resource Planning (ERP) implementation costs and schedules are inaccurate, despite increased oversight.

All major DoD ERP deployed programs experienced:
- Cost Growth
- Schedule Delays

As of Dec. 2016, DoD has invested more than $16B in their deployed nine ERP programs!
Purpose of Study

- Analyze performance of nine (9) ERP programs in terms of cost and schedule growth at each Authority to Proceed (ATP) event

- Establish cost and schedule benchmarks to crosscheck early estimates, such as Business Case Analysis and/or Special Studies
Overview
Enterprise Resource Planning (ERP) systems are typically **commercial software systems** that **integrate** an organization’s **core business functions** around a **unified data base**.

ERP definition, in terms of cost characteristics, is related to the **scope and integration of multiple business systems/processes**.

If a program is not labeled an ERP, it still may be one.
How is ERP implemented?

Business processes are automated via an integrated COTS software application:

Integration is typically done by a 3rd Party Vendor.
DoD Acquisition Cycle
Current vs Future*

System Acquisition

Current
- Materiel Solution Analysis
- Technology Development
- Full Scale Development
- Materiel Development Decision
- PDR
- Contract Award
- CDR
- Post-CDR Assessment

Future (Sep-2017)
- ATP1 - Capability Needs Identification
- ATP2 - Business Solution Development
- ATP3 - Business System Functional Requirements & Acquisition Planning
- ATP4 - Business System Acquisition, Testing & Deployment
- ATP5 - Capability Support
- ATP

Sustainment
- IOC
- FD
- Operations & Support
- Go-Live
- Disposal

Other Key Decisions/Reviews

Milestone / ATP

Phases

Adapted from DoDI 5000.02, November 26, 2013, pp. 5, Figure 1
*Adapted from DoDI 5000.75, February 2, 2017, pp 5, Figure 1

New Defense Business System (DBS) Acquisition Cycle uses the Authority to Proceed (ATP) decision points roughly equivalent to Milestones in the previous DoDI release

PDR = Preliminary Design Review; CDR = Critical Design Review; IOC = Initial Operational Capability; FD = Full Deployment;
Business System Acquisition, Testing & Deployment

Business Capability Acquisition Cycle (Future)*

Solution Analysis | Functional Requirements | Acquisition | Limited Deployment | Full Deployment | Capability Support

ATP1 | ATP2 | ATP3 | ATP4 | ATP5 | ATP


Market Research
Process ------------------> IT

Authority to Proceed (ATP) are “milestone-like events”

IT Solution
Approach ------------------> Selection

IT Requirements
Functional Requirements----------> Design Specification

<----------------------------------------------------------------------------------------------- Organizational Change Management -->

*Adapted from DoDI 5000.75, February 2, 2017, pp 5, Figure 1
Data Analysis Approach
• Dataset normalized to “account for sizing units, application complexity, and content so they are consistent for comparisons” (source: GAO)
Data Sources

Cost, Schedule, and Technical Data from Authoritative Sources:

**Cost**
- Approved Cost Estimate
- Final Cost Model

**Schedule**
- MAIS Annual Report (MAR)
- MAIS Quarterly Report

**Technical**
- Cost Analysis Requirements Document (CARD)
- Software Resources Data Report (SRDR)

Data analysis is based on nine ERP deployed programs


http://www.acq.osd.mil/damir/
Cost Assumptions

System Acquisition*
* Acquisition includes all associated costs from Solution Analysis ATP throughout Full Deployment ATP

Cost Elements*  
- Design/Configuration/Customization
- Program Management
- Systems Engineering
- Change Management
- Training Development
- Development Test & Evaluation  
- Deployment Software Licenses
- Deployment Hardware Procurement  
- User Training
- Site Installation/Activation
- Data Conversion
- Execution Cut-over
- Interim Sustainment
- Operational Test & Evaluation

Assumptions  
- Cost in Base Year 2016 Dollars

*Adapted from MIL-STD-881D Appendix K (unpublished draft as of March 6, 2017)  
Authority to Proceed (ATP) are “milestone-like events”
Schedule Assumptions
Current vs. Future Acquisition Process

Current Acquisition Cycle
DoDI 5000.02

Milestone A

Milestone B

Milestone C

Full Deployment Decision

Full Deployment

Future Acquisition Cycle
DoDI 5000.75

Solution Analysis ATP

Functional Requirements ATP

Acquisition ATP

Limited Deployment ATP

Full Deployment ATP

ATP = Authority to Proceed
Data Demographics
Project Characteristics

DoD Component
- DoD: 22%
- NAVY: 22%
- ARMY: 45%
- AIR FORCE: 11%

Functional Area
- Financial: 33%
- Logistics: 34%
- HR: 11%
- Multiple: 22%

Program Heritage
- New: 33%
- Follow-on: 67%
- Analysis based on 9 deployed ERP programs
Acquisition cost includes development, procurement, and fielding costs.

All programs experienced Acquisition cost growth from Solution Analysis ATP to Full Deployment.

- 80% of programs between 50 and 115 months
- Median Development Duration: 39 months
- Median Deployment Duration: 53 months

Average ERP acquisition costs ~ $0.9 billion, with 70% of the programs ranging between $0.6 B and $1.9 B

~60% of the programs experienced critical breach for time (failure to meet Limited Deployment ATP within five years of Solution Analysis ATP)

FD = Full Deployment Authority to Proceed (ATP)
Technical Requirements at FD

- RICE Counts median: 413
- User median: 26,600

RICE: Reports, Interfaces, Conversions, Extensions
Majority of Deployed ERP systems have fewer than 40,000 Users
Cost Growth
Cost Growth Overview

**Milestones**
- ATP1
- ATP2
- ATP3
- ATP4
- ATP5

**Activities**
- Development
- Procurement
- Fielding

**Cost Group**
- Development
- Deployment

**Cost Elements included**
- ERP Configuration/Customization
- Program Management
- Systems Engineering
- Change Management
- Training Development
- Development Test & Evaluation
- Deployment Software Licenses
- Deployment Hardware Procurement
- User Training
- Site Installation/Activation
- Data Conversion
- Execution Cut-over
- Interim Sustainment
- Operational Test & Evaluation

**Key Metric:**
- Development Cost Growth
- Procurement Cost Growth
- Fielding Cost Growth

Use as secondary method to adjust point estimate for cost growth

Use descriptive statistics (as last resort) for defining cost risk/uncertainty bounds
Development Cost Growth (Planned to Actual at each ATP)

- Delays were triggered by ERP software customization, including scope creep and re-work.
- Cost growth in ATP1 and ATP2 was primarily driven by schedule delays.
- Schedule delays extend the “standing-Army” personnel, up to 50% of total development cost.
Procurement Cost Growth (Planned to Actual at each ATP)

Lower procurement cost volatility is attributed to stable user counts and negotiated license fees.
Acquisition Cost Growth (Planned to Actual at each ATP)

Acquisition Cost includes Development, Procurement and Fielding costs
Reasons for Cost Growth

1. Failure to implement Business Process Reengineering (BPR) best practices: Difficult to change business processes / culture to exploit ERP strengths.

2. Scope and requirement growth: Inexperience with Oracle/SAP customization and configuration

3. Optimistic acquisition planning contributed to underestimation of both effort and duration.

4. Schedule: Limited budgets forced delays and extended fixed staffing cost; not meeting user expectations generated unanticipated rework.
Schedule Growth
**Schedule Growth Overview**

**Milestones**
- ATP1
- ATP2
- ATP3
- ATP4
- ATP5

**Activities**
- Development
- Procurement
- Fielding

**Phases**
- Development Phase
- Deployment Phase

**Key Schedule metrics:**
- Program Length (in months)
- Program Length (in months)

**What does this measure?**
- Actual vs Planned Duration (at ATP1 or ATP2)
- Actual vs Planned Duration (at ATP4 or ATP5)

**Usefulness**
- To adjust deployment duration using the schedule growth factors
- For defining schedule risk/uncertainty bound
Deployed ERP programs have slipped an average of 47 months from the original schedule, ranging between 9 to 97 months.
At Functional Requirements ATP, deployed ERP programs experienced an average of 25 months schedule slip. Schedule slip is lower than at Solution Analysis ATP as scope is better defined/identified.
Reasons for Schedule Growth

1. Premature fielding: Failing to meet user expectations generated unanticipated rework.

2. Developmental Testing: Significant system deficiencies to fix before fielding.

3. Engineering: Inexperienced with Oracle/SAP Configuration and Customization led to underestimation of delivery timeline. Difficulty changing business processes to exploit ERP.

4. Quantity: War-fighter needs led some program offices to reassess user and implementation requirements.

5. Schedule Uncertainty Analysis: Recommended now, but in the past, Program Office’s optimistic schedule was a ground rule.
Cost Benchmarks
### Cost Factors Overview

#### Activities

<table>
<thead>
<tr>
<th>Solution Analysis</th>
<th>Functional Requirements</th>
<th>Acquisition</th>
<th>Limited Deployment</th>
<th>Full Deployment</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>ATP1</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>ATP2</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>ATP3</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>ATP4</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>ATP5</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

#### Cost Elements

- **Development**
  - Design/Configuration/Customization
  - Program Management
  - Systems Engineering
  - Change Management
  - Training Development
  - Development Test & Evaluation
- **Procurement**
  - Deployment Software Licenses
  - Deployment Hardware Procurement
- **Fielding**
  - User Training
  - Site Installation/Activation
  - Data Conversion
  - Execution Cut-over
  - Interim Sustainment
  - Operational Test & Evaluation

#### Key Cost metrics:

- Development Cost per RICE*
- Development Cost per Requirement
- Procurement Cost per User
- Fielding Cost per User

#### What this measures?

- Volume of development work units addressed by a number of either RICE or requirement
- IT Hardware and Software License Costs addressed by a number of users
- Volume of deployment & fielding work units addressed by a number of users

#### Rationale for metric

- Interfaces and requirements often available at Solution Analysis ATP
- RICE often available at Functional Requirements ATP
- Number of users are available at early ATP and tends to be stable throughout life cycle
- Number of users are available at early ATP and tends to be stable throughout life cycle

---

*Authority to Proceed (ATP) are “milestone-like events”

*RICE = reports, interfaces, conversions, and extensions of software objects*
Development Cost per RICE

Formula:

\[
\text{Development Cost Factor} = \frac{(\text{Cost})_{FD}}{(\text{RICE})_{ATP1}} \quad \frac{(\text{Cost})_{FD}}{(\text{RICE})_{ATP2}} \quad \frac{(\text{Cost})_{FD}}{(\text{RICE})_{FD}}
\]

\(\text{COST}_{FD} = \) Actual Development Cost at FD; \(\text{RICE}_{ATP1} = \) Estimated RICE at ATP1; \(\text{RICE}_{ATP2} = \) Estimated RICE at ATP2; \(\text{RICE}_{FD} = \) Actual RICE at FD
Development Cost per Requirement

Formula:

Development Cost Factor = \frac{(Cost)_{FD}}{(REQ)_{ATP1}} \quad \frac{(Cost)_{FD}}{(REQ)_{ATP2}} \quad \frac{(Cost)_{FD}}{(REQ)_{FD}}

COST_{FD} = \text{Actual Development Cost at FD}; \quad REQ_{ATP1} = \text{Estimated Requirements at ATP1}; \quad REQ_{ATP2} = \text{Estimated Requirements at APT2}; \quad REQ_{FD} = \text{Actual Requirements at FD}
Procurement Cost per User

Procurement Cost Factors

Formula:

\[ \text{Procurement Cost Factor} = \frac{(\text{Cost})_{FD}}{(\text{User})_{ATP1}} \cdot \frac{(\text{Cost})_{FD}}{(\text{User})_{ATP2}} \cdot \frac{(\text{Cost})_{FD}}{(\text{User})_{FD}} \]

\( \text{COST}_{FD} = \text{Actual Procurement Cost at FD} \); \( \text{USER}_{ATP1} = \text{Estimated users at ATP1} \); \( \text{USER}_{ATP2} = \text{Estimated users at ATP2} \); \( \text{USER}_{FD} = \text{Actual users at Full Deployment} \)
Schedule Benchmarks
# Schedule Factors Overview

## Key Schedule metrics:

<table>
<thead>
<tr>
<th>Development Phase</th>
<th>Deployment Phase</th>
</tr>
</thead>
<tbody>
<tr>
<td>RICE per Month</td>
<td>Users per Month</td>
</tr>
<tr>
<td>Requirements per Month</td>
<td></td>
</tr>
</tbody>
</table>

## What this measures?

<table>
<thead>
<tr>
<th>Development Phase</th>
<th>Deployment Phase</th>
</tr>
</thead>
<tbody>
<tr>
<td>Development phase duration using number of RICE or requirements</td>
<td>Deployment phase duration based on the number of users</td>
</tr>
</tbody>
</table>

## Rationale for metric

<table>
<thead>
<tr>
<th>Development Phase</th>
<th>Deployment Phase</th>
</tr>
</thead>
<tbody>
<tr>
<td>Interfaces and requirements often available at Solution Analysis ATP RICE often available at Functional requirements ATP</td>
<td>Number of users often available at Solution Analysis ATP Percent change in user count is very low throughout FD</td>
</tr>
</tbody>
</table>
RICE per Development Months

**Formula:**

\[
\text{Development Schedule Factor} = \frac{(RICE)_{ATP1}}{(Month)_{FD}} \quad \frac{(RICE)_{ATP2}}{(Month)_{FD}} \quad \frac{(RICE)_{FD}}{(Month)_{FD}}
\]

\(MONTH_{FD} = \text{Actual Development Duration at FD}; \quad RICE_{ATP1} = \text{Estimated RICE at ATP1}; \quad RICE_{ATP2} = \text{Estimated RICE at ATP2}; \quad RICE_{FD} = \text{Actual RICE at Full Deployment}
Requirements per Development Months

Formula:

\[
\text{Development Schedule Factor} = \frac{(REQ)_{ATP1}}{(Month)_{FD}} = \frac{(REQ)_{ATP2}}{(Month)_{FD}} = \frac{(REQ)_{FD}}{(Month)_{FD}}
\]

MONTH\textsubscript{FD} = Actual Development Duration at FD; \quad REQ\textsubscript{ATP1} = Estimated requirements at ATP1; \quad REQ\textsubscript{ATP2} = Estimated requirements at ATP2; \quad REQ\textsubscript{FD} = Actual requirements at FD
Users per Deployment Months

Formula:

\[
\text{Deployment Schedule Factor} = \frac{\text{USER}_{\text{ATP1}}}{(\text{Month})_{\text{FD}}} \times \frac{\text{USER}_{\text{ATP2}}}{(\text{Month})_{\text{FD}}} \times \frac{\text{USER}_{\text{FD}}}{(\text{Month})_{\text{FD}}}
\]

MONTH_{FD} = Actual Deployment Duration at FD; \quad \text{USER}_{\text{ATP1}} = \text{Estimated users at ATP1}; \quad \text{USER}_{\text{ATP2}} = \text{Estimated users at ATP2}; \quad \text{USER}_{\text{FD}} = \text{Actual requirements at FD}
Conclusion
Primary Findings

• All major deployed ERP programs in DoD experienced cost and schedule growth from initial estimates
  ▪ Actual data suggests cost and duration are always underestimated at ATP1 and ATP2

• Cost and schedule overruns were each over 100% from Solution Analysis ATP

• Most ERP programs exceeded five years guideline to limited deployment from Solution Analysis ATP

• Deployment Schedule overruns were greater than Development overruns
Lessons Learned

- Adjust your point estimate for growth, as all ERP programs have exceeded original estimates, account for the uncertainty

- Add growth according to the program’s maturity

- Cost factors should be developed using initial size estimates to minimize estimating error and account for growth

- Cost analysts should add uncertainty to schedule as it is the primary contributor to cost overruns
QUESTIONS?

Thank you for your attention
Objective SLOC: An Alternative Method to Sizing Software Development Efforts

Andrew Kicinski
Integrity Applications Incorporated
NRO/Cost and Acquisition Assessment Group (CAAG)

SW and IT-CAST September 2017
Agenda

- Software Estimating Today: the ESLOC Method
- Need for a New Approach
- ESLOC Alternatives – OSLOC (Objective SLOC) and Parametric Models
- Future of Software Estimating

**BLUF:** A parametric model and an estimate by analogy approach have been developed to provide a more objective, simplified and defendable software development cost estimate.
How Software Development Effort is Measured

- Level of Effort
- Function Points
- Source Lines of Code (SLOC)
- Commercial Models – SEER SEM, COCOMO, SLIM, Price
How Software Development Effort is Measured at the CAAG

✦ Equivalent Source Lines of Code (ESLOC)
  ✦ Primary method of software (SW) estimating by NRO CAAG
  ✦ A proxy for effective software development effort
  ✦ Standardizes new and reuse code to a single effective measure
    ✦ Assumes effort to reuse SW is less than or equal to new SW development
  ✦ Derived from commercial standards

\[
ESLOC = \text{New} + 0.25 \times \text{Autogen} + (\text{Unmodified} + \text{Modified}) \times \% \text{Rework}
\]

where

\[
\% \text{Rework} = (0.4 \times \% \text{RD}) + (0.25 \times \% \text{RI}) + (0.35 \times \% \text{RT})
\]

\[
\% \text{RD} = \% \text{Redesign}
\]
\[
\% \text{RI} = \% \text{Reimplementation}
\]
\[
\% \text{RT} = \% \text{Retest}
\]
ESLOC Alternative Analysis

- The CAAG recognizes the weakness of the current ESLOC method is rooted in the subjective RD/RI/RT inputs

\[
ESLOC = \text{New} + 0.25 \times \text{Autogen} + (\text{Unmodified} + \text{Modified}) \times (0.4 \times \%RD + 0.25 \times \%RI + 0.35 \times \%RT)
\]

- The “ESLOC Alternative Analysis” study was recently implemented to assess **objective** alternatives to ESLOC

- Goals of this study were:
  - Evaluate the current ESLOC method
  - Propose and develop new objective measures for estimating effective SW size
  - Assess viability and compare performance of objective measures to ESLOC
  - Recommend path forward for CAAG SW estimating team
ESLOC Advantages

- ESLOC allows the scaling of reuse code based on the expected or observed effort to use the existing software
- Higher RD/RI/RT values should accompany more effort to utilize pre-existing code

Lower RD/RI/RT
- Internal reuse
- Non-mission critical SW
- Mature reuse baseline

Higher RD/RI/RT
- External reuse
- Mission critical SW
- Low-maturity reuse

Example (perspective of SME populating SW datasheets):

<table>
<thead>
<tr>
<th>Logical SLOC</th>
<th>SOURCE</th>
<th>LANGUAGE</th>
<th>DELIVERED NEW CODE</th>
<th>PRE-EXISTING CODE</th>
<th>ITEM SIZE DATA</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td>UNIQUE SLOC</td>
<td>AUTO GEN SLOC</td>
<td>TOTAL UNMODIFIED SLOC</td>
</tr>
<tr>
<td>32,000</td>
<td>C++</td>
<td>5,000</td>
<td>0</td>
<td>25,000</td>
<td>2,000</td>
</tr>
<tr>
<td>32,000</td>
<td>C++</td>
<td>5,000</td>
<td>0</td>
<td>25,000</td>
<td>2,000</td>
</tr>
</tbody>
</table>
ESLOC Disadvantages

- Although well intentioned, ESLOC parameters (RD/RI/RT):
  - Need to be populated by an analyst intimately familiar with the SW
  - Are often misunderstood, misinterpreted, not populated, or populated with repeating values (same value for all SW components)
  - Can have large impact on ESLOC from small changes
  - Vary widely across programs, contributing to additional uncertainty and variability in SW productivities
  - Compound pre-existing code in cases of multiple SW snapshots
  - Cannot be independently verified – defending changes is difficult

- Example (perspective of CAAG analyst verifying SW datasheets):

<table>
<thead>
<tr>
<th>Logical SLOC</th>
<th>SOURCE</th>
<th>LANGUAGE</th>
<th>DELIVERED NEW CODE</th>
<th>PRE-EXISTING CODE</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td>UNIQUE SLOC</td>
<td>AUTOGEN SLOC</td>
</tr>
<tr>
<td>32,000</td>
<td>C++</td>
<td></td>
<td>5,000</td>
<td>0</td>
</tr>
<tr>
<td>32,000</td>
<td>C++</td>
<td></td>
<td>5,000</td>
<td>0</td>
</tr>
</tbody>
</table>
ESLOC Disadvantages Quantified

- We hypothesize ESLOC has many issues. What data backs up this claim? An all-encompassing NRO ground dataset was compiled and the following metrics were calculated:

<table>
<thead>
<tr>
<th>RD/RI/RT Source</th>
<th>% Total ESLOC</th>
</tr>
</thead>
<tbody>
<tr>
<td>KTR Populated with Repeating Values</td>
<td>60%</td>
</tr>
<tr>
<td>CAAG Populated with Formula</td>
<td>40%</td>
</tr>
</tbody>
</table>

- More than one-third of ESLOC was based on CAAG-populated RD/RI/RT
- Half of the ESLOC resulting from contractor-populated RD/RI/RT used repeating RD/RI/RT values (same values for multiple SW items)

\[
%\text{Rework} = (0.4 \times %RD) + (0.25 \times %RI) + (0.35 \times %RT)
\]

- %Rework shows very little correlation to %New or %Modified
- There is significant variation, verifying low quality of subjective RD/RI/RT
  - High %New but low %Rework
  - Low %New but high %Rework
  - Low %Modified but high %Rework
ESLOC Alternatives

✦ The evidence is clear: ESLOC needs to be replaced

*What are the objective alternatives?*

✦ Option 1: Set RD/RI/RT objectively
✦ Option 2: Assert an Objective SLOC (OSLOC) formula
✦ Option 3: Use regression techniques to derive CER-type method
Evaluation of Methods

- Standard model quality metrics were used to evaluate different options, including Standard Percent Error (SPE), correlation ($R^2$), average bias and error residual trending.

- Distribution and range of productivities was also considered as a way to compare methods.
  - ESLOC has a large range of productivities and is highly skewed, due to variability and uncertainty surrounding RD/RI/RT.
  - Less skew and tighter range of productivities indicates less uncertainty of inputs.
  - Evaluated standard deviation, skewness and 80th percentile divided by 20th percentile as characterizations of productivity distribution.
Option 1: Set RD/RI/RT Objectively

- RD/RI/RT vary significantly due to their high subjectivity. If these values could be assigned objectively, our sizing method would contain less uncertainty.

- We have observed contractors using formulas to populate RD/RI/RT and have begun internally populating %RI as %Modified when no better information is available.

- Option 1a: set RD/RI/RT as the following:

<table>
<thead>
<tr>
<th></th>
<th>RD</th>
<th>RI</th>
<th>RT</th>
</tr>
</thead>
<tbody>
<tr>
<td>ESLOC</td>
<td>5%</td>
<td>Modified/Pre-Existing</td>
<td>10%</td>
</tr>
<tr>
<td>Hours/ESLOC Distribution</td>
<td>Model Statistics</td>
<td></td>
<td></td>
</tr>
<tr>
<td>80th / 20th</td>
<td>6.09</td>
<td>Bias</td>
<td>-2%</td>
</tr>
<tr>
<td>Skew</td>
<td>0.43</td>
<td>SPE</td>
<td>76%</td>
</tr>
<tr>
<td>Stdev</td>
<td>0.36</td>
<td>R^2</td>
<td>0.28</td>
</tr>
</tbody>
</table>

Hours/OSLOC Distribution | Model Statistics |
| 80th / 20th | 6.68 | Bias | 0% |
| Skew | 0.84 | SPE | 84% |
| Stdev | 0.45 | R^2 | 0.26 |

*Results on subset of ground data that identify Modified SLOC

Using %Modified as %RI and using SEER standards for %RD and RT does not improve estimating method.
Option 1: Set RD/RI/RT Objectively

- Option 1a: set RD and RT to SEER SEM standards for reuse. This standard may not be appropriate for every SW CSCI.
- Option 1b: set all of RD/RI/RT to Modified/Pre-Existing, so

\[ OSLOC = New + 0.25 \times Autogen + (Unmod + Mod) \times \frac{Mod}{PreExisting} \]

**ESLOC**

<table>
<thead>
<tr>
<th>Hours/ESLOC Distribution</th>
<th>Model Statistics</th>
</tr>
</thead>
<tbody>
<tr>
<td>80th / 20th</td>
<td>6.09 Bias -2%</td>
</tr>
<tr>
<td></td>
<td>Skew 0.43 SPE 76%</td>
</tr>
<tr>
<td></td>
<td>Stdev 0.36 R² 0.28</td>
</tr>
</tbody>
</table>

**OSLOC Option 1b**

<table>
<thead>
<tr>
<th>Hours/OSLOC Distribution</th>
<th>Model Statistics</th>
</tr>
</thead>
<tbody>
<tr>
<td>80th / 20th</td>
<td>4.72 Bias 0%</td>
</tr>
<tr>
<td></td>
<td>Skew 0.33 SPE 67%</td>
</tr>
<tr>
<td></td>
<td>Stdev 0.36 R² 0.39</td>
</tr>
</tbody>
</table>

*Results on subset of ground data that identify Modified SLOC

Using %Modified as the entire rework percentage provides some improvement over ESLOC
Option 2: Assert an OSLOC Formula

- Option 1b was $OSLOC = New + 0.25 \times Autogen + (Unmod + Mod) \times \frac{Mod}{PreExisting}$
- If Autogen is small, and not expected to be a large influencer, and since $PreExisting = Unmod + Mod - Deleted$, if Deleted is small then effectively,

$$OSLOC = New + (Unmod + Mod) \times \frac{Mod}{Unmod + Mod}$$

Option 2a:

$$OSLOC = New + Mod$$

<table>
<thead>
<tr>
<th>Hours/ESLOC Distribution</th>
<th>Model Statistics</th>
</tr>
</thead>
<tbody>
<tr>
<td>80th / 20th</td>
<td>Bias -2%</td>
</tr>
<tr>
<td></td>
<td>Skew 0.43</td>
</tr>
<tr>
<td></td>
<td>Stdev 0.36</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Hours/OSLOC Distribution</th>
<th>Model Statistics</th>
</tr>
</thead>
<tbody>
<tr>
<td>80th / 20th</td>
<td>Bias 0%</td>
</tr>
<tr>
<td></td>
<td>Skew 0.36</td>
</tr>
<tr>
<td></td>
<td>Stdev 0.37</td>
</tr>
</tbody>
</table>

*Results on subset of ground data that identify Modified SLOC

**New + Modified is a simple sizing metric and performs better than ESLOC and similar to Option 1b**
Option 2: Assert an OSLOC Formula

+ Dataset includes programs of varying levels of confidence
  + Completed/on-going
  + UCC/contractor counter/estimate
  + Normalization/mappings being reassessed
  + Modified code identified/not identified

+ Option 2a was run on three datasets
  1. Ground programs that identify modified (previous chart)
  2. All ground programs
  3. Ground programs that identify modified using UCC and have no significant DQ issues

<table>
<thead>
<tr>
<th>Hours/OSLOC Distribution</th>
<th>Model Statistics</th>
</tr>
</thead>
<tbody>
<tr>
<td>80 / 20</td>
<td>4.23 Bias 0%</td>
</tr>
<tr>
<td>Skew</td>
<td>0.47 SPE 64%</td>
</tr>
<tr>
<td>Stdev</td>
<td>0.32 R^2 0.55</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Hours/OSLOC Distribution</th>
<th>Model Statistics</th>
</tr>
</thead>
<tbody>
<tr>
<td>80th / 20th</td>
<td>4.17 Bias 0%</td>
</tr>
<tr>
<td>Skew</td>
<td>0.32 SPE 62%</td>
</tr>
<tr>
<td>Stdev</td>
<td>0.33 R^2 0.26</td>
</tr>
</tbody>
</table>

New + Mod performs similarly on a larger set including low quality data and on a small set of high quality data
Option 2: Assert an OSLOC Formula

+ Recently we have begun collecting metrics on data SLOC (XML and HTML) and have been decrementing Data ESLOC in some cases.
+ The effect of data SLOC was tested on New + Modified (Option 2a) on the UCC data subset by removing all New and Modified data code (Option 2b).

**Option 2a**

<table>
<thead>
<tr>
<th>Hours/OSLOC Distribution</th>
<th>Model Statistics</th>
</tr>
</thead>
<tbody>
<tr>
<td>80th / 20th</td>
<td>4.17 Bias 0%</td>
</tr>
<tr>
<td>Skew</td>
<td>0.32 SPE 62%</td>
</tr>
<tr>
<td>Stdev</td>
<td>0.33 R^2 0.26</td>
</tr>
</tbody>
</table>

**Option 2b**

<table>
<thead>
<tr>
<th>Hours/OSLOC Distribution</th>
<th>Model Statistics</th>
</tr>
</thead>
<tbody>
<tr>
<td>80th / 20th</td>
<td>2.42 Bias 0%</td>
</tr>
<tr>
<td>Skew</td>
<td>0.43 SPE 51%</td>
</tr>
<tr>
<td>Stdev</td>
<td>0.29 R^2 0.33</td>
</tr>
</tbody>
</table>

+ Similar results show removing HTML and XML from code counts improves OSLOC model on set of all NRO ground SW programs.

Removing data from OSLOC improve Standard Error and reduces range of OSLOC productivities.
Option 3: Use Regression Techniques to Derive CER-type Method

- Parametric models were run to see if they could outperform a simple New + Modified OSLOC equation

- Due to the skewed distributions of New, Unmodified, Modified and Deleted SLOC, LOLS on multiplicative forms is the preferred regression method

<table>
<thead>
<tr>
<th>CER Tab Name</th>
<th>CER Function</th>
<th>SPE</th>
<th>$R^2$</th>
</tr>
</thead>
<tbody>
<tr>
<td>ZMPE ESLOC Base</td>
<td>SW Dev Hours = a*ESLOC</td>
<td>66.2%</td>
<td>0.45</td>
</tr>
<tr>
<td>LOLS ESLOC Base Exp</td>
<td>SW Dev Hours = a * ESLOC^b</td>
<td>69.0%</td>
<td>0.46</td>
</tr>
<tr>
<td>ZMPE 1</td>
<td>SW Dev Hours = a*New</td>
<td>147.1%</td>
<td>0.53</td>
</tr>
<tr>
<td>ZMPE 2</td>
<td>SW Dev Hours = a*(New+Modified)</td>
<td>63.5%</td>
<td>0.55</td>
</tr>
<tr>
<td>LOLS 3</td>
<td>SW Dev Hours = a * New^b</td>
<td>119.2%</td>
<td>0.56</td>
</tr>
<tr>
<td>LOLS 4</td>
<td>SW Dev Hours = a * (New+Mod)^b</td>
<td>63.6%</td>
<td>0.55</td>
</tr>
<tr>
<td>ZMPE 5</td>
<td>SW Dev Hours = a<em>New^b + c</em>Mod^d</td>
<td>63.3%</td>
<td>0.52</td>
</tr>
<tr>
<td>LOLS 6</td>
<td>SW Dev Hours = a*New^b * (Mod/New+1)^c</td>
<td>65.4%</td>
<td>0.55</td>
</tr>
<tr>
<td>LOLS 7</td>
<td>SW Dev Hours = a*New^b * (Mod/New+1)^c * (Unmod/New+1)^d</td>
<td>65.3%</td>
<td>0.74</td>
</tr>
</tbody>
</table>

*Results on set of all NRO ground data

- CER models produce similar regression statistics to OSLOC models

- LOLS 7 produced a model suggesting high unmodified SLOC was associated with less effort ($d < 0$), inconsistent with expectations
Investigating Unexpected CER Behavior

+ High amounts of unmodified reuse should take some additional effort to understand, integrate with new code, and retest

*What could cause a regression model to produce the opposite conclusion?*

+ **LOLS 6:** \( SW \text{ Dev Hours} = a \times \text{New}^b \times (1 + \frac{\text{Mod}}{\text{New}})^c \)

![Residual plot on LOLS 6](image)

+ Residual plot on LOLS 6 shows adding an unmodified scaling factor does not improve model based on expectations
  + SW programs with large amounts of unmodified SLOC are already being over-estimated

+ It was discovered that six of seven data points that consisted of multiple deliveries were over-estimated and are contained within the **red oval** – maybe these programs are being over-estimated because of how code counts were reported

<table>
<thead>
<tr>
<th>Multiple Differencing Example</th>
<th>Baseline A</th>
<th>Baseline B</th>
<th>New</th>
<th>Unmod</th>
<th>Mod</th>
<th>Deleted</th>
<th>Pre-Existing</th>
<th>DSLOC</th>
</tr>
</thead>
<tbody>
<tr>
<td>DLV 1.0</td>
<td>DLV 2.0</td>
<td>100</td>
<td>900</td>
<td>50</td>
<td>50</td>
<td>1,000</td>
<td>1,050</td>
<td></td>
</tr>
<tr>
<td>DLV 2.0</td>
<td>DLV 3.0</td>
<td>150</td>
<td>950</td>
<td>75</td>
<td>25</td>
<td>1,050</td>
<td>1,175</td>
<td></td>
</tr>
<tr>
<td><strong>Sum</strong></td>
<td><strong>250</strong></td>
<td><strong>1,850</strong></td>
<td><strong>125</strong></td>
<td><strong>75</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Single Diff</strong></td>
<td><strong>DLV 1.0</strong></td>
<td><strong>DLV 3.0</strong></td>
<td>225</td>
<td>850</td>
<td>100</td>
<td>50</td>
<td>1,000</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Multiple differencing snapshots tend to capture more churn and have higher SLOC counts than a single diff run
CER on Subset of Data

- Promising CER models were run on the set of ground SW programs that reported SW sizing based on one differencing run (7 DPs removed)

<table>
<thead>
<tr>
<th>CER Tab Name</th>
<th>CER Function</th>
<th>SPE</th>
<th>R2</th>
</tr>
</thead>
<tbody>
<tr>
<td>LOLS 6</td>
<td>SW Dev Hours = a*New^b * (Mod/New+1)^c</td>
<td>65.4%</td>
<td>0.55</td>
</tr>
<tr>
<td>LOLS 6 single diff subset</td>
<td>SW Dev Hours = a*New^b * (Mod/New+1)^c</td>
<td>57.7%</td>
<td>0.92</td>
</tr>
<tr>
<td>ZMPE 6 single diff subset</td>
<td>SW Dev Hours = a*New^b * (Mod/New+1)^c</td>
<td>51.7%</td>
<td>0.89</td>
</tr>
</tbody>
</table>

- Standard error and correlation improve significantly
- Unmodified now shows expected positive relationship, but provides very little additional explanatory power
- 7 data points composed of multiple SW deliveries have virtually nothing else in common – different contractors, ground function, size, etc. – there is no reason to believe there is another reason contributing to their previous overestimation
- Removing XML and HTML code improves models further

<table>
<thead>
<tr>
<th>CER Tab Name</th>
<th>CER Function</th>
<th>SPE</th>
<th>R2</th>
</tr>
</thead>
<tbody>
<tr>
<td>LOLS 6 single diff w/o data</td>
<td>SW Dev Hours = a*New^b * (Mod/New+1)^c</td>
<td>54.2%</td>
<td>0.91</td>
</tr>
<tr>
<td>ZMPE 6 single diff w/o data</td>
<td>SW Dev Hours = a*New^b * (Mod/New+1)^c</td>
<td>49.0%</td>
<td>0.88</td>
</tr>
</tbody>
</table>
Option 2: OSLOC Formula – on Subset

+ Removing data points that were composed of multiple SW deliveries improved the CER models

*Can reducing the set to those with one SW differencing summary improve the results of the OSLOC model?*

+ Recall the best performing OSLOC model was Option 2b:

\[
\text{OSLOC} = \text{New} + \text{Modified (excl. XML, HTML)}
\]

**Option 2b on all NRO ground**

<table>
<thead>
<tr>
<th>Hours/OSLOC Distribution</th>
<th>Model Statistics</th>
</tr>
</thead>
<tbody>
<tr>
<td>80 / 20</td>
<td>3.34 Bias 0%</td>
</tr>
<tr>
<td>Skew</td>
<td>0.54 SPE 61%</td>
</tr>
<tr>
<td>Stdev</td>
<td>0.32 R^2 0.60</td>
</tr>
</tbody>
</table>

**Option 2b on subset**

<table>
<thead>
<tr>
<th>Hours/OSLOC Distribution</th>
<th>Model Statistics</th>
</tr>
</thead>
<tbody>
<tr>
<td>80 / 20</td>
<td>3.31 Bias 0%</td>
</tr>
<tr>
<td>Skew</td>
<td>0.27 SPE 55%</td>
</tr>
<tr>
<td>Stdev</td>
<td>0.33 R^2 0.88</td>
</tr>
</tbody>
</table>

OSLOC Model improves when removing programs with multiple diffs, but does underestimate larger programs.
## Recommended Models

<table>
<thead>
<tr>
<th>#</th>
<th>Model Attribute</th>
<th>OSLOC 2b</th>
<th>CER 6</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Data collection going forward will be completely objective through the use of UCC-G</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>2</td>
<td>Simple to understand and implement</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>3</td>
<td>Reduces burden to contractor and improves CAAG ability to defend estimates</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>4</td>
<td>Performs significantly better when all data is based on a single SW differencing summary</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>5</td>
<td>Estimate by analogy (choose analogous program SW productivity)</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>Estimate by parametric model (no analogy needed)</td>
<td></td>
<td>X</td>
</tr>
</tbody>
</table>

**OSLOC 2b:**  \( OSLOC = New + Modified \) \( (\text{excl. XML, HTML}) \)

**CER 6:**  \( SW\ Dev\ Hrs = a \times New^b \times \left(1 + \frac{Mod}{New}\right)^c \)

Best OSLOC and parametric model perform similarly and share many of the same desirable characteristics.
Future of SW Estimating at CAAG

- CAAG to begin a parallel path approach to SW sizing and estimating
  - OSLOC metrics will be calculated and collected for all historic programs and future collections
  - Future estimates will investigate applying OSLOC method and parametric model as alternative methods of estimating and as cross checks
  - ESLOC metrics will be maintained and ESLOC inputs will continue to be collected to allow the analyst the option of reverting to estimate by ESLOC analogy should OSLOC and the parametric model not meet their needs

- Good practices that will be sought after to improve objective SW estimating
  - Recommend calculating SW differencing counts between the initial and current SW baselines
  - CAAG should ensure contractors always run UCC-G and run it correctly
  - Ensure documentation of software functionality exists to complement software sizing

- While OSLOC is still in “beta testing” we hope to see improvements in our ability to objectively estimate software development. Results and implementation will be reviewed and shared in the future
Questions?
Thank you!

Andrew Kicinski

kicinski@nro.mil
akicinski@integrity-apps.com

571-304-8867
The ESLOC Method

- The CAAG has historically used the ESLOC method to estimate SW development
- Equivalent Source Lines of Code (ESLOC) is a standardizing measure
  - 1 new line of code = 1 ESLOC
  - 1 autogenerated line of code = 0.25 ESLOC
  - 1 unmodified or modified line of code ≤ 1 ESLOC
    - Reuse is scaled based on an assessment of the percent redesign, reimplementation and retest (RD/RI/RT)

\[
ESLOC = \text{New} + 0.25 \times \text{Autogen} + (\text{Unmodified} + \text{Modified}) \times (0.4 \times \%\text{RD} + 0.25 \times \%\text{RI} + 0.35 \times \%\text{RT})
\]

- How the ESLOC method applies to our processes:

Data collection process:
1. Contractor runs UCC to collect objective sizing
2. Contractor assesses rework effort and provides RD/RI/RT
3. CAAG normalizes raw data including mapping hours/costs to SW Dev
4. SW Metrics are produced (Hours/ESLOC)

Point estimate process:
1. Contractor populates SW Datasheet including SW sizing and RD/RI/RT
2. SW sizing and RD/RI/RT are assessed for reasonability and adjusted as necessary, producing ESLOC
3. Analogous program SW productivities and labor rates are pulled as assumptions
4. \[
\text{ESLOC} \times \frac{\text{Hours}}{\text{ESLOC}} \times \frac{\$BY}{\text{Hour}} = \text{Point Est}
\]
Evaluation of Methods

Typically in model development, parametric models, such as CERs, can be evaluated by comparing actual costs to predicted costs by utilizing the proposed model and assessing SPE, $R^2$, bias, residual trending, etc.

This approach was taken for Option 3 (use regression techniques to derive CER-type method)

Assessing Options 1 (set RD/RI/RT objectively), 2 (assert OSLOC formula) and the current ESLOC method are more difficult

In practice these methods involve estimating by analogy

During methods development it is difficult to apply an analogous productivity to make the actual to predicted hours comparison

For our assessments, it was assumed that the average data set productivity would be the applied analogy to derive predicted hours

Distribution and range of productivities were also considered as ways to compare methods

ESLOC has a large range of productivities and is highly skewed, due to variability and uncertainty surrounding RD/RI/RT

Less skew and tighter range of productivities indicates less uncertainty of inputs

Evaluated standard deviation, skewness and 80th percentile divided by 20th percentile as characterizations of productivity distribution
## End Item Software Datasheet 1

**Preparer:**  
**Baseline A:**  
**Baseline B:**  
**Baseline A Date:**  
**Baseline B Date:**  
**Date:**  
**Site:**  

**Instructions:** Identify Baselines A and B that were run through the UCC differencing function to populate this Datasheet. Identify the dates of most recent update to the baselines.  
Use the tool on sheet 2.a2 RD RI RT Calculation Tool in to aid in the determination of these very important reuse factors.  
Use logical code for all SLOC counts.  
Links are provided for CA rankings at the top of each column  
Use a separate line for each CSCI. If more than one language is used within the CSCI, use a different line for each language.  
See Notes at the bottom of each page for explanation of columnar headings.

<table>
<thead>
<tr>
<th>Contractor</th>
<th>WBS NO.</th>
<th>Item ID</th>
<th>CSCI</th>
<th>Description</th>
<th>Logical SLOC</th>
<th>LANGUAGE</th>
<th>DELIVERED NEW CODE</th>
<th>SOURCE</th>
<th>UNIQUE SLOC</th>
<th>TOTAL UNMODIFIED SLOC</th>
<th>AUTO GEN SLOC</th>
<th>TOTAL MODIFIED SLOC</th>
<th>TOTAL DELETED SLOC</th>
<th>%RD</th>
<th>%RI</th>
<th>%RT</th>
<th>ESLOC</th>
<th>S/W</th>
<th>CONTR</th>
<th>SOURCE OF S/W</th>
<th>PERCENT SUBCONTRACT</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

See Notes at the bottom of each page for explanation of columnar headings.
RD/RI/RT Calculation Tool

Tool provided in CAAG datasheet package to assist in RD/RI/RT population

### Step 1: Set Redesign Factors

#### Redesign Breakdown

<table>
<thead>
<tr>
<th>Redesign Component</th>
<th>Least</th>
<th>Likely</th>
<th>Most</th>
<th>Percentage of the existing software that...</th>
</tr>
</thead>
<tbody>
<tr>
<td>Architectural Design Change</td>
<td>0%</td>
<td>0%</td>
<td>0%</td>
<td>requires architectural design change</td>
</tr>
<tr>
<td>Detailed Design Change</td>
<td>0%</td>
<td>0%</td>
<td>0%</td>
<td>requires detailed design change</td>
</tr>
<tr>
<td>Reverse Engineering Required</td>
<td>0%</td>
<td>0%</td>
<td>0%</td>
<td>requires reverse engineering</td>
</tr>
<tr>
<td>Redocumentation Required</td>
<td>0%</td>
<td>0%</td>
<td>0%</td>
<td>requires redocumentation</td>
</tr>
<tr>
<td>Revalidation Required</td>
<td>0%</td>
<td>0%</td>
<td>0%</td>
<td>requires revalidation with the new design</td>
</tr>
</tbody>
</table>

#### SEER-SEM Rework Percentage Calculation

<table>
<thead>
<tr>
<th>Weight</th>
<th>Redesign Component</th>
<th>Result Redesign Percentage</th>
<th>0.00%</th>
<th>0.00%</th>
<th>0.00%</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.22</td>
<td>Architectural Design Change</td>
<td>$A17$A$+A$18B$+$A$19C$-($A20$D$+$A21$E$) / (1-($A17$A$+$A18B$))</td>
<td>0%</td>
<td>0%</td>
<td>0%</td>
</tr>
</tbody>
</table>

### Step 2: Set Reimplementation Factors

#### Reimplementation Breakdown

<table>
<thead>
<tr>
<th>Reimplementation</th>
<th>Result Reimplementation Percentage</th>
<th>0.00%</th>
<th>0.00%</th>
<th>0.00%</th>
</tr>
</thead>
<tbody>
<tr>
<td>$A$37A$ - .37B$ - .045C$</td>
<td>$0.37A + 0.1B - 0.52C$</td>
<td>0%</td>
<td>0%</td>
<td>0%</td>
</tr>
</tbody>
</table>

### Step 3: Set Retest Factors

#### Retest Breakdown

<table>
<thead>
<tr>
<th>Test Plan Input</th>
<th>Result Retest Percentage</th>
<th>0.00%</th>
<th>0.00%</th>
<th>0.00%</th>
</tr>
</thead>
<tbody>
<tr>
<td>Test Plans Required</td>
<td>$0.1A + 0.2B + 0.3C + 0.4D + 0.5E + 0.6F$</td>
<td>0%</td>
<td>0%</td>
<td>0%</td>
</tr>
</tbody>
</table>
Definitions

- **Average Bias:**

\[
\text{%Bias} = 100 \times \frac{1}{n} \sum \frac{y_i - \hat{y}}{\hat{y}}
\]

- **Pearson R^2:** Pearson product-moment correlation correlation squared (between actual and estimated costs), which is the percentage of variation in actual costs that is explained by the CER.

\[
R^2 = \left[ \frac{n \sum y_i f(x_i) - \sum y_i \sum f(x_i)}{\sqrt{n \sum y_i^2 - (\sum y_i)^2} \sqrt{n \sum f(x_i)^2 - (\sum f(x_i))^2}} \right]^2
\]

- **SPE:** Standard Percent Error. For \(n\) data points and \(m\) estimated coefficients,

\[
SPE = 100 \times \sqrt{\frac{1}{n-m} \sum_{i=1}^{n} \left( \frac{y_i - \hat{y}}{\hat{y}} \right)^2}
\]
Software Cost Estimation Meets Software Diversity

Barry Boehm, USC
Software and IT- CAST Meeting
August 22, 2017
Outline

Sources of Software Diversity
- A Short History of Software Estimation Accuracy
- Process, Product, Property, and People Drivers

Options for Software Cost Estimation
- Expert Judgement/Consensus; Size-Based; Productivity-Based; Component-Based; Process-Based; Composites

Best Fits of Estimation-Types to Diversity-Types
- Extensions of ICSM Common Cases

Charting Your Path to Improved Estimates
A Short History of Software Estimation Accuracy

IDPD: Incremental Development Productivity Decline
MBSSE: Model-Based Systems and Sw Engr.
COTS: Commercial Off-the-Shelf
SoS: Systems of Systems

Relative Productivity
Estimation Error

Unprecedented
Preceded
Component-based
COTS
Agile
SoS. Apps, Widgets, IDPD, Clouds, Security, MBSSE

Time, Domain Understanding
Legend:
- Model has been calibrated with historical project data and expert (Delphi) data
- Model is derived from COCOMO II
- Model has been calibrated with expert (Delphi) data

Dates indicate the time that the first paper was published for the model.
Future Software Process Diversity

- Sequential Phases
  - Waterfall, V-Model
- Sequential Increments
  - Most agile methods: XP, Scrum, Crystal, SAFE
  - Pre-Planned Product Improvement (P3I)
- Continuous reprioritization
  - Kanban, DevOps
- Evolutionary Definition and Development
  - Incremental Commitment Spiral, Rational Unified Process
- Fully concurrent: Open Source
# ICSM Software Strategy Examples

<table>
<thead>
<tr>
<th><strong>Accounting Application</strong></th>
<th><strong>Simple Customer Business App</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Size/Complexity:</strong> Small/low</td>
<td><strong>Size/Complexity:</strong> Small/low</td>
</tr>
<tr>
<td><strong>Typical Change Rate/Month:</strong> Low</td>
<td><strong>Typical Change Rate/Month:</strong> Medium to high</td>
</tr>
<tr>
<td><strong>Criticality:</strong> High</td>
<td><strong>Criticality:</strong> Medium</td>
</tr>
<tr>
<td><strong>NDI Support:</strong> NDI-driven architecture</td>
<td><strong>NDI Support:</strong> No COTS, development and target environment well-defined</td>
</tr>
<tr>
<td><strong>Organizational Personnel Capability:</strong> NDI-experienced, medium to high</td>
<td><strong>Organizational Personnel Capability:</strong> Agile-ready, domain experience high</td>
</tr>
<tr>
<td><strong>Software Strategy:</strong> COTS</td>
<td><strong>Software Strategy:</strong> Architected agile</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>Cellphone Feature</strong></th>
<th><strong>Security Kernel</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Size/Complexity:</strong> Medium/medium</td>
<td><strong>Size/Complexity:</strong> Small/low</td>
</tr>
<tr>
<td><strong>Typical Change Rate/Month:</strong> Medium to high</td>
<td><strong>Typical Change Rate/Month:</strong> Low</td>
</tr>
<tr>
<td><strong>Criticality:</strong> Low</td>
<td><strong>Criticality:</strong> Extra high</td>
</tr>
<tr>
<td><strong>NDI Support:</strong> No COTS, development and target environment well-defined</td>
<td><strong>NDI Support:</strong> No COTS, development and target environment well-defined</td>
</tr>
<tr>
<td><strong>Organizational Personnel Capability:</strong> Agile-ready, domain experience high</td>
<td><strong>Organizational Personnel Capability:</strong> Strong formal methods experience</td>
</tr>
<tr>
<td><strong>Software Strategy:</strong> Agile</td>
<td><strong>Software Strategy:</strong> Formal methods</td>
</tr>
</tbody>
</table>
Incremental Development Productivity Decline (IDPD)

- **Example: Site Defense BMD Software**
  - 5 builds, 7 years, $100M; operational and support software
  - Build 1 productivity over 300 LOC/person month
  - Build 5 productivity under 150 LOC/PM
    - Including Build 1-4 breakage, integration, rework
    - 318% change in requirements across all builds
    - IDPD factor = 20% productivity decrease per build
- Similar trends in later unprecedented systems
- Not unique to DoD: key source of Windows Vista delays

- **Maintenance of full non-COTS SLOC, not ESLOC**
  - Build 1: 200 KSLOC new; 200K reused@20% = 240K ESLOC
  - Build 2: 400 KSLOC of Build 1 software to maintain, integrate
**Effects of IDPD on Number of Increments**

- Model relating productivity decline to number of builds needed to reach 8M SLOC Full Operational Capability
- Assumes Build 1 production of 2M SLOC @ 100 SLOC/PM
  - 20000 PM / 24 mo. = 833 developers
  - Constant staff size for all builds
- Analysis varies the productivity decline per build
  - Extremely important to determine the incremental development productivity decline (IDPD) factor per build
Future Software **Product Diversity**

- **Developed, Reused, Generated Software**
  - Source Lines of Code (SLOC), Function Points (FP)
  - Reused: Equivalent SLOC
  - Generated: Model Directives

- **Product Line Definition and Development**
  - Reused, Modified, Generated SLOC or FP

- **Non-Developmental Items (NDI), Cloud Services**
  - NDI: Commercial Off-the-Shelf (COTS), Open Source
  - Costing: Assessment, Tailoring, Glue Code, New-Release Adaptation

- **Domain Languages: Business, Supply Chain, Space**

- **Datasource-Driven: Selection Criteria**
Reuse at HP’s Queensferry Telecommunication Division

Time to Market (months)

Year

Non-reuse Project
Reuse project

0 10 20 30 40 50 60 70

86 87 88 89 90 91 92
Multi-Mission Support Systems Costing

• Product Line Engineering
  – Identify multi-mission commonalities and variabilities
  – Identify fully, partially sharable commonalities
  – Develop plug-compatible interfaces for variabilities

• Product Line Costing (COPLIMO) Parameters
  – Fractions of system fully reusable, partially reusable and cost of developing them for reuse
  – Fraction of system variabilities and cost of development
  – System lifetime and rates of change

• Product Line Life Cycle Challenges
  – Layered services vs. functional hierarchy
  – Modularization around sources of change
  – Version control, COTS refresh, and change prioritization
  – Balancing agility, assurance, and affordability
The Basic COPLIMO Model
- Constructive Product Line Investment Model

• Based on COCOMO II software cost model
  – Statistically calibrated to 161 projects, representing 18 diverse organizations

• Based on standard software reuse economic terms
  – RCR: Relative cost of reuse
  – RCWR: Relative cost of writing for reuse

• Avoids overestimation
  – Avoids RCWR for non-reused components

• Provides experience-based default parameter values

• Simple Excel spreadsheet model
  – Easy to modify, extend, interoperate
Basic COPLIMO Output Summary

Summary of Inputs:

- AVPROD: 300
- AVSIZE: 50000 (SLOC)
- UNIQ%: 40 (%)
- ADAP%: 30 (%)
- RUSE%: 30 (%)
- RCR-UNIQ: 100 (%)
- RCR-ADAP: 40 (%)
- RCR-RUSE: 5 (%)
- RCWR: 1.7

(Note: Do not change above values!)
(Change from "Input" sheet.)

7 year Product Line Effort Savings:

Table of Results:

<table>
<thead>
<tr>
<th># of Products</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
</tr>
</thead>
<tbody>
<tr>
<td>Unique SLOC</td>
<td>0</td>
<td>20000</td>
<td>40000</td>
<td>60000</td>
<td>80000</td>
<td>100000</td>
<td>120000</td>
<td>140000</td>
</tr>
<tr>
<td>Adapted SLOC</td>
<td>0</td>
<td>15000</td>
<td>30000</td>
<td>45000</td>
<td>60000</td>
<td>75000</td>
<td>90000</td>
<td>105000</td>
</tr>
<tr>
<td>Reused SLOC</td>
<td>0</td>
<td>15000</td>
<td>30000</td>
<td>45000</td>
<td>60000</td>
<td>75000</td>
<td>90000</td>
<td>105000</td>
</tr>
<tr>
<td>Total Non-PL SLOC</td>
<td>0</td>
<td>50000</td>
<td>100000</td>
<td>150000</td>
<td>200000</td>
<td>250000</td>
<td>300000</td>
<td>350000</td>
</tr>
<tr>
<td>Non-PL Effort (PM)</td>
<td>0</td>
<td>166.667</td>
<td>333.333</td>
<td>500</td>
<td>666.667</td>
<td>833.333</td>
<td>1000</td>
<td>1166.667</td>
</tr>
<tr>
<td>1-Product Equiv. SLOC</td>
<td>0</td>
<td>71000</td>
<td>26750</td>
<td>26750</td>
<td>26750</td>
<td>26750</td>
<td>26750</td>
<td>26750</td>
</tr>
<tr>
<td>1-Product Equiv. Effort</td>
<td>0</td>
<td>236.667</td>
<td>89.1667</td>
<td>89.1667</td>
<td>89.1667</td>
<td>89.1667</td>
<td>89.1667</td>
<td>89.1667</td>
</tr>
<tr>
<td>Cum. Equiv. PL SLOC</td>
<td>0</td>
<td>71000</td>
<td>97750</td>
<td>124500</td>
<td>151250</td>
<td>178000</td>
<td>204750</td>
<td>231500</td>
</tr>
<tr>
<td>Cum. PL Effort</td>
<td>0</td>
<td>236.667</td>
<td>325.833</td>
<td>415</td>
<td>504.167</td>
<td>593.333</td>
<td>682.5</td>
<td>771.667</td>
</tr>
<tr>
<td>PL Effort Savings</td>
<td>0</td>
<td>-70</td>
<td>7.5</td>
<td>85</td>
<td>162.5</td>
<td>240</td>
<td>317.5</td>
<td>395</td>
</tr>
<tr>
<td>PL Reuse Investment</td>
<td>0</td>
<td>70</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Return on Investment</td>
<td>N/A</td>
<td>-1</td>
<td>0.10714</td>
<td>1.21429</td>
<td>2.32143</td>
<td>3.42857</td>
<td>4.53571</td>
<td>5.642857</td>
</tr>
</tbody>
</table>

Product Line Development Cost Estimation

Net development effort savings vs. # of products in product line.
Persistence of Legacy Systems

• Before establishing new-system increments
  – Determine how to undo legacy system
Failed Greenfield Corporate Financial System

- Used waterfall approach
  - Gathered requirements
  - Chose best-fit ERP system
  - Provided remaining enhancements

- Needed to ensure continuity of service
  - Planned incremental phase-in of new services

- Failed due to inability to selectively phase out legacy services
  - Dropped after 2 failed tries at cost of $40M
Legacy Systems Patched, Highly Coupled
Financial and Non-Financial Services

Legacy Business Services

Contract Services
- Deliverables Management
- Billing
- Subcontracting

Project Services
- Budgeting
- Staffing
- Work Breakdown Structure
- Scheduling
- Change Tracking
- Progress Tracking
- Earned Value Management
- Reqs, Configuration Management
Result of Legacy Re-engineering

Legacy Business Services

Contract Services
- Contract Financial Services
  - Billing
  - Subcontract payments
  - ...
- Contract Non-Financial Services
  - Deliverables mgmt.
  - Terms compliance
  - ...

General Financial Services
- Accounting
- Budgeting
- Earned value
- Payroll
- ...

General Non-Financial Services
- Progress tracking
- Change tracking
- ...

Project Services
- Project Financial Services
  - WBS
  - Expenditure categories
  - ...
- Project Non-Financial Services
  - Scheduling
  - Staffing
  - Reqs CM
  - ...

Project Non-Financial Services
- Scheduling
- Staffing
- Reqs CM
- ...

Financial Services
- Billing
- Subcontract payments
- ...

Non-Financial Services
- Deliverables mgmt.
- Terms compliance
- ...

General Financial Services
- Accounting
- Budgeting
- Earned value
- Payroll
- ...

General Non-Financial Services
- Progress tracking
- Change tracking
- ...

Project Financial Services
- WBS
- Expenditure categories
- ...

Project Non-Financial Services
- Scheduling
- Staffing
- Reqs CM
- ...

Copyright © USC-CSSE
Future Software Properties Diversity

• Dependability
  – Reliability, Availability, Safety, Security

• Changeability
  – Adaptability, Maintainability, Modifiability, Repairability

• Mission Effectiveness
  – Response Time, Throughput, Accuracy, Usability, Scalability, Interoperability

• Life Cycle Efficiency (Cost-Effectiveness)
  – Development and Maintenance Cost, Schedule; Reusability
Response Time Rqt. Impact on Cost

- Original Spec
- After Prototyping

Original Cost

$100M
$50M

Original Architecture: Modified Client-Server
Required Architecture: Custom; many cache processors

Response Time (sec)
1 2 3 4 5
Future Software People Diversity

• Desired Software People Capabilities
  – Software System Analysis
  – Software System Development
  – Application Domain Experience
  – Software Languages and Tools Experience
  – Software Process Maturity
  – Team Cohesion
  – Low Personnel Turnover
  – Familiarity with Apps, Widgets, Social Media, Data Analytics, Multimedia, Virtual Reality
Outline

• Sources of Software Diversity
  – A Short History of Software Estimation Accuracy
  – Process, Product, Property, and People Drivers

Options for Software Cost Estimation
  – Expert Judgement/Consensus; Size-Based; Productivity-Based; Component-Based; Process-Based; Composites

• Best Fits of Estimation-Types to Diversity-Types
  – Extensions of ICSM Common Cases

• Charting Your Path to Improved Estimates
Estimation-Type Options

- **Expert-Judgement; Stakeholder Consensus**
  - Planning Poker, Wideband Delphi, Bottom-Up

- **Analogy: Previous Projects; Yesterday’s Weather**
  - Agile COCOMO II, Case-Based Reasoning, Causal Modeling

- **Parametric Models**
  - COCOMO/COSTAR, Knowledge Plan, SEER, SLIM, True-S

- **Resource-Limited**
  - Cost or Schedule as Independent Variable (CAIV, SAIV)

- **Reuse-Driven: Equivalent Size**
  - Adjusted for %Design,Code,Test Modified, Understandability

- **Product Line**
  - % Development for Reuse; % Development with Reuse
Outline

• Sources of Software Diversity
  – A Short History of Software Estimation Accuracy
  – Process, Product, Property, and People Drivers

• Options for Software Cost Estimation
  – Expert Judgement/Consensus; Size-Based; Productivity-Based; Component-Based; Process-Based; Composites

→ Best Fits of Estimation-Types to Diversity-Types
  – Extensions of ICSM Common Cases

• Charting Your Path to Improved Estimates
Best Fits of Estimation-Types to Diversity-Types

- **Pure Agile:** Planning Poker, Agile COCOMO II
- **Architected Agile**
  - COSYSMO for architecting; Planning Poker, CAIV-SAIV for sprints, releases; IDPD for large systems
- **Formal Methods:** $/SLOC by Evaluated Assurance Level
- **NDI/Services-Intensive:** Oracle, SAP, other ERP
  - RICE Objects: (R)eports, (I)nterfaces, (C)onversions, (E)nhancements
  - COCOTS, Value-Added Function Points, Agile for portions
- **Hybrid Agile/Plan-Driven**
  - Expert Delphi, Parametric Models, Agile for portions; IDPD
- **Systems of Systems**
  - COSYSMO for Integrator; Hybrid Agile/Plan-Driven for component systems
- **Family of Systems:** COPLIMO
- **Brownfield:** Experiment for refactoring; above for rebuilding
Proliferation of Estimation Types
Thanks to Capers Jones

• Source Lines of Code (SLOC)
  – Physical/Logical; Executable/nonexecutable; New/reused; 
    Programmed/generated/translated; Added/modified/deleted

• Function points (FP)
  – Original IBM; IFPUG 2,3,4; Fast; COSMIC; Mark II, FISMA, NESMA; Unadjusted/adjusted; RICE Objects

• SLOC/FP backfire ratios
  – SPR, QSM, DAVIDS, Gartner Group

• Agile sizing
  – Story points (Planning Poker, T-shirt size); ideal person-weeks

• Risky: high variability
  – Number of requirements/shalls; nonfunctional requirements (SNAP points); UML diagram counts
Outline

• **Sources of Software Diversity**
  – A Short History of Software Estimation Accuracy
  – Process, Product, Property, and People Drivers

• **Options for Software Cost Estimation**
  – Expert Judgement/Consensus; Size-Based; Productivity-Based; Component-Based; Process-Based; Composites

• **Best Fits of Estimation-Types to Diversity-Types**
  – Extensions of ICSM Common Cases

→ Charting Your Path to Improved Estimates
Charting Your Path to Improved Estimates

• Identify your most critical future improvement areas

• Identify, experiment with best candidate estimation methods in most critical areas

• Experiment with available methods for others; evaluate further improvement needs

• Build up, analyze experience base, use to steer path
COCOMO II Experience Factory: IV

System objectives: fcny, perf., quality
Corporation parameters: tools, processes, reuse

COCOMO 2.0

Evaluate Corporate SW Improvement Strategies
Recalibrate COCOMO 2.0
Accumulate COCOMO 2.0 calibration data

Rescope

Cost, Sched, Risks

No

Ok?
Yes

Execute project to next Milestone

Revise Milestones, Plans, Resources

Milestone plans, resources

Milestone expectations

Ok?
Yes
No

M/S Results

Revised Expectations

Done?
Yes
No

End

Cost, Sched, Quality drivers

Improved Corporate Parameters

Copyright © USC-CSSE
Backup Charts
USC-CSSE Modeling Methodology
- concurrency and feedback implied

1. Determine Model Needs
2. Analyze existing literature
3. Perform Behavioral analyses
4. Define relative significance, data, ratings
5. Perform expert-judgment Delphi assessment, formulate a priori model
6. Gather project data
7. Determine Bayesian A-Posteriori model
8. Gather more data; refine model
Step 6: Gather, Analyze Project Data

• Best to pilot data collection with early adopters
  – Identifies data definition ambiguities
  – Identifies data availability problems
  – Identifies need for data conditioning

• Best to collect initial data via interviews
  – Avoids misinterpretations
    • Endpoint milestones; activities included/excluded; size definitions
  – Uncovers hidden assumptions
    • Schedule vs. cost minimization; overtime effort reported
Initial Data Analysis May Require Model Revision

- Initial COCOTS model adapted from COCOMO II, with different parameters
  - Effort = A* (Size)^B* \prod (Effort Multipliers)
- Amount of COTS integration glue code used for Size
- Data analysis showed some projects with no glue code, much effort
  - Effort devoted to COTS assessment, tailoring
COCOTS Effort Distribution: 20 Projects

Mean % of Total COTS Effort by Activity (+/- 1 SD)

- Assessment: 49.07% ± 7.57%
- Tailoring: 50.99% ± 7.48%
- Glue Code: 61.25%
- System Volatility: 20.27% ± 2.35%
Revised COCOTS Model

- COCOMO-like model for glue code effort
- Unit cost approach for COTS assessment effort
  - Number of COTS products to assess
  - Number of attributes to assess, weighted by complexity
- Activity-based approach for COTS tailoring effort
  - COTS parameters setting, script writing, reports layout, GUI tailoring, protocol definitions
New Glue Code Submodel Results

• New calibration results
  – Excluding projects with very large, very small amounts of glue code
    • [0.5 - 100 KLOC]: Pred (.30) = 9/17 = 53%
    • [2 - 100 KLOC]: Pred (.30) = 8/13 = 62%
  – Previous calibration results:
    • [0.1 - 390 KLOC]: Pred (.30) = 4/13 = 31%
• Pred(.30) = percent of projects with estimates within 30% of actuals
COCOMO II Experience Factory: I

System objectives: fcn’y, perf., quality

Corporate parameters: tools, processes, reuse

COCOMO 2.0

Rescope

Cost, Sched, Risks

Ok?

Yes

No
COCOMO II Experience Factory: II

System objectives: fcn’y, perf., quality

Corporate parameters: tools, processes, reuse

COCOMO 2.0

Rescope

Ok?

Cost, Sched, Risks

Yes

No

Execute project to next Milestone

Revise Milestones, Plans, Resources

Milestones plans, resources

M/S Results

No

Yes

Milestone plans, resources

Yes

No

Milestone expectations

Ok?

Done?

Yes

End

No

Revised Expectations
COCOMO II Experience Factory: III

System objectives: fcn’y, perf., quality

Corporate parameters: tools, processes, reuse

COCOMO 2.0

Rescope

Cost, Sched, Risks

Ok?

Yes

No

Execute project to next Milestone

Revise Milestones, Plans, Resources

Milestone plans, resources

Milestone expectations

Accumulate COCOMO 2.0 calibration data

Recalibrate COCOMO 2.0

Done?

Yes

No

End

Yes

No

Yes

No

Yes

No

10/19/2016 - Copyright © USC-CSSE
Better, Cheaper, Faster: Pick Any Two

COCOMO II Model Results

- For 100-KSLOC set of features
- Can “pick all three” with 77-KSLOC set of features
Deriving Software Sustainment Cost Estimating Relationships in a Diverse Army Execution Environment

IT CAST
23 August 2017
SWM Initiative Objective and Strategy

Accurately estimate Army system software maintenance costs to:
- Effectively project and justify software and system life cycle costs
- Objectively evaluate Army system software maintenance execution costs
- Inform and optimize the allocation of available maintenance resources across the Army

Collect and evaluate SWM cost and technical data for all Army operational systems (Phase I and Phase II data call)
Generate and validate cost estimating relationships from Phase I and Phase II data collection
Implement systemic Army SWM data collection via the SRDR-M, Populate cost and technical data repository
Improve Army SWM policy, business, and technical requirements

**Effective software maintenance cost estimation is the basis for Army system software life cycle cost management**
Phase I
Data Collection and Evaluation
For this effort, software maintenance is defined as:

- Software maintenance includes all software change activities and products associated with modifying a software system after EMD has completed and a software release has been provided to an external party.
- The release is the primary SWM change product - a composite of one or more changes - it can be either a formal release or an engineering release.
- SWM includes software enhancements and software corrections/adaptations.
- SWM includes activities and change products funded by multiple funding sources.
- Fixed and Variable costs accrued at both the system and organizational levels by both organic and contractor resources.
- Software maintenance and software sustainment are considered to be synonymous.
Army Software Maintenance WBS

Software Maintenance

System Specific

1.0 Software Change Product
- Change Requirements
- Change Development
- B/L Integration & Test
- IV&V

System Specific

2.0 Project Management
- Planning
- Execution Management
- Configuration Management
- Resource & Team Management
- Contracting Management
- Measurement - Reporting

System Specific

3.0 Software Licenses
- License Management
- License - Right to Use
- License - Maintenance
  - COTS
  - NDI
  - Other

System Specific

4.0 Certification & Accreditation
- Security
- Safety
- Networthiness
- Airworthiness

System Specific

5.0 System Facilities
- Hardware
  - Software Development
  - Assets/Workstations
  - System Integration & Test Facilities
  - Test Equipment - Tools
  - Facility Operations

System Specific

6.0 Sustaining Engineering
- Engineering Support
  - Test Support
  - Software Delivery
  - Technical Studies
  - User Support
    - Help Desk
    - Training

Non-System Specific

8.0 Operational Management
- Operations
- Organization Management
- Personnel Management
- Financial Management
- Information Management
- Process Management
- Change Management

UNCLASSIFIED
Distribution Statement A: Approved for Public Release: Distribution is Unlimited
**Phase I Data Requirements**

<table>
<thead>
<tr>
<th>System Context</th>
<th>Program Level</th>
<th>Release Level</th>
</tr>
</thead>
<tbody>
<tr>
<td>• System description</td>
<td>• Annual effort/cost data (total annual plus WBS elements #2 through #8)</td>
<td>• Release context information</td>
</tr>
<tr>
<td>• Organizations involved</td>
<td>• broken out by government and contractor (3 years of data)</td>
<td>• Application domain</td>
</tr>
<tr>
<td>• Maintenance process</td>
<td>• Labor rates</td>
<td>• Operating environment</td>
</tr>
<tr>
<td>• CMMI rating</td>
<td>• Hourly basis for FTEs</td>
<td>• Release effort / cost</td>
</tr>
<tr>
<td>• Number of software baselines</td>
<td>• Software licenses</td>
<td>• Schedule - start and end dates</td>
</tr>
<tr>
<td>• Number of hardware platforms/number of users</td>
<td></td>
<td>• Size data (those that apply)</td>
</tr>
<tr>
<td>• Analogous systems</td>
<td></td>
<td>• Software requirements</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• External requirements</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Source Lines of Code (SLOC)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Non-SLOC based size (e.g. RICE-FW, use cases, story points)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Software changes counts by priority (e.g. change requests, problem reports, defects)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• IAVAs</td>
</tr>
</tbody>
</table>
Phase I Data Collection Process

PEOs/SECs/SEDs identified 5 programs per organization for Phase I data collection (56 programs)

DASA-CE met with program/system representative to explain data collection questionnaire and clarify requirements

System representative completed and submitted initial draft of questionnaire

DASA-CE team reviewed questionnaire, identified questions, and met with representative to discuss context and issues

System representative updated questionnaire based on DASA-CE findings

DASA-CE reviewed submission and continued to rework with system representative as necessary

Final data submission was accepted and evaluated for availability, integrity, and usability

UNCLASSIFIED
Distribution Statement A: Approved for Public Release: Distribution is Unlimited
Software Maintenance Data Evaluation

Availability

- Completeness of required data set
- Underlying SWM business and technical processes are well enough defined to produce objective data on a periodic and/or event driven basis
- IT systems and tools exist to enable systematic and timely data collection

Integrity

- Data are derivatives of actual SWM technical and management processes
- All data (measures) are explicitly defined - measurement contexts are known
- Cost data is directly correlated with the WBS defined output products and activities
- Data is consistent - methods exist to address system conflicts (normalization)

Usability

- Data is aligned with stakeholder decision information needs
- Data can be objectively characterized and interpreted
- Mapping and aggregation structures and methods exist to combine data
- Potential emerging information requirements have been considered
# Detailed Data Evaluation

## Detailed System Assessment

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>(PEO 4)</td>
<td>SEC 3</td>
<td>System 1</td>
<td>G</td>
<td>G</td>
<td>G</td>
<td>G</td>
<td>N/A</td>
<td>G</td>
<td>G</td>
<td>G</td>
<td>N/A</td>
<td>R</td>
<td>N/A</td>
</tr>
<tr>
<td>1</td>
<td>(PEO 4)</td>
<td>SEC 3</td>
<td>System 2</td>
<td>G</td>
<td>G</td>
<td>Y</td>
<td>Y</td>
<td>R</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>N/A</td>
<td>R</td>
<td>G</td>
</tr>
<tr>
<td>1</td>
<td>(PEO 4)</td>
<td>SEC 3</td>
<td>System 3</td>
<td>R</td>
<td>G</td>
<td>Y</td>
<td>R</td>
<td>G</td>
<td>G</td>
<td>Y</td>
<td>Y</td>
<td>N/A</td>
<td>Y</td>
<td>R</td>
</tr>
<tr>
<td>1</td>
<td>PEO 1</td>
<td>(SEC 2)</td>
<td>System 4</td>
<td>G</td>
<td>R</td>
<td>O</td>
<td>R</td>
<td>R</td>
<td>R</td>
<td>R</td>
<td>R</td>
<td>N/A</td>
<td>R</td>
<td>R</td>
</tr>
<tr>
<td>1</td>
<td>PEO 1</td>
<td>(SEC 2)</td>
<td>System 5</td>
<td>G</td>
<td>G</td>
<td>Y</td>
<td>R</td>
<td>R</td>
<td>G</td>
<td>Y</td>
<td>G</td>
<td>G</td>
<td>G</td>
<td>R</td>
</tr>
<tr>
<td>1</td>
<td>PEO 1</td>
<td>(SEC 2)</td>
<td>System 6</td>
<td>G</td>
<td>G</td>
<td>O</td>
<td>R</td>
<td>R</td>
<td>Y</td>
<td>R</td>
<td>R</td>
<td>Y</td>
<td>R</td>
<td>G</td>
</tr>
<tr>
<td>1</td>
<td>PEO 1</td>
<td>(SEC 2)</td>
<td>System 7</td>
<td>G</td>
<td>G</td>
<td>G</td>
<td>G</td>
<td>G</td>
<td>G</td>
<td>G</td>
<td>G</td>
<td>G</td>
<td>G</td>
<td>G</td>
</tr>
<tr>
<td>1</td>
<td>PEO 1</td>
<td>(SEC 2)</td>
<td>System 8</td>
<td>G</td>
<td>R</td>
<td>R</td>
<td>R</td>
<td>R</td>
<td>R</td>
<td>R</td>
<td>R</td>
<td>R</td>
<td>R</td>
<td>R</td>
</tr>
</tbody>
</table>

## Initial System Overall

## Detailed Release Assessment

<table>
<thead>
<tr>
<th>PEO</th>
<th>SEC</th>
<th>System</th>
<th>Release</th>
<th>CER Usability</th>
<th>SER Usability</th>
<th>Size: Requirements</th>
<th>Size: External Interfaces</th>
<th>Size: SLOC</th>
<th>Size: Non-SLOC</th>
<th>Size: SW Changes</th>
<th>IAVAs</th>
<th>Effort (WBS-1)</th>
<th>Schedule (WBS-1&amp;2)</th>
</tr>
</thead>
<tbody>
<tr>
<td>(PEO 4)</td>
<td>SEC 3</td>
<td>System 1</td>
<td>Release 1</td>
<td>Y</td>
<td>Y</td>
<td>G</td>
<td>G</td>
<td>G</td>
<td>N/A</td>
<td>G</td>
<td>G</td>
<td>G</td>
<td>Y</td>
</tr>
<tr>
<td>(PEO 4)</td>
<td>SEC 3</td>
<td>System 1</td>
<td>Release 2</td>
<td>Y</td>
<td>Y</td>
<td>G</td>
<td>G</td>
<td>G</td>
<td>N/A</td>
<td>G</td>
<td>G</td>
<td>Y</td>
<td>G</td>
</tr>
<tr>
<td>(PEO 4)</td>
<td>SEC 3</td>
<td>System 2</td>
<td>Release 1</td>
<td>G</td>
<td>G</td>
<td>R</td>
<td>R</td>
<td>R</td>
<td>Y</td>
<td>N/A</td>
<td>G</td>
<td>G</td>
<td>G</td>
</tr>
<tr>
<td>(PEO 4)</td>
<td>SEC 3</td>
<td>System 3</td>
<td>Release 1</td>
<td>G</td>
<td>G</td>
<td>G</td>
<td>N/A</td>
<td>G</td>
<td>N/A</td>
<td>G</td>
<td>N/A</td>
<td>G</td>
<td></td>
</tr>
<tr>
<td>(PEO 4)</td>
<td>SEC 3</td>
<td>System 3</td>
<td>Release 2</td>
<td>R</td>
<td>G</td>
<td>R</td>
<td>G</td>
<td>N/A</td>
<td>G</td>
<td>N/A</td>
<td>G</td>
<td></td>
<td></td>
</tr>
<tr>
<td>(PEO 4)</td>
<td>SEC 3</td>
<td>System 4</td>
<td>Release 1</td>
<td>G</td>
<td>R</td>
<td>R</td>
<td>N/A</td>
<td>G</td>
<td>N/A</td>
<td>G</td>
<td>N/A</td>
<td>G</td>
<td></td>
</tr>
<tr>
<td>(PEO 4)</td>
<td>SEC 3</td>
<td>System 4</td>
<td>Release 2</td>
<td>G</td>
<td>G</td>
<td>G</td>
<td>N/A</td>
<td>G</td>
<td>N/A</td>
<td>G</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(PEO 4)</td>
<td>SEC 4</td>
<td>System 5</td>
<td>Release 1</td>
<td>Y</td>
<td>R</td>
<td>R</td>
<td>G</td>
<td>G</td>
<td>N/A</td>
<td>G</td>
<td>N/A</td>
<td>Y</td>
<td></td>
</tr>
<tr>
<td>(PEO 4)</td>
<td>SEC 4</td>
<td>System 5</td>
<td>Release 2</td>
<td>Y</td>
<td>R</td>
<td>R</td>
<td>R</td>
<td>R</td>
<td>R</td>
<td>R</td>
<td>R</td>
<td>R</td>
<td></td>
</tr>
<tr>
<td>PEO 1</td>
<td>(SEC 2)</td>
<td>System 5</td>
<td>Release 1</td>
<td>Y</td>
<td>R</td>
<td>R</td>
<td>G</td>
<td>G</td>
<td>N/A</td>
<td>G</td>
<td>N/A</td>
<td>R</td>
<td></td>
</tr>
<tr>
<td>PEO 1</td>
<td>(SEC 2)</td>
<td>System 5</td>
<td>Release 2</td>
<td>G</td>
<td>G</td>
<td>R</td>
<td>R</td>
<td>R</td>
<td>R</td>
<td>R</td>
<td>R</td>
<td>R</td>
<td></td>
</tr>
<tr>
<td>PEO 1</td>
<td>(SEC 2)</td>
<td>System 5</td>
<td>Release 3</td>
<td>G</td>
<td>G</td>
<td>R</td>
<td>R</td>
<td>R</td>
<td>N/A</td>
<td>R</td>
<td>G</td>
<td></td>
<td></td>
</tr>
<tr>
<td>PEO 1</td>
<td>(SEC 2)</td>
<td>System 5</td>
<td>Release 4</td>
<td>G</td>
<td>G</td>
<td>R</td>
<td>G</td>
<td>N/A</td>
<td>R</td>
<td>G</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PEO 1</td>
<td>(SEC 2)</td>
<td>System 5</td>
<td>Release 5</td>
<td>G</td>
<td>G</td>
<td>R</td>
<td>G</td>
<td>N/A</td>
<td>R</td>
<td>G</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PEO 1</td>
<td>(SEC 2)</td>
<td>System 5</td>
<td>Release 6</td>
<td>G</td>
<td>G</td>
<td>R</td>
<td>G</td>
<td>N/A</td>
<td>R</td>
<td>G</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PEO 1</td>
<td>(SEC 2)</td>
<td>System 6</td>
<td>Release 1</td>
<td>R</td>
<td>R</td>
<td>R</td>
<td>G</td>
<td>G</td>
<td>Y</td>
<td>N/A</td>
<td>G</td>
<td>R</td>
<td></td>
</tr>
<tr>
<td>PEO 1</td>
<td>(SEC 2)</td>
<td>System 6</td>
<td>Release 2</td>
<td>R</td>
<td>R</td>
<td>G</td>
<td>G</td>
<td>Y</td>
<td>N/A</td>
<td>G</td>
<td>R</td>
<td></td>
<td></td>
</tr>
<tr>
<td>PEO 1</td>
<td>(SEC 2)</td>
<td>System 6</td>
<td>Release 3</td>
<td>R</td>
<td>R</td>
<td>G</td>
<td>G</td>
<td>Y</td>
<td>N/A</td>
<td>Y</td>
<td>R</td>
<td></td>
<td></td>
</tr>
<tr>
<td>PEO 1</td>
<td>(SEC 2)</td>
<td>System 7</td>
<td>Release 1</td>
<td>R</td>
<td>R</td>
<td>G</td>
<td>Y</td>
<td>G</td>
<td>N/A</td>
<td>G</td>
<td>O</td>
<td>R</td>
<td></td>
</tr>
<tr>
<td>PEO 1</td>
<td>(SEC 2)</td>
<td>System 7</td>
<td>Release 2</td>
<td>R</td>
<td>R</td>
<td>G</td>
<td>Y</td>
<td>G</td>
<td>N/A</td>
<td>G</td>
<td>R</td>
<td>O</td>
<td></td>
</tr>
<tr>
<td>PEO 1</td>
<td>(SEC 2)</td>
<td>System 8</td>
<td>Release 1</td>
<td>G</td>
<td>G</td>
<td>G</td>
<td>G</td>
<td>G</td>
<td>N/A</td>
<td>Y</td>
<td>N/A</td>
<td>R</td>
<td></td>
</tr>
<tr>
<td>PEO 1</td>
<td>(SEC 2)</td>
<td>System 9</td>
<td>Release 1</td>
<td>R</td>
<td>R</td>
<td>Y</td>
<td>G</td>
<td>G</td>
<td>N/A</td>
<td>Y</td>
<td>N/A</td>
<td>R</td>
<td></td>
</tr>
<tr>
<td>PEO 1</td>
<td>(SEC 2)</td>
<td>System 9</td>
<td>Release 2</td>
<td>R</td>
<td>R</td>
<td>Y</td>
<td>G</td>
<td>R</td>
<td>N/A</td>
<td>N/A</td>
<td>N/A</td>
<td>R</td>
<td></td>
</tr>
<tr>
<td>PEO 1</td>
<td>(SEC 2)</td>
<td>System 9</td>
<td>Release 3</td>
<td>R</td>
<td>R</td>
<td>Y</td>
<td>G</td>
<td>N/A</td>
<td>R</td>
<td>N/A</td>
<td>R</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**UNCLASSIFIED**

Distribution Statement A: Approved for Public Release: Distribution is Unlimited
Data Evaluation Phase I - Summary

• Formal data evaluation process was used to rate the data
  - Data was collected from 56 programs*
    • 43 programs provided total system SWM costs (G, Y)
  - Rating criteria is shown below:

<table>
<thead>
<tr>
<th>Initial System Overall</th>
<th>Detailed System Assessment</th>
</tr>
</thead>
<tbody>
<tr>
<td>R</td>
<td>14</td>
</tr>
<tr>
<td>O</td>
<td>0</td>
</tr>
<tr>
<td>Y</td>
<td>1</td>
</tr>
<tr>
<td>G</td>
<td>40</td>
</tr>
<tr>
<td>N/A</td>
<td>1</td>
</tr>
<tr>
<td>Total</td>
<td>56</td>
</tr>
</tbody>
</table>

Table 1. Data Quality Levels

<table>
<thead>
<tr>
<th>Color</th>
<th>Definition</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>R</td>
<td>Red indicates there is no planning or actual data reported</td>
<td>0</td>
</tr>
<tr>
<td>O</td>
<td>Orange indicates only planning data was reported</td>
<td>1</td>
</tr>
<tr>
<td>Y</td>
<td>Yellow indicates FTE or partial, actual data was reported</td>
<td>2</td>
</tr>
<tr>
<td>G</td>
<td>Green indicates that actual data was reported</td>
<td>3</td>
</tr>
</tbody>
</table>

*Detailed breakout of data evaluation by data point provided in backup
**Data Evaluation - WBS 1.0 Release Data**

### Initial Release Overall

<table>
<thead>
<tr>
<th></th>
<th>CER Usability</th>
<th>SER Usability</th>
<th>Size: Requirements</th>
<th>Size: External Interfaces</th>
<th>Size: SLOC</th>
<th>Size: Non-SLOC</th>
<th>Size: SW Changes</th>
<th>IAVAs</th>
<th>Effort (WBS-1)</th>
<th>Schedule (WBS-1)</th>
</tr>
</thead>
<tbody>
<tr>
<td>R</td>
<td>71</td>
<td>77</td>
<td>101</td>
<td>79</td>
<td>46</td>
<td>28</td>
<td>39</td>
<td>67</td>
<td>60</td>
<td>40</td>
</tr>
<tr>
<td>O</td>
<td>44</td>
<td>43</td>
<td>5</td>
<td>2</td>
<td>5</td>
<td>0</td>
<td>3</td>
<td>47</td>
<td>47</td>
<td>12</td>
</tr>
<tr>
<td>Y</td>
<td>23</td>
<td>22</td>
<td>3</td>
<td>2</td>
<td>4</td>
<td>2</td>
<td>5</td>
<td>0</td>
<td>23</td>
<td>27</td>
</tr>
<tr>
<td>N/A</td>
<td>1</td>
<td>1</td>
<td>39</td>
<td>80</td>
<td>94</td>
<td>176</td>
<td>55</td>
<td>42</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Total</td>
<td>218</td>
<td>218</td>
<td>218</td>
<td>218</td>
<td>218</td>
<td>218</td>
<td>218</td>
<td>218</td>
<td>218</td>
<td>218</td>
</tr>
</tbody>
</table>

### Detailed Release Assessment

- Data was collected from 218 releases
  - 146 releases had sufficient data to use in CER cost calculations (G,Y,O)
  - Size data was not always consistently tracked and generally was not mapped to resource (effort/cost/schedule) information
    - 124 releases tracked some sort of software change counts (defects, PTRs)
    - 109 releases tracked IAVAs
    - Systems in different super-domains used different size measures
      - Many weapon systems tracked SLOC data

*Detailed breakout of data evaluation by data point provided in backup*
Phase I SWM Data Analysis
Analysis Background

- The analysis covers Phase I data only
  - Phase II data will result in updated CERs and data demographics
- Estimating approach is specific to the SWM WBS. For any relationship identified, the WBS coverage should be noted
- Given the data sample size, the super domain classification is used to group similar data points
- All data points and associated classification are listed in the backup
- Utilized data represents both post deployment software support (PDSS) as well as post production software support (PPSS)
- Utilized data was from a variety of appropriations (see normalization for how this was handled)
- All costs shown are in BY 2016 $
- For regression analysis, the following fit statistics were utilized:
  - $R^2$
  - $P$-value/T-stat/F-stat
  - Standard Error of the Estimate
  - Pred (30)
Limitations

• Data is not from a formal deliverable from a performing organization or vendor. It was provided by programs via the DASA-CE SWM questionnaire.
• Programs have not historically tracked SWM execution costs according to the DASA-CE SWM WBS. Data was often provided at an aggregate level or broken out using SME judgement.
• Due to the nature of the data collection, it is assumed that reported costs are more accurate than reported effort (hours). Future analysis will also utilize effort data.
• It is assumed the super domain is a meaningful way to aggregate data points.
• Given the data sample size, all data points were used for analysis*

*In a few cases outliers were removed, these instances are noted within the analysis
# Super Domain Definitions

## Real-Time

Real-Time is the most constrained type of software. These are specific solutions limited by system characteristics such as memory size, performance, or battery life. These projects take the most time and effort due to constraints.

### Application Domains
- Microcode & Firmware
- Signal Processing
- Vehicle Control/Vehicle Payload
- Other Real-Time Embedded Command & Control Communications

### Examples
- Field Programmable Gate Arrays
- Flight Control
- Missile Control
- Radar Altimeter
- Network Operations
- Signal Electronics
- Tracking Sensors
- Encryption
- Radio Networks
- Propulsion

## Engineering

Engineering software operates under less severe constraints than real-time software. This software may take real-time software outputs and further process them to provide human consumable information or automated control of devices. Or the software may perform transformation and aggregation / distribution of data.

### Application Domains
- System
- Process Control
- Scientific and Simulation
- Test, Measurement, Diagnostic and Evaluation

### Examples
- Operating Systems
- Image processing
- Simulation & Modeling
- Test Equipment
- File Management
- Artificial Intelligence
- Manufacturing Process Control

## Support

Support software assists with operator training and software testing. This software has few constraints.

### Application Domains
- Training
- Software Tools

### Examples
- Computer Based Training
- Compilers
- Programming Aids
- Code Generators
- Assemblers
- Courseware
- Test case generation
- Linker/loaders
- Code Auditors

## AIS

Automated information system software provides information processing services to humans or software applications. These applications allow the designated authority to exercise control and have access to typical business / intelligence processes and other types of information access. These systems also includes software that facilitates the interface and control among multiple COTS / GOTS software applications.

### Application Domains
- Mission Planning
- Custom AIS Software
- Enterprise Service Systems
- Enterprise Information Systems

### Examples
- Scenario Generators
- Target Planning
- Enterprise Service Management
- Enterprise Resource Planning
- Transaction Processing
- Data Warehousing
- Financial Transactions
Cost Allocation Across the SWM WBS
All Phase I Systems

Sample Size: 43 Systems
113 Data Points

- Pgm Mgmt 7.5%
- Lic Mgmt 0.4%
- Lic ODC 6.9%
- C&A 11.5%
- Fac 9.4%
- Fac ODC 1.9%
- SW Change Product 30.7%
- Op Mgmt 5.3%
- FSE 8.9%
- Sust Engr 17.6%

UNCLASSIFIED
Distribution Statement A: Approved for Public Release: Distribution is Unlimited
## Cost Allocation by Super Domain

<table>
<thead>
<tr>
<th>Category</th>
<th>SW Change</th>
<th>Program Mgmt</th>
<th>SW Lic</th>
<th>C&amp;A</th>
<th>Facilities</th>
<th>Sust Engr</th>
<th>FSE</th>
<th>Oper Mgmt</th>
</tr>
</thead>
<tbody>
<tr>
<td>Engineering</td>
<td>35%</td>
<td>5%</td>
<td>9%</td>
<td>10%</td>
<td>10%</td>
<td>14%</td>
<td>14%</td>
<td>3%</td>
</tr>
<tr>
<td>Real Time</td>
<td>28%</td>
<td>6%</td>
<td>5%</td>
<td>12%</td>
<td>16%</td>
<td>20%</td>
<td>9%</td>
<td>4%</td>
</tr>
<tr>
<td>Support</td>
<td>28%</td>
<td>20%</td>
<td>14%</td>
<td>4%</td>
<td>4%</td>
<td>19%</td>
<td>2%</td>
<td>9%</td>
</tr>
<tr>
<td>AIS</td>
<td>31%</td>
<td>10%</td>
<td>7%</td>
<td>16%</td>
<td>5%</td>
<td>15%</td>
<td>6%</td>
<td>11%</td>
</tr>
</tbody>
</table>
Distribution of Annual Cost
System Annual Cost (WBS 1.0 - 8.0)

- **N**: 122
- **Min**: $117,428
- **Q1**: $2,350,442
- **Median**: $5,250,527
- **Q3**: $11,948,232
- **Max**: $103,731,995
## System Annual Cost Summary

<table>
<thead>
<tr>
<th>SD</th>
<th>N</th>
<th>Minimum</th>
<th>Q1</th>
<th>Median</th>
<th>Q3</th>
<th>Maximum</th>
</tr>
</thead>
<tbody>
<tr>
<td>AIS</td>
<td>16</td>
<td>$2,783,335</td>
<td>$9,727,349</td>
<td>$15,301,506</td>
<td>$45,125,746</td>
<td>$66,448,489</td>
</tr>
<tr>
<td>ENG</td>
<td>38</td>
<td>$250,732</td>
<td>$2,320,639</td>
<td>$4,773,907</td>
<td>$10,734,471</td>
<td>$24,870,059</td>
</tr>
<tr>
<td>RT</td>
<td>55</td>
<td>$117,428</td>
<td>$1,363,244</td>
<td>$4,323,691</td>
<td>$10,355,772</td>
<td>$103,731,995</td>
</tr>
<tr>
<td>SUP</td>
<td>13</td>
<td>$3,729,674</td>
<td>$4,363,762</td>
<td>$6,003,356</td>
<td>$7,467,262</td>
<td>$9,120,451</td>
</tr>
</tbody>
</table>
Cost Estimating Relationships
Exploratory Data Analysis

Scatter plots at the top level show significant variance. Phase II should reduce variance and allow analysis on meaningful data subsets.
## Initial Phase I CERs

<table>
<thead>
<tr>
<th>Dependent Variable</th>
<th>Equation</th>
<th>Super Domain</th>
<th>R²</th>
<th>Sample Size</th>
<th>PRED(30)</th>
<th>SEE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Duration</td>
<td>$1.355 \times Req^{0.3323}$</td>
<td>AIS</td>
<td>60.04%</td>
<td>16</td>
<td>12.50%</td>
<td>4.7</td>
</tr>
<tr>
<td>Total Rel Cost</td>
<td>$6.981 \times New_Mod^{0.4004} \times Dur^{0.755}$</td>
<td>All</td>
<td>65.10%</td>
<td>43</td>
<td>27.91%</td>
<td>7,260,456</td>
</tr>
<tr>
<td>Total Rel Cost</td>
<td>$1.955 \times Dur^{0.6423} \times New_Mod^{0.5382} \times 1.796^{RTDummy}$</td>
<td>All - Outliers Removed*</td>
<td>81.10%</td>
<td>39</td>
<td>25.64%</td>
<td>5,941,321</td>
</tr>
<tr>
<td>Total Rel Cost</td>
<td>$2.878 \times Dur^{0.8052} \times New_Mod^{0.4938}$</td>
<td>All - Outliers Removed*</td>
<td>79.70%</td>
<td>39</td>
<td>35.90%</td>
<td>6,032,352</td>
</tr>
<tr>
<td>Ctr Hours</td>
<td>$24.49 \times New_Mod^{0.624}$</td>
<td>All Non-IAVA</td>
<td>75.15%</td>
<td>38</td>
<td>26.53%</td>
<td>51,539</td>
</tr>
<tr>
<td>Total Hours</td>
<td>$43.35 \times New_Mod^{0.5932}$</td>
<td>All Non-IAVA</td>
<td>71.75%</td>
<td>47</td>
<td>19.12%</td>
<td>180,076</td>
</tr>
<tr>
<td>Total Hours</td>
<td>$34.67 \times New_Mod^{0.5911}$</td>
<td>ENG</td>
<td>76.47%</td>
<td>23</td>
<td>21.74%</td>
<td>44,340</td>
</tr>
<tr>
<td>Total Rel Cost</td>
<td>$22,159 \times New_Mod^{0.4362}$</td>
<td>ENG</td>
<td>73.00%</td>
<td>14</td>
<td>21.43%</td>
<td>3,506,848</td>
</tr>
<tr>
<td>Total Rel Cost</td>
<td>$28,941 \times ESLOC^{0.413}$</td>
<td>ENG</td>
<td>72.80%</td>
<td>14</td>
<td>21.43%</td>
<td>3,093,766</td>
</tr>
<tr>
<td>Ctr Hours</td>
<td>$29.58 \times New_Mod^{0.5851}$</td>
<td>ENG</td>
<td>72.34%</td>
<td>20</td>
<td>15.00%</td>
<td>37,164</td>
</tr>
<tr>
<td>Cost per Month</td>
<td>$65,626 + 10.82\times\text{New_Mod}$</td>
<td>RT</td>
<td>79.63%</td>
<td>23</td>
<td>34.78%</td>
<td>174,130</td>
</tr>
<tr>
<td>Total Rel Cost</td>
<td>$4,775 \times New_Mod^{0.4554} \times Dur^{0.764}$</td>
<td>RT</td>
<td>72.00%</td>
<td>27</td>
<td>22.22%</td>
<td>7,332,110</td>
</tr>
<tr>
<td>Total Rel Cost</td>
<td>$2,697 \times ESLOC^{0.3728} \times Dur^{1.058}$</td>
<td>RT</td>
<td>68.10%</td>
<td>28</td>
<td>28.57%</td>
<td>7,495,672</td>
</tr>
<tr>
<td>Total Hours</td>
<td>$939.51 \times SC^{0.5177}$</td>
<td>SUP</td>
<td>89.91%</td>
<td>13</td>
<td>61.54%</td>
<td>5,309</td>
</tr>
<tr>
<td>Ctr Hours</td>
<td>$794.69 \times SC^{0.516}$</td>
<td>SUP</td>
<td>88.59%</td>
<td>13</td>
<td>69.23%</td>
<td>5,126</td>
</tr>
<tr>
<td>Total Rel Cost</td>
<td>$47,858 \times SC^{0.3267} \times Dur^{0.516}$</td>
<td>SUP</td>
<td>75.50%</td>
<td>13</td>
<td>46.15%</td>
<td>242,287</td>
</tr>
<tr>
<td>Total Rel Cost</td>
<td>$123,588 \times SC^{0.3847}$</td>
<td>SUP</td>
<td>64.90%</td>
<td>14</td>
<td>28.57%</td>
<td>393,099</td>
</tr>
</tbody>
</table>

### Explanation of Variables:

ESLOC = Equivalent Source Lines of Code  
New\_Mod = Sum of New and Modified Lines of Code  
SC = Software Change Count (Problem Reports, Defects, Issues, Change Requests, etc.)  
Dur = Release Duration in months  
Req = Software Requirements (SRS equivalent requirements)  

*All CERs shown have a p-value < .005  
Min/Max values for each coefficient are shown in backup
Cost Benchmarks
**Cost Per Software Change**

**Support and Engineering Super Domain**

- Cost per Software Change is shown by Super Domain
- Software change count only includes program reported software changes. It does not separately include IAVA counts
- Software Changes are also commonly referred to as problem reports, change requests, defects, etc.

### Support (Cost in BY 2016 $)

<table>
<thead>
<tr>
<th></th>
<th>Sample</th>
<th>Log Normal</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mean</td>
<td>36,095.37</td>
<td>39,483.68</td>
</tr>
<tr>
<td>Std Dev</td>
<td>35,829.35</td>
<td>39,076.58</td>
</tr>
<tr>
<td>CV</td>
<td>0.99</td>
<td>0.99</td>
</tr>
<tr>
<td>Min</td>
<td>2,915.83</td>
<td></td>
</tr>
<tr>
<td>Mode</td>
<td></td>
<td>14,177.15</td>
</tr>
<tr>
<td>Max</td>
<td>109,267.40</td>
<td></td>
</tr>
<tr>
<td>Count</td>
<td>13</td>
<td></td>
</tr>
<tr>
<td>Standard Error of Estimate</td>
<td>11,687.15</td>
<td></td>
</tr>
</tbody>
</table>

### Engineering (Cost in BY 2016 $)

<table>
<thead>
<tr>
<th></th>
<th>Sample</th>
<th>Log Normal</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mean</td>
<td>203,834.97</td>
<td>240,949.52</td>
</tr>
<tr>
<td>Std Dev</td>
<td>329,675.15</td>
<td>795,858.95</td>
</tr>
<tr>
<td>CV</td>
<td>1.62</td>
<td>3.30</td>
</tr>
<tr>
<td>Min</td>
<td>3,507.83</td>
<td></td>
</tr>
<tr>
<td>Mode</td>
<td>5,862.26</td>
<td></td>
</tr>
<tr>
<td>Max</td>
<td>1,336,963.22</td>
<td></td>
</tr>
<tr>
<td>Count</td>
<td>16</td>
<td></td>
</tr>
<tr>
<td>Standard Error of Estimate</td>
<td>41,198.89</td>
<td></td>
</tr>
</tbody>
</table>
Cost per Software Change is shown by Super Domain

One data point was removed from the Real Time dataset for this chart. See backup for distribution with outlier included.

Software change count only includes program reported software changes. It does not separately include IAVA counts.

Software Changes are also commonly referred to as problem reports, change requests, defects etc.
Cost per IAVA

- Only Information Assurance Vulnerability Alert (IAVA) releases were used, which is a subset of the release data set
- Graph represents (IAVA release cost) / (IAVA count for the release)
  - Includes government and contractor effort

### Cost in BY 2016 $ Table

<table>
<thead>
<tr>
<th></th>
<th>Sample</th>
<th>Log Normal</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mean</td>
<td>7,546.79</td>
<td>7,608.06</td>
</tr>
<tr>
<td>Std Dev</td>
<td>3,767.77</td>
<td>3,783.15</td>
</tr>
<tr>
<td>CV</td>
<td>0.49</td>
<td>0.49</td>
</tr>
<tr>
<td>Min</td>
<td>2,659.91</td>
<td></td>
</tr>
<tr>
<td>Mode</td>
<td></td>
<td>5,461.81</td>
</tr>
<tr>
<td>Max</td>
<td>17,623.77</td>
<td></td>
</tr>
<tr>
<td>Count</td>
<td>40</td>
<td></td>
</tr>
<tr>
<td>Standard Error of Estimate</td>
<td>537.34</td>
<td></td>
</tr>
</tbody>
</table>
Measurement Benchmarks
Graph illustrates the number of IAVAs per license per year for each Super Domain
IAVA release rhythm is different for each program. Data is normalized to a yearly amount
Two Outliers removed (Engineering and Support Domains). See backup for analysis with outliers included
DSLOC per FTE

- DSLOC represents Delivered Source Lines of Code which counts all code equally
- The earliest baseline size reported was used to represent DSLOC
- Full Time Equivalent (FTE) counts were derived by including the following WBS Elements: SW Change Product (1.0), Program Management (2.0), Sustaining Engineering (5.0), and Certification and Accreditation (4.0)
- FTEs were derived by using labor hours per man-year and labor rate reported for each program
- Only Real Time and Engineering had sufficient data to derive DSLOC/FTE

### Sample Size

<table>
<thead>
<tr>
<th></th>
<th>RT</th>
<th>ENG</th>
</tr>
</thead>
<tbody>
<tr>
<td>Min</td>
<td>877.8</td>
<td>3,983.1</td>
</tr>
<tr>
<td>Q1</td>
<td>1,558.4</td>
<td>12,687.0</td>
</tr>
<tr>
<td>Median</td>
<td>6,736.0</td>
<td>21,436.4</td>
</tr>
<tr>
<td>Q3</td>
<td>18,534.1</td>
<td>41,624.9</td>
</tr>
<tr>
<td>Max</td>
<td>80,734.1</td>
<td>55,863.18</td>
</tr>
<tr>
<td>Mean</td>
<td>13,501.5</td>
<td>26,171.9</td>
</tr>
</tbody>
</table>
Baseline percent change was calculated for each release as follows:
- \((\text{New Code} + \text{Modified code}) / \text{Delivered Code (DSLOC)}\)
- The earliest baseline size reported was used to represent DSLOC
- Only Real Time and Engineering had sufficient data to derive Baseline Percent Change

<table>
<thead>
<tr>
<th></th>
<th>RT</th>
<th>ENG</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sample Size</td>
<td>38</td>
<td>18</td>
</tr>
<tr>
<td>Min</td>
<td>0.2%</td>
<td>0.2%</td>
</tr>
<tr>
<td>Q1</td>
<td>1.5%</td>
<td>0.7%</td>
</tr>
<tr>
<td>Median</td>
<td>3.4%</td>
<td>6.6%</td>
</tr>
<tr>
<td>Q3</td>
<td>16.5%</td>
<td>21.9%</td>
</tr>
<tr>
<td>Max</td>
<td>44.6%</td>
<td>37.5%</td>
</tr>
<tr>
<td>Mean</td>
<td>10.3%</td>
<td>12.3%</td>
</tr>
</tbody>
</table>
Summary of Phase I Data Issues & Way Ahead

Phase I Data Issues

- Lack of standardized process for data collection for software maintenance
- Inability to map executed cost/effort data to software maintenance output activities and software change products
- Volatile change requirements and execution priorities hinder execution tracking
- Multiple funding streams are often separately managed
- For many systems, the government is heavily leveraged on contractors which limits insight into cost data

Way Ahead

- Phase II data collection (in-progress)
  - Phase II includes an additional 196 Army programs
  - Examples of future research using Phase II data:
    - Refined CERs by application domain, organization, operating environment, etc.
    - Schedule Estimating Relationships (SERs)
    - Release rhythm analysis
    - Release characterization (enhancement, defects, cybersecurity) analysis on WBS 1.0 SW Change Product
    - Software Maintenance cost model
  - Phase II data will be used to validate CERs and Phase I analysis
SWM Data Demographics

Overview

<table>
<thead>
<tr>
<th>Measure</th>
<th>Phase I</th>
<th>Phase II</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of Programs</td>
<td>56</td>
<td>97*</td>
</tr>
<tr>
<td>Total Cost Captured</td>
<td>$683,974,500</td>
<td>$2,108,960,500</td>
</tr>
</tbody>
</table>

* Approx. 100 additional records will be added to the database.

Number of Releases

<table>
<thead>
<tr>
<th>Super Domain</th>
<th>Phase I</th>
<th>Phase II</th>
</tr>
</thead>
<tbody>
<tr>
<td>AIS</td>
<td>36</td>
<td>197</td>
</tr>
<tr>
<td>ENG</td>
<td>66</td>
<td>180</td>
</tr>
<tr>
<td>RT</td>
<td>89</td>
<td>299</td>
</tr>
<tr>
<td>SUP</td>
<td>19</td>
<td>19</td>
</tr>
<tr>
<td>Total</td>
<td>210</td>
<td>695</td>
</tr>
</tbody>
</table>

Size Measures by Release

<table>
<thead>
<tr>
<th>Measure</th>
<th>Phase I</th>
<th>Phase II</th>
</tr>
</thead>
<tbody>
<tr>
<td>SLOC</td>
<td>79</td>
<td>388</td>
</tr>
<tr>
<td>SC</td>
<td>66</td>
<td>30</td>
</tr>
<tr>
<td>IAVAs</td>
<td>240</td>
<td>41</td>
</tr>
<tr>
<td>Other</td>
<td>21</td>
<td>33</td>
</tr>
</tbody>
</table>

Size Measures Reported by Programs

<table>
<thead>
<tr>
<th>Measure</th>
<th>Phase I</th>
<th>Phase II</th>
</tr>
</thead>
<tbody>
<tr>
<td>SLOC</td>
<td>30</td>
<td>41</td>
</tr>
<tr>
<td>SC</td>
<td>19</td>
<td>23</td>
</tr>
<tr>
<td>IAVAs</td>
<td>19</td>
<td>3</td>
</tr>
<tr>
<td>Other</td>
<td>15</td>
<td>4</td>
</tr>
</tbody>
</table>
Implementing the SRDR-M

- **Systemic data collection**
  - The Software Resources Data Reporting for Maintenance (SRDR-M*) closely aligns to the DASA-CE SWM WBS and data requirements
  - Moving forward, the SRDR-M will be utilized to collect SWM data from a large number of programs across the Army

- **Challenges with implementation**
  - Army contracting strategy for sustainment does not lend itself to strategic CSDR planning
  - Policy for reporting for ACAT II/III programs (acquisition vs sustainment policy)
  - No standardized government labor tracking for organizations performing SWM
  - Cost model/training required for sustainment community

*See [http://cade.osd.mil/policy/dids](http://cade.osd.mil/policy/dids) for more information*
<table>
<thead>
<tr>
<th>Contributors</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>James Judy</td>
<td>NISEC Division Chief</td>
<td>ODASA-CE</td>
</tr>
<tr>
<td></td>
<td>703-697-1612</td>
<td></td>
</tr>
<tr>
<td>Cheryl Jones</td>
<td>Software Measurement Analyst</td>
<td>US Army ARDEC</td>
</tr>
<tr>
<td></td>
<td>973-724-2644</td>
<td></td>
</tr>
<tr>
<td>John Staiger</td>
<td>Principal Consultant</td>
<td>Quantitative Software Management, Inc.</td>
</tr>
<tr>
<td></td>
<td>(703) 790-0055</td>
<td></td>
</tr>
<tr>
<td>James Doswell</td>
<td>Senior Operations Research Analyst</td>
<td>ODASA-CE</td>
</tr>
<tr>
<td></td>
<td>703-697-1572</td>
<td></td>
</tr>
<tr>
<td>John McGarry</td>
<td>Software Measurement and Analysis Lead</td>
<td>US Army ARDEC</td>
</tr>
<tr>
<td></td>
<td>973-724-2644</td>
<td></td>
</tr>
<tr>
<td>Robert Charette</td>
<td>President</td>
<td>ITABHI Corporation</td>
</tr>
<tr>
<td></td>
<td>(540) 972-8150</td>
<td></td>
</tr>
<tr>
<td>Jenna Meyers</td>
<td>Senior Operations Research Analyst</td>
<td>ODASA-CE</td>
</tr>
<tr>
<td></td>
<td>703-697-1645</td>
<td></td>
</tr>
<tr>
<td>Brad Clark</td>
<td>Vice President</td>
<td>Software Metrics, Inc.</td>
</tr>
<tr>
<td></td>
<td>(703) 754-0115</td>
<td></td>
</tr>
<tr>
<td>Doug Putnam</td>
<td>Principal Consultant</td>
<td>Quantitative Software Management, Inc.</td>
</tr>
<tr>
<td></td>
<td>(703) 790-0055</td>
<td></td>
</tr>
<tr>
<td>Colin Strataakes</td>
<td>Operations Research Analyst</td>
<td>ODASA-CE</td>
</tr>
<tr>
<td></td>
<td>703-697-1626</td>
<td></td>
</tr>
</tbody>
</table>

UNCLASSIFIED
Distribution Statement A: Approved for Public Release: Distribution is Unlimited
Apples and Oranges
An experience in using cloud
Cost Calculators and Rate Cards

3rd annual Software and IT-CAST forum
22-24 August 2017

Presenters:
Kevin Buck
Daniel Harper
Charlie Dobbs

Additional contributors:
Tony Dziepak
John Metcalf
Eric Krebs
Outline

- Motivation and Approach
- Our Application Questionnaire
- Calculators and Tools Considered
- Cost Estimate Comparison
- Comparing Apples and Oranges
- Lessons Learned
- Next Steps
Motivation and Approach

Team tasked with achieving DoD sponsor’s objectives:
- Instantiate an on-premises, contractor-owned, contractor-operated (COCO) cloud pilot
- Better understand cloud cost, schedule, and performance implications

Assessed reasonableness of sponsor Independent Government Cost Estimate (IGCE) by comparing to available cost estimate benchmarks
- Team input candidate systems’ data into over a dozen calculators and rate cards for estimating storage and hosting costs for cloud applications

Evaluated relationship between application complexity and cloud cost
- Developed an Application Complexity Plotter to visualize complexity

Began developing a parameterized cloud cost model that could support Total Ownership Cost (TOC) assessment, Return-on-Investment (ROI) analysis, and “what if” scenario-building
Our Application Questionnaire

To select, prioritize, and plan

- App Type
- User Locations
- Accreditation
- Impact Level
- Criticality
- NIPR/SIPR
- User Types
- # of Users
- Demand Volatility
- Virtualization
- Operating System
- Load Balancers
- Type and Number
- Hard Coding
- Licensing
- Cores
- Server Types/Qty
- RAM
- System Utilization
- Network Utilization
- Method
- Connection Speed
- Peak Rate
- App Storage
- DB Storage
- Logs Allocation
- App Type
- User Locations
- Accreditation
- Impact Level
- Criticality
- NIPR/SIPR
- User Types
- # of Users
- Demand Volatility
- Virtualization
- Operating System
- Load Balancers
- Type and Number
- Hard Coding
- Licensing
- Cores
- Server Types/Qty
- RAM
- System Utilization
- Network Utilization
- Method
- Connection Speed
- Peak Rate
- App Storage
- DB Storage
- Logs Allocation
- Refactoring Risks
- Migration Risks
- Documentation Quality
- Encryption
- Identity Mgmt
- Authentication
- Requirements
- Backup Size
- Svc Continuity
- Other complexity considerations

Complexity?
Candidate Application Data

- Received a wide variety of data on ~30 systems from multiple commands and CONUS geographies, complexities (low-high)
- Key usable inputs across the various calculators were number of cores*, required memory/RAM, and required storage

A core is the central processing unit (CPU) that executes sequential instructions. A single silicon chip can have as many as 22 cores. A core is the basic computation unit of the CPU.
Status Quo O&S Cost for Some Apps

- Received one year of status quo operations cost for 23 systems

Collection of status quo apps ops costs provided insight into costs, but lacked fidelity.
Early Analysis: Number of Cores as a Predictor of Memory & Storage Cost

Circle color corresponds to memory as a multiple of # cores
(Green=small; gold=medium; purple=large)

Circle size corresponds to storage as a multiple of # cores
Blue line is the cost line by core using the standard 2 x GB memory and typical storage
Calculators and Tools Considered

Online calculators:
- FEDRAMP GovCloud Shopper
- Cloud.gov
- Microsoft Azure online calculator
- Amazon Web services online calculator
- Google Web services calculator
- Cloudorado

Rate cards/Spreadsheet calculators:
- DISA MilCloud Rate Card
- DOD Rate Card Estimator (draft)
- Navy Cloud Store (AWS)
- LOGSA Rate Card
- GSA IAAS Estimator
- Cloud Cost Lite-MITRE developed tool
- Technology Insertion Model (MITRE-developed tool with migration component)
- DOD CIO Cloud Calculator (in development)

Individual vendor rates
Commercial Parametric Models
Rate Cards…User Beware!

- Rate cards should be taken with a grain of the proverbial salt
- IT Shops sometimes provide "rate cards" or catalogs that provide costs for various services.
- Unfortunately, frequently little to no context is provided, And there may be little insight as to how inputs are applied ("blackbox"). Excel-based calculators were more transparent, but insight was still lacking.
- For example, the DISA Rate Card is a single pdf spreadsheet listing prices for various services, with very little explanation.
- Some sort of estimating methodology has to be assumed/created, as well as a discussion with the maker of the rate card to get context.

![2016 DISA Rate card](image)
Cost Estimate Comparison

We compared the sponsor’s IGCE to 17 calculator estimates.

Hosting Costs for 34 Apps for Current Year
Estimates from Available Rate Cards and Tools

The Independent Government Cost Estimate (IGCE) is 81% less than the Status Quo ($20M) and is 3% less than the average of other comparative benchmarks.

Compare the IGCE (excluding integration) cost of $3,770,704 to the average of the estimates that are definitely not off-premises, public, or PaaS = $3,879,492.

IGCE estimate excluding integration cost = $3.8M
To assess the reasonableness of an IGCE, developed ROM estimates using available tools, calculators, and rate cards from Government, DoD, and commercial industry.

The current legacy status quo estimate is approximately $20M/year for 34 apps. In a Cloud environment, all comparison benchmarks are less than $10M/year, representing more than a 50% reduction in expected cost.

The second stacked bar from the left represents the IGCE. $3.8M per year to host 34 apps compares reasonably with other benchmarks.
Comparing Apples and Oranges

- The cloud achieves efficiencies through **standardization**, shared resources and commoditization.

Items may be bundled differently by vendor and by model

- Many vendors include or leave out items that do not vary in their standard offerings

- Most vendors will offer a custom quote for non-standard items

- Apples and Oranges can still be compared. They must first be normalized as much as possible.
### Calculator/Tool Inputs

#### Partial Universe of Cost Calculator/Rate Card Inputs

<table>
<thead>
<tr>
<th></th>
<th>1. # of User Accounts</th>
<th>10. Req. RAM: DB</th>
<th>19. vCPUs per instance</th>
<th>28. PUT requests</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.</td>
<td># Concurrent Users</td>
<td>11. # cores: Web/App</td>
<td>20. Cores per instance</td>
<td>29. OS</td>
</tr>
<tr>
<td>3.</td>
<td>Types of Servers</td>
<td>12. # cores: DB</td>
<td>21. RAM per instance</td>
<td>30. # additional elastic IPs</td>
</tr>
<tr>
<td>6.</td>
<td>Logs allocation</td>
<td>15. CPU Power</td>
<td>24. Web Storage</td>
<td>33. total hrs per month</td>
</tr>
<tr>
<td>7.</td>
<td>Current system utilization rate</td>
<td>16. Web app instances</td>
<td>25. Transfer in-GB</td>
<td>34. VM class</td>
</tr>
<tr>
<td>9.</td>
<td>WAN bandwidth for currently traversed circuits</td>
<td>18. Instance Type</td>
<td>27. GET requests</td>
<td>36. Available local Solid State Drive (SSD) space</td>
</tr>
</tbody>
</table>

---

**Of these, here are the inputs that appear to matter the most**

- **Almost all models ask about:**
  - Compute
  - Memory
  - Storage

- **The majority of models ask about:**
  - Transmission
    - Labor inducing factors (like # of VMs, PaaS)
  - Utilization
  - Operating System (i.e. Windows, Linux)

- **Occasionally models/rate cards prompt questions on:**
  - Software & Server Types
  - Value added offerings e.g.,
    - Architecture
    - Monitoring
    - Security

**Inputs that Typically Drive Costs are shown by red ovals**
Lessons Learned about Calculators, Rate cards, and Cloud Cost Tools

- Most models only calculate annual recurring costs, with no allowance for storage and compute growth year-over-year
- Models do not estimate other major cost elements such as: system engineering and program management; integration and test; security-related costs; professional/managed services, migration costs
- Some tools include an option to estimate Disaster Recovery, COOP, and some additional professional services
- On versus off-premise considerations were not inputs to most calculators
- Private vs. public considerations were not inputs to most calculators
- Few tools include cost for uncertainty/risk
- Some models do not use cloud impact level (DOD-specific term) but instead use Federal Information Security Management Act (FISMA); others had no security variable
Next Steps: Parameterized Lifecycle Model for Cloud

MODEL INPUT

Cloud Hosting Options

Deployment Models:
- IaaS
- PaaS
- SaaS

Service Models:
- Private
- Community
- Public

Hosting Requirements

Current Cost
Basis of Estimate
GR&A

Functional Drivers of Cloud Cost

Objective

Subjective

MODEL TRANSFORM

Application Complexity Plotter (Next slide)

MODEL OUTPUT

Insourced vs Outsourced:
- Storage $
- Compute $
- Transmit $
- Managed Service $
- Other – Optional

CBA

BCA

ROI Calculation
Suite of Metrics

Payback Period

Internal Rate of Return (IRR)

Cost/Benefit Ratio

Uncertainty Assessment

Reporting

What-If Calculator

Pay per Use

IGCE

LCCE
Next Steps: Application Complexity as an Indicator of Cloud Cost Impact

XYZ Application

XYZ App overall: 6.0
Average App overall: 5.5
Backup
NIST Cloud Definition

Deployment Models
- Private Cloud
- Community Cloud
- Public Cloud
- Hybrid Clouds

Service Models
- Infrastructure as a Service (IaaS)
- Platform as a Service (PaaS)
- Software as a Service (SaaS)

Essential Characteristics
- On Demand Self-Service
  - Broad Network Access
  - Resource Pooling
  - Rapid Elasticity
  - Measured Service

Common Characteristics
- Massive Scale
- Homogeneity
- Virtualization
- Low Cost Software
- Resilient Computing
- Geographic Distribution
- Service Orientation
- Advanced Security
NIST Cloud Services

Public Sector Management of XaaS Platforms

- Traditional IT
  - Applications
  - Data
  - Runtime
  - Middleware
  - Operating System
  - Virtualization
  - Servers
  - Storage
  - Networking

- Infrastructure (as a Service)
  - Applications
  - Data
  - Runtime
  - Middleware
  - Operating System
  - Virtualization
  - Servers
  - Storage
  - Networking

- Platform (as a Service)
  - Applications
  - Data
  - Runtime
  - Middleware
  - Operating System
  - Virtualization
  - Servers
  - Storage
  - Networking

- Software (as a Service)
  - Applications
  - Data
  - Runtime
  - Middleware
  - Operating System
  - Virtualization
  - Servers
  - Storage
  - Networking

Legend:
- You manage
- Delivered as a service

Source: Adapted from IDC Government Insights
Lessons Learned about our Analysis

- **It’s a pilot—we will learn from it!**
  For this project, we had to remind ourselves that the reason our customer was conducting a pilot in the first place was because we did not have all the answers—including what it would cost.

- **Inform the customer to manage expectations.**
  In our customer’s case, they were going to a private, on-premises cloud; much of the cost savings associated with the public cloud (due to amortized costs over multiple customers) would not be realized.
  
  - Note cost differences (e.g., high upfront costs, less realized savings) as well as benefits (e.g., higher security)

- **No formal survey existed** that we could find comparing multiple calculators, though several had examined AWS & DISA.

- **Use an RFI** as a tool to gather information directly from vendors

- **Access to cloud subject matter experts** key
### Google Cloud Platform Pricing Calculator

<table>
<thead>
<tr>
<th>Feature</th>
<th>Description</th>
<th>Unit</th>
<th>Cost</th>
</tr>
</thead>
<tbody>
<tr>
<td>Compute Engine</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Region</td>
<td>US</td>
<td></td>
<td></td>
</tr>
<tr>
<td>VM class</td>
<td>regular</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Instance type</td>
<td>n1-standard-2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Cores per instance</td>
<td>2 units</td>
<td></td>
<td></td>
</tr>
<tr>
<td>RAM per instance</td>
<td>7.5 GB</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Paid OS Cost (Windows)</td>
<td>$32,236.80</td>
<td></td>
<td></td>
</tr>
<tr>
<td>GCE Instance Cost</td>
<td>$28,207.20</td>
<td></td>
<td></td>
</tr>
<tr>
<td>total hrs per month</td>
<td>402,960 hrs/mo</td>
<td></td>
<td></td>
</tr>
<tr>
<td>price per instance</td>
<td>$0.28/hr</td>
<td></td>
<td></td>
</tr>
<tr>
<td>monthly price</td>
<td>$208.00/mo</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Total available local SSD space</td>
<td>1x375 GB</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Sustained use discount</td>
<td>30%</td>
<td></td>
<td></td>
</tr>
<tr>
<td>effective hourly rate</td>
<td>$0.26/hr</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Estimated monthly cost</td>
<td>$105,777.00/mo</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Persistent Disk (Storage)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>storage</td>
<td>288,870 GB</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Estimated monthly cost</td>
<td>$11,554.80</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Load Balancing (global)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Forwarding rules</td>
<td>68</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Network ingress/egress</td>
<td>3200 GB</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Estimated monthly cost</td>
<td>$503.75</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Network Bandwidth</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Egress to different zone in same region</td>
<td>100 GB</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Egress to different region within the US</td>
<td>100 GB</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Estimated monthly cost</td>
<td>$2.00</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Total Estimated Monthly Cost</td>
<td>$117,837.55/mo</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Total Estimated Annual Cost</td>
<td>$1,414,050.60/yr</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Clouderado Pricing Tool-the “Kelly Blue Book” of Clouds

- Key Inputs:
  - VM Size, Qty, Storage (Doubling storage doubles $)
- FISMA low/moderate only.
- Backup/storage computed together.
- Provides low-average high range
- Also provides prices for utility cloud services
- Considers 19 vendors
Discovering the Rosetta Stone: A Strategic Method to Software Sizing

Presented at the Software IT-CAST Meeting

August 23, 2017
Crystal City, VA
INTRODUCTION

VS.
Contents

• Sizing Overview
• Rosetta Stone Approach to Sizing
• Gearing Factors Research
• Theory into Practice
Here are three different approaches to estimating projects. All methods have value. You can use more than one approach, depending on the lifecycle phase and the data available.

<table>
<thead>
<tr>
<th>Estimate Based On:</th>
<th>Role-Based</th>
<th>Task-Based</th>
<th>Scope-Based</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Roles and skills</strong> required to build the product</td>
<td><strong>Activities</strong> necessary to build the product</td>
<td>Attributes of the <strong>product</strong> to be built</td>
<td></td>
</tr>
<tr>
<td><strong>Estimation Strength:</strong></td>
<td>Most <em>implicit</em>: depends on task, experience, culture, lifecycle, etc.</td>
<td>Somewhat <em>implicit</em>: affected by lifecycle used, project variables, etc.</td>
<td>Most <em>explicit</em>: driven by what has to be delivered</td>
</tr>
</tbody>
</table>
Work Activity Statement

“People, working at some level of productivity, produce a quantity of function or a work product at a level of reliability by the expenditure of effort over a time interval.”

- Lawrence H. Putnam & Ware Myers
SIZING CHALLENGES
What is “Size”?

- A proxy for the functionality and knowledge content of the delivered system—what the system is worth

- Size can be indicated by a number of metrics:

**Front end: Unit of Need**  
*Based on characteristics of the statement of needs*
- Requirements
- Function Points/Object Points
- IO Counts
- States/Events/Actions
- Use Cases
- Stories/Story Points/Epics
- Objects/Classes
- Components
- Design Pages
- Web Pages

**Back end: Unit of Work**  
*Based on the characteristics of the system when built*
- Lines of Code
- Statements
- Actions
- Modules
- Subsystems
- GUI Components
- Logic Components
- Logic Gates
- Tables

Each unit has a “relative weight” and precision. Front end units tend to be less precisely defined, while the prediction of the count of back end units tends to be less precise. The relative weight measures the size or “complexity” of the unit and is called a Gearing Factor.
The Sizing Dilemma

The Challenge

People speak vastly different languages when it comes to software product sizing. Excluding ISO standard function point methods, there are no standard definition for the most popular sizing techniques.

Root Causes

Perception of Right vs Wrong Method
- Sizing should make sense to your organization, but depends on empirical measurement.
- Consistency is paramount.

An Emotional Topic
- Important considerations like budget allocation, resource commitments, and schedules are usually inextricably linked to project scope: stakeholders take a vested and serious interest in all of those variables.

Perception that Useful Comparisons Are Impossible
- Organizations that have the mindset “we’re different” miss the opportunity to benchmark their estimates and completed projects against industry data for improved decision making.

Insights

1. Objectively measure the envisioned software output and start with a base unit of both functional and technical size – using the best information you have.
2. Secure sizing inputs and endorsement of those closest to the project who understand what the desired outcome is, and what it might take to achieve it.
3. Translate ballpark size with other sizing techniques so audiences speaking different languages get the same understanding.
Sizing Units Defined

- **Source Lines of Code** (SLOC) - Equal to the new and/or modified code delivered to an end user. Reused and deleted code, blank lines, comment lines, and test scripts are not included. Equal to a Basic Unit of Work.

- **Function Points** – ISO standard method of counting the amount of business functionality an information system provides to a user. Certified FP counters record the number of EI, EO, EQ, ILF, and EIF in a system.

- **Functional Requirements** – Describe the functions that the software is supposed to execute\(^1\). Typically written as “shall” statements\(^2\).

- **Business Requirements** – Higher-level of abstraction that can be useful for initial estimates. Each high-level business requirement is a “container” for multiple lower-level functional requirements.

- **RICEFW Objects** - Common ERP sizing method which includes both custom development (Reports, Interfaces, Conversions, Extensions, Forms, and Workflows) as well as the configuration portion (high-level and detailed business processes or scenarios).

\(^1\)2004 IEEE Software Engineering Body of Knowledge
\(^2\)see IEEE Std 830-1998
Agile Sizing Units Defined

- **Use Cases** – Used in both Agile and traditional development, is a technique documenting functional requirements that (1) describe the interactions between an actor and a software system to achieve a goal, and (2) include a main success scenario as well as extensions that represent alternate paths in the logic flow.¹

- **User Stories** – similar to a single scenario of a use case.² Unlike a use case, user stories do not specify requirement details; they are placeholders for future conversations between developer and customers to quantify the requested functionality. Defects were not included as user stories.

- **Epics** – Epics should be considered at a higher level of abstraction which are still useful for initial size approximations and product estimates. Similar to Homer’s epic, The Odyssey, which is a collection of stories, Agile epics function similarly.

¹ Ivar Jacobson, Object-Oriented Software Engineering: A Use Case Driven Approach, 1992
² Mike Cohn, author of User Stories Applied for Agile Software Development.
Sizing Methods Used at Various Stages of the Software Development Life Cycle

The sizing method used should be based on available information and where you are in the software development life cycle vs. the “Cone of Uncertainty.”

**Ballpark feasibility estimates:**
- Sizing by analogy
- T-shirt sizing

**Initial project-level estimate:**
- Business requirements
- Use cases

**Final project estimate:**
- Functional requirements
- User stories
- Use case scenarios
- Function points

**In-flight forecasting / re-planning:**
- Functional requirements
- User stories
- Use case scenarios
- Function points
- RICEFW objects
- Modules
- Technical components
- Source code files

**Project closeout / benchmarking:**
- Function points
- Source lines of code

Facilitate Communication Between Requirements Authors and Developers

Requirements Author

Developer

Requirement

User Story

User Story

User Story
The Ol’ Bait and Switch
ROSETTA STONE APPROACH
The Rosetta Stone

• Discovered in 1799 near the town of Rashid, the Rosetta Stone translated the same text into three different languages
  – Ancient Egyptian hieroglyphics
  – Demotic script
  – Ancient Greek

• Helped give meaningful understanding of hieroglyphics, which previously could not be understood

• The Rosetta Stone became essential in understanding ancient Egyptian literature and civilization
Applying Rosetta Stone Concept to Software Sizing

How it works:
1. Identify any available early sizing methods and a basic sizing unit
2. Apply gearing factors to the early sizing unit
3. Translate the early size into other popular sizing techniques, so an audience speaking different “languages” gets the same understanding of product size
Applying Gearing Factors

Front End = Function Unit
- Customer language
- Scope definition

Gearing Factor
- Relative complexity

Back End = Basic Unit of Work
- Most elementary development step
- Normalizing size unit

<table>
<thead>
<tr>
<th>Component</th>
<th># of Components</th>
<th>Gearing Factor</th>
</tr>
</thead>
<tbody>
<tr>
<td>Simple Purple Dragons</td>
<td>25</td>
<td>600 SLOC/ PD</td>
</tr>
<tr>
<td>Average Purple Dragons</td>
<td>35</td>
<td>1200 SLOC / PD</td>
</tr>
<tr>
<td>Complex Purple Dragons</td>
<td>10</td>
<td>5500 SLOC / PD</td>
</tr>
</tbody>
</table>
Calculating the Gearing Factor

Unit of Need

Unit of Work

Project

Requirement

RICEFW Object

SLOC

SLOC

SLOC

SLOC

Requirement

RICEFW Object

SLOC

SLOC

SLOC

SLOC

Requirement

RICEFW Object

SLOC

SLOC

SLOC

SLOC

Requirement

RICEFW Object

SLOC

SLOC

SLOC

SLOC

RICEFW Object

SLOC

SLOC

SLOC

SLOC

RICEFW Object

SLOC

SLOC

SLOC

SLOC

RICEFW Object

SLOC

SLOC

SLOC

SLOC

RICEFW Object

SLOC

SLOC

SLOC

SLOC

RICEFW Object

SLOC

SLOC

SLOC

SLOC

RICEFW Object

SLOC

SLOC

SLOC

SLOC
Gearing Factors Research

• Examined a sample of 150+ projects that used two or more sizing methods:
  – SLOC \( (n \approx 150) \)
  – Requirements \( (n \approx 65) \)
  – Function Points \( (n \approx 100) \)
  – Use Cases \( (n \approx 10) \)
  – User Stories \( (n \approx 10) \)
  – RICEFW Objects \( (n \approx 85) \)

• Business, Engineering, and Real Time domains represented from 100+ organizations

• Calculated the average gearing factor ranges for each sizing method

Note: Projects included only those recent, completed and validated across government and industry that reported at least 2 sizing metrics.
Methods

Example: Requirements

- Examined the relationship between functional requirements and a base size unit (SLOC) to calculate gearing factors
- Examined relationship between functional requirements and “container” sizing units (i.e., Business Requirements)
- Certified Function Point Specialist performed a Function Point count on a representative sample of programs that were sized in functional requirements
- Repeated this process for other represented sizing methods
Translating from One Sizing Unit into Others

If one of the sizing units is known, any of these other popular sizing methods can be approximated as well.
PUTTING THEORY INTO PRACTICE
Potential Uses

- Facilitate communication among stakeholders who may think differently about sizing
- Aid discussions around sizing to avoid pitfalls and improve estimates at each stage of the software development lifecycle
- Audit the size estimate if sizing methods change
  - Did the program size grow significantly after changing the sizing method from RICEFW objects to Stories?
  - Is functionality duplicated or lost by changing the sizing method?
- Crosscheck the estimate
Vendor Bid Assessment

- Common application for DoD and Industry
- Evaluating different types of proposals
- Not a silver-bullet, but starts the discussion within the evaluation process and enables comparison

RFI/RFP

Data Request Template
(Size, Schedule, Cost, etc)

Responses Received
### Vendor Bid Assessment

<table>
<thead>
<tr>
<th>Vendor</th>
<th>Sizing</th>
<th>Cost (Effort/Staffing)</th>
<th>Schedule</th>
<th>Risk ?</th>
</tr>
</thead>
<tbody>
<tr>
<td>Alpha</td>
<td>200 RICEFW Objects</td>
<td>2M</td>
<td>12 Mo</td>
<td></td>
</tr>
<tr>
<td>Bravo</td>
<td>3000 Function Points</td>
<td>2M</td>
<td>22 Mo</td>
<td></td>
</tr>
<tr>
<td>Charlie</td>
<td>150 Functional Reqs</td>
<td>5M</td>
<td>16 Mo</td>
<td></td>
</tr>
<tr>
<td>Delta</td>
<td>50 Business Reqs</td>
<td>10M</td>
<td>24 Mo</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Vendor</th>
<th>Sizing</th>
<th>Rosetta Technique</th>
<th>Cost</th>
<th>Schedule</th>
<th>Risk ?</th>
</tr>
</thead>
<tbody>
<tr>
<td>Alpha</td>
<td>200 RICEFW Objects</td>
<td>60k IU</td>
<td>2M</td>
<td>12 Mo</td>
<td></td>
</tr>
<tr>
<td>Bravo</td>
<td>3000 Function Points</td>
<td>150k IU</td>
<td>2M</td>
<td>22 Mo</td>
<td></td>
</tr>
<tr>
<td>Charlie</td>
<td>150k Functional Reqs</td>
<td>25k IU</td>
<td>5M</td>
<td>16 Mo</td>
<td></td>
</tr>
<tr>
<td>Delta</td>
<td>50 Business Reqs</td>
<td>100k IU</td>
<td>10M</td>
<td>24 Mo</td>
<td></td>
</tr>
</tbody>
</table>

With comparable sizing we can begin to discuss reasons for discrepancy and better model feasibility of the vendor's proposed development plan.
Organizational Translator

- Common application for DoD and Industry
- Many stakeholder layers and perspectives often create obstacles for a common sizing picture
- Enables better expectation realism and stakeholder buy-in

VS.
Organizational Translator

- Organization implementing new ERP solution
- Senior leadership wanted to ensure all requirements were met
- Vendor was proposing in their common RICE Objects methodology
- IV&V Team wanted to ensure all aspects of development were being considered among stakeholders (Org reqs, all interface considerations, etc.)
Closing

• Sizing
  – Many metrics available - be consistent
  – Challenges exists (environment, history, business)
• Rosetta Stone Approach to Sizing
  – Methods for translation among stakeholders
• Gearing Factors Research
  – Initial findings provide emerging utility
• Theory into Practice
  – Successful implementation in various environments
SURF Process Summary & Initial Findings: A Deeper Focus on Software Data Quality

Presented by:
Ranae Woods, AFCAA
Dan Strickland, MDA
Nicholas Lanham, NCCA
Marc Russo, NCCA
Haset Gebre-Mariam, NCCA

This document was generated as a result of the AFCAA-led, Software Resource Data Report Working Group (SRDRWG). This working group represented a joint effort amongst all DoD service cost agencies. The following guidance describes SRDR data verification and validation best practices as documented by NCCA, NAVAIR 4.2, AFCAA, ODASA-CE, MDA, and many more.
Table of Contents

• Purpose
• SRDR Need Statement
• SURF Purpose
• SURF Created Process Initiation
• SURF Team Structure
• SURF Verification & Validation (V&V) Guide
• SRDR V&V Process
• SRDR Database
• SRDR Data Quality Review
• SURF Status and Metrics
• Summary
Presentation Purpose

• To familiarize the audience with recent Software Resource Data Report (SRDR) Working Group (WG) efforts to update existing SRDR DID language and implement data quality improvement

• To clarify how these SRDRWG efforts led to the development of a SRDR Unified Review Function (SURF) team

• To highlight:
  − SURF mission
  − Highlight SURF team and Verification and Validation (V&V) guide positive impact on SRDR data quality
SURF Need Statement
Why do these reports need to be reviewed?

- Reduces inaccurate use of historical software data
  - Aligns with OSD CAPE initiative(s) to improve data quality
- Helps correct quality concerns prior to final SRDR acceptance
- Allows a central group of software V&V SMEs to tag SRDR data
- SRDR submissions are used by all DoD cost agencies when developing or assessing cost estimates
- Quality data underpins quality cost and schedule estimates

BBP Principle 2: Data should drive policy. Outside my door a sign is posted that reads, "In God We Trust; All Others Must Bring Data." The quote is attributed to W. Edwards Deming

- Mr. Frank Kendall, AT&L Magazine Article, January-February 2016
Purpose:
- To supplement the Defense Cost Resource Center (DCARC) quality review for SRDR submissions
- To develop a consistent, service-wide set of quality questions for all DoD cost community members to reference
- To provide a consistent, structured list of questions, focus areas, and possible solutions to cost community members tasked with inspecting SRDR data submissions for completeness, consistency, quality, and usability (e.g. SRDR V&V Guide)

Why?
- SURF represents an effort to establish a consistent guide for any organization assessing the realism, quality, and usability of SRDR data submissions
- Quality data underpins quality cost and schedule estimates

Question: What services helped develop the questions included within the latest SRDR V&V guide?
Answer: All services participating in the SRDR WG provided feedback, comments, and reviews over a year long SRDRWG effort focused on establishing higher quality review efforts coupled with an ongoing SRDR DID update
### How Was SURF Created?

**Cost Leadership Forum (CLF) Approved SRDR Initiatives**  
*(Dec 2014)*

<table>
<thead>
<tr>
<th>Recommendation</th>
<th>Benefit</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Revised SRDR Development Data Item Description (DID)</td>
<td>1. Reduces inconsistency, lack of visibility, complexity, and subjectivity in reporting</td>
</tr>
<tr>
<td>2. New SRDR Maintenance Data Item Description (DID)</td>
<td>2. Aligned w/ dev. but w/ unique data/metrics available/desired for maintenance phase</td>
</tr>
<tr>
<td>4. Software Database Initial Design and Implementation Process</td>
<td>4. Avoids duplication, variations - ONE central vs many distributed; Based on surveyed best practices and user expectations</td>
</tr>
</tbody>
</table>

**Question:** How was the SURF team created and is it linked to the SRDRWG?  
**Answer:** Yes. The SRDR Unified Review Function (SURF) team was organized as part of the larger, SRDRWG initiative during 2015.
**SURF Team Structure**

- Team is comprised of one primary member per service along with support from secondary team members (Government Only)
- As submissions are received, SRDR review efforts will be distributed amongst SURF team members to balance workload
- SURF Team Coordinators (STC): Marc Russo & Haset Gebre-Mariam
- Current SURF structure:

  **DCARC Analyst & STC:**
  - SURF Team Coordinators (STC)
    - Marc Russo
    - Haset Gebre-Mariam
  - SRDR Submission received from DCARC
  - SURF Advisor & Process Owner (SAPO)
    - Nick Lanham

  **SURF Primary:**
  - DoD
    - William Raines
  - Navy
    - Corrine Wallshein
    - Wilson Rosa
  - Marine Corps
    - Noel Bishop
  - Air Force
    - Ron Cipressi
  - Army
    - Jim Judy
    - Jenna Meyers
    - James Doswell
  - SPAWAR
    - Jeremiah Hayden
  - MDA
    - Dan Strickland
  - Stephen Palmer
  - Philip Draheim
  - Sarah Lloyd
  - John Bryant
  - Janet Wentworth
  - Chinson Yew
  - Eric Sommer
  - Michael Smith
  - Michael Duarte
  - Min-Jung Gantt

**Question:** How do members get involved with SURF? Why are there “primary” and “secondary” members?

**Answer 1:** The SURF team was established by Government SRDRWG members who were recommended/volunteered by each DoD service.

**Answer 2:** Primary members are included on CSDR S-R IPT email notifications for their specific service. Secondary members are contacted during periods of increased review demands, if necessary.
UNCLASSIFIED

SRDR V&V Guide
Initial Draft Completed in Mar 2015

- Guide represents first-ever, joint effort amongst DoD service cost agencies
  - OSD public release approved 5 April 2016
  - Kickoff email distributed on 1 May 2017 to update guide with latest DID requirements
  - Files can be downloaded using following link: http://cade.osd.mil/roles/reviewers#surf

- Enables ability to consistently isolate software cost relationships and trends based on quality SRDR data
  - Now includes quick-reference MS excel question checklist by SRDR DID section

- Two main purposes:
  - SRDR V&V training guide (V&V questions)
  - Focus areas used to determine SRDR quality tags

Question: Did a standardized-joint service, software-specific quality review guide exist prior to the SURF V&V guide? Who contributed to the development of this document?

Answer 1: No. Services implemented very inconsistent SRDR review methodologies (if conducted at all) prior to DCARC acceptance

Answer 2: The SRDR V&V guide was developed by the SURF team and has been reviewed by numerous SRDRWG, OSD CAPE, and other cost community team members. Feedback from other services has generated significant improvements from initial draft.
1.0 Review of an SRDR submitted to DCARC
   1.1 Reporting Event
   1.2 Demographic Information
   1.3 Software Char. and Dev. Process
      1.3.1 Super Domain and Application Domains
      1.3.2 Operating Environment (OE) Designation
      1.3.3 Development Process
   1.4 Personnel
   1.5 Sizing and Language
      1.5.1 Requirements
      1.5.2 Source Lines of Code (SLOC)
      1.5.3 Non-SLOC Based Software Sizing
      1.5.4 Product Quality Reporting
   1.6 Effort

1.7 Schedule
1.8 Estimate at Completion (EAC) Values
2.0 Quality Tagging
3.0 Solutions for Common Findings
   3.1 Allocation
   3.2 Combining
   3.3 Early Acquisition Phase Combining

4.0 Pairing Data
5.0 Possible Automation
   Appendix A – SD and AD Categories
   Appendix B – Productivity Quality Tags
   Appendix C – Schedule Quality Tags
   Appendix D – SRDR Scorecard Process

V&V Questions and Examples Developed and Organized by Individual SRDR reporting Variable
When assessing Effort, the V&V priority is determining completeness.

Determining completeness is not always easy due to:
- The contractor possibly collecting/reporting their actual performance using categories that differ from the IEEE 12207 standard.
- The contractor reporting all of their Effort within the Other category.

Common questions to ask when looking at the effort are:
- Was effort data reported for each CSCI or WBS?
- Was effort data reported as estimated or actual results? If the submission includes estimated values and actual results, does the report include a clear and documented split between actual results and estimated values?
- Is the effort data reported in hours?
- Is effort data broken out by activity?
- What activities are covered in the effort data? Is there an explanation of missing activities included within the supporting SRDR data dictionary? ....

V&V Guide Includes Specific Questions For SURF Members to Confirm Prior to Accepting the Report
**SURF Team V&V Process**

**Monthly Recurring SURF and DCARC Communications**

**DCARC: Step 1**
- SRDR status list sent to SURF Team Coordinator

**SURF: Step 1**
- SRDR status list distributed to Primary and Secondary POCs

**SURF: Step 2**
- Conduct V&V reviews by populating MS Excel question template

**SURF: Step 3**
- Provide completed V&V question templates back to DCARC

**DCARC: Step 2**
- Combine SURF and DCARC comments
- Coordinate comment resolution with submitting organization

**Database: Step 1**
- Adjudicated SRDR sent to NAVAIR 4.2 for data entry into DACIMs dataset
- Note: Future database(s) will be hosted via CADE

**1st week of every month**

+2 Days

+ 13 Days

NLT + 14 Days

Varies by Contractor

Varies by No. Submissions

**Purpose of SURF Process:** To provide completed V&V checklists to DCARC within 2 weeks of request

**Important Note:** CADE is developing relational databases for new DID formats. Over time, data entry will be automated. Until that time, manual data entry continues by NAVAIR 4.2 team for only the development format. Please refer to V&V guide for additional automation details and future data quality initiatives.
Login to CADE

- http://cade.osd.mil/

Navigate to DACIMs

Select “SRDR Data Library” from folder tree on left side of screen

Filter by “Report As Of Date” to download latest version of dataset

Database to be updated in CADE by end of June 2017

- Quarterly updates to database after June release

Question: Where does SRDR data go after SURF Review?
Answer: Once SRDR record has been accepted, Data is entered into SRDR dataset posted to CADE>DACIMs web portal

Question: Who enters the data into the dataset?
Answer: Currently members from NAVAIR 4.2 enter data to SRDR dataset (10+ years of experience). Future data entry is planned to be automated using .XML schemas linked to latest DID formats
SRDR Data Quality Review
DatasetPosted to OSD CAPE DACIMS Web Portal

- SRDR database is available to Government analysts with access to the CADE portal
  - This dataset is the authoritative source for SRDR data (10+ years of uploads)
- Data is not automatically considered “Good” for analysis
- SURF team may recommend DCARC not accept a submission due several data quality concerns outlined in the V&V guide. Examples include:
  - Roll-up of lower level data (Did not want to double count effect)
  - Significant missing content in hours, productivity, and/or SLOC data missing
  - Interim build actual that is not stand alone
  - Inconsistencies or oddities in the submit
  - Additional reasons discussed in the V&V guide

<table>
<thead>
<tr>
<th>Data Segments</th>
<th>Dec-07</th>
<th>Dec-08</th>
<th>Oct-10</th>
<th>Oct-11</th>
<th>Aug-13</th>
<th>Apr-14</th>
<th>Apr-15</th>
<th>Dec-16</th>
<th>Jun-17</th>
</tr>
</thead>
<tbody>
<tr>
<td>CSCI Records</td>
<td>688</td>
<td>964</td>
<td>1473</td>
<td>1890</td>
<td>2546</td>
<td>2624</td>
<td>2853</td>
<td>3487</td>
<td>3583</td>
</tr>
<tr>
<td>Completed program or build</td>
<td>88</td>
<td>191</td>
<td>412</td>
<td>545</td>
<td>790</td>
<td>911</td>
<td>1074</td>
<td>1326</td>
<td>1391</td>
</tr>
<tr>
<td>Actuals considered for analysis (e.g., “Good”)</td>
<td>0</td>
<td>119</td>
<td>206</td>
<td>279</td>
<td>400</td>
<td>403</td>
<td>682</td>
<td>829</td>
<td>974</td>
</tr>
<tr>
<td>Paired Initial and Final Records</td>
<td>0</td>
<td>0</td>
<td>78</td>
<td>142</td>
<td>212</td>
<td>212</td>
<td>212</td>
<td>240</td>
<td>271</td>
</tr>
</tbody>
</table>
Prior to SURF process, only **15%** of SRDR data was considered “Good”

After one+ year of SURF reviews, **~24%** of data has been tagged as “Good”

Currently, **~27%** of the data had been tagged as “Good”

Army team currently working to review historical data. Once completed, “Good” percentage will likely increase to **~31%**

SURF Team Combined With V&V Guide and DCARC
Have Significantly **Improved** Software Data Quality
Recurring SURF team meetings kicked off on 23 June 2015

- Group includes ~19 Government team members from across the DoD
- Has received very positive feedback from DoD cost estimation community, DCARC analyst(s), and even program office communities since inception
- Completed initial version of SRDR V&V guide March 2015
- Initiated SURF Initial team training using draft V&V guide June 2015
- Completed development of SURF team charter July 2015
- During training period, SURF generated 483 V&V comments provided to DCARC (June 2015 to March 2016)
- Completed official SURF kickoff with DCARC and published V&V guide March 2016
- After training period, formal SURF process generated 889 V&V comments (March 2016 to December 2016)
- Concluding CY16, SURF team generated 1,372 V&V comments from 92 SRDR submissions (1,282 during CY16)

Current Status

- CY17 represents first full year of official SURF reviews using published V&V guide
- Team recently kicked off effort to update existing V&V questions to align with the latest SRDR DID
- Co-chaired Collaborative Cost Research Group (CCRG) focused on increasing "Good" SRDR records (March 2017)
- Continued process improvement efforts to maintain efficient and effective process
- Working with DCARC to develop SURF User Interface within CADE

V&V Comments Have Significantly Improved SRDR Data Quality
### V&V Questions Most Frequently With “No” Response

**All Reviews (Mar - Dec 2016)**

<table>
<thead>
<tr>
<th>Question ID</th>
<th>Question from V&amp;V Guide Template</th>
<th>N(No)</th>
<th>N(Yes)</th>
<th>N(N/A)</th>
<th>N(No Resp.)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.5.1.1</td>
<td>Does the submission clearly illustrate the number of Inherited, Added, Modified, Deleted, and Deferred requirements for both internal and external categories?</td>
<td>31</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1.5.2.16</td>
<td>If COTS or GOTS items have been included within the submission, has the submitting organization provided the SLOC total required to integrate the identified COTS/GOTS product (i.e. Glue code)?</td>
<td>28</td>
<td>0</td>
<td>4</td>
<td>0</td>
</tr>
<tr>
<td>1.5.1.2</td>
<td>Has the submitting organization separated the provided requirements by Security, Safety, and Privacy or Cybersecurity?</td>
<td>26</td>
<td>1</td>
<td>5</td>
<td>0</td>
</tr>
<tr>
<td>1.5.2.4</td>
<td>Did the submitter us the Aerospace-approved version of the University of Southern California (USC) Center for Systems and Software Engineering (CSSE) Unified Code Count (UCC) tool to count the provided SLOC totals? If not, was the name of the code counting tool used by the submitting organization included within the supporting comments section and/or data dictionary?</td>
<td>25</td>
<td>6</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1.2.5</td>
<td>Is the system description been included within the submission?</td>
<td>24</td>
<td>8</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1.2.2</td>
<td>Has the Major Defense Acquisition Program (MDAP) or Major Automated Information System (MAIS) designation been listed?</td>
<td>20</td>
<td>2</td>
<td>10</td>
<td>0</td>
</tr>
<tr>
<td>1.3.2.3</td>
<td>Has the state of development been identified (For example: Prototype, Production Ready, or a mix of the two)?</td>
<td>19</td>
<td>11</td>
<td>2</td>
<td>0</td>
</tr>
<tr>
<td>1.5.4.2</td>
<td>Has the priority level for each category of software defects been provided?</td>
<td>18</td>
<td>9</td>
<td>5</td>
<td>0</td>
</tr>
<tr>
<td>1.1.9</td>
<td>Is it clear if the information represents a Technology Demonstration (TD) or Engineering and Manufacturing Development (EMD) phase if the program is in that stage of development?</td>
<td>17</td>
<td>8</td>
<td>7</td>
<td>0</td>
</tr>
<tr>
<td>1.5.2.13</td>
<td>Has the contractor or submitting organization provided the name of the software products that have been referenced to generate the provided reuse SLOC totals?</td>
<td>17</td>
<td>14</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1.5.4.1</td>
<td>Has the submitting organization provided a breakout of the number of software defects Discovered, Removed, and Deferred?</td>
<td>17</td>
<td>10</td>
<td>5</td>
<td>0</td>
</tr>
<tr>
<td>1.7.2</td>
<td>Has the submitting organization clearly stated if the provided schedule data was reported as estimated, allocated, or actual results?</td>
<td>16</td>
<td>16</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1.2.16</td>
<td>Is the specific U.S. Military service branch or customer identified (For example: Navy, Air Force, Army, prime contractor, etc.)?</td>
<td>15</td>
<td>14</td>
<td>3</td>
<td>0</td>
</tr>
<tr>
<td>1.3.1.1</td>
<td>Does the SRDR submission, comments section, or data dictionary include a clear system level functional description and software operational overview?</td>
<td>15</td>
<td>17</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1.2.6</td>
<td>Have the program phase and/or milestone been included within the report (for example: Pre-A, A, B, C-LRIP, C-FRP, O&amp;S, etc.)?</td>
<td>14</td>
<td>18</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1.3.2.1</td>
<td>Does the SRDR data dictionary include a clear system-level functional description and software operational overview?</td>
<td>14</td>
<td>17</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1.2.19</td>
<td>Has the contract Period of Performance (PoP) been identified?</td>
<td>13</td>
<td>19</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1.2.23</td>
<td>Does the submission include adequate detail within the comments section to support analysts who may reference the submission sometime in the future (For example: Provide context for analyzing the provided data, such as any unusual circumstances that may have caused the data to diverge from historical norms)?</td>
<td>13</td>
<td>18</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1.3.3.3</td>
<td>If an upgrade, does the SW sizing reflect significant reuse or modification SLOC totals when compared to New code?</td>
<td>13</td>
<td>16</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>1.4.5</td>
<td>Does the peak headcount make sense against the reported schedule and hours? A simple test is to divide the total reported hours by the schedule months and then convert the resulting average monthly hours into an average Full Time Equivalent (FTE) count using the reported hours in a man-month. The peak headcount must be higher than this FTE monthly average. At the same time the peak headcount should not be wildly disproportional to that average either.</td>
<td>13</td>
<td>17</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1.5.2.10</td>
<td>Were code adaptation factors reported (percent redesign, recode, reintegration)? Do they appear to be unique for each CSCI, or are they standard rules of thumb?</td>
<td>13</td>
<td>4</td>
<td>15</td>
<td>0</td>
</tr>
<tr>
<td>1.2.17</td>
<td>Has the specific contract type been identified? For contracts, task orders, or delivery orders with multiple CLINs of varying contract types, the Contract Type reporting should be the one associated with the plurality of cost.</td>
<td>12</td>
<td>20</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1.2.22</td>
<td>Has the funding appropriation been identified (for examples: Research, Development, Test and Evaluation (RDT&amp;E), Procurement, Operation and Maintenance (O&amp;M), Foreign Military Sales (FMS), etc.)?</td>
<td>12</td>
<td>20</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1.2.4</td>
<td>Has the Defense material item category been provided in accordance with MIL-STD-881C guidance (for example: Aircraft, radar, ship, Unmanned Ariel Vehicle (UAV) system)?</td>
<td>12</td>
<td>12</td>
<td>8</td>
<td>0</td>
</tr>
<tr>
<td>1.3.3.5</td>
<td>Has the development method also been identified (for example: Structured Analysis, Object Oriented, Vienna Development, etc.)?</td>
<td>12</td>
<td>20</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1.6.2</td>
<td>Was effort data reported as estimated or actual results? If the submission includes estimated values and actual results, does the report include a clear and documented split between actual results and estimated values?</td>
<td>12</td>
<td>18</td>
<td>2</td>
<td>0</td>
</tr>
<tr>
<td>Question ID</td>
<td>Question from V&amp;V Guide Template</td>
<td>N(No)</td>
<td>N(Yes)</td>
<td>N(N/A)</td>
<td>N(No Resp.)</td>
</tr>
<tr>
<td>-------------</td>
<td>------------------------------------------------------------------------------------------------</td>
<td>-------</td>
<td>--------</td>
<td>--------</td>
<td>-------------</td>
</tr>
<tr>
<td>1.2.12</td>
<td>Is the contract number reported?</td>
<td>0</td>
<td>32</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1.2.20</td>
<td>Has the report type been identified (for example: Initial, Interim, or Final)?</td>
<td>0</td>
<td>32</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1.2.7</td>
<td>Has the contractor or organization that performed the work been identified?</td>
<td>0</td>
<td>32</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1.2.21</td>
<td>Is there a single submission Point of Contact (POC) and supporting contact information included within the report?</td>
<td>3</td>
<td>29</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1.7.1</td>
<td>Has schedule data been included in the submission?</td>
<td>3</td>
<td>29</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1.7.4</td>
<td>Is schedule data broken out by SRDR activity?</td>
<td>3</td>
<td>29</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1.1.2</td>
<td>Does the report reference the CSDR Plan?</td>
<td>4</td>
<td>28</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1.2.1</td>
<td>Has the program name been identified?</td>
<td>4</td>
<td>28</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1.5.2.1</td>
<td>Was the primary programming language reported?</td>
<td>4</td>
<td>28</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1.5.2.6</td>
<td>Are the SLOC counts for different types of code (e.g., new, modified, reused, auto-generated, Government-furnished, and deleted) separated or are they mixed together?</td>
<td>4</td>
<td>28</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1.6.3</td>
<td>Is the effort data reported in hours?</td>
<td>4</td>
<td>28</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1.6.4</td>
<td>Is effort data broken out by activity?</td>
<td>4</td>
<td>28</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1.6.5</td>
<td>Was the specific ISO 12207:2008 activities that are covered in the effort data (For example: Requirements analysis, architectural design, detailed design, construction, integration, qualification testing, and support processes) clearly discernable?</td>
<td>4</td>
<td>28</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1.6.6</td>
<td>Is there an easily identifiable event associated with the submission (for example: Contract Award, Build 2 Release, Build 1 Complete, Contract Complete, etc.)?</td>
<td>5</td>
<td>27</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1.6.1</td>
<td>Was effort data reported for each CSCI or WBS?</td>
<td>5</td>
<td>27</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1.1.2.14</td>
<td>Is the software process maturity and quality reporting definition provided (For example: Capability Maturity Model (CMM), Capability Maturity Model Integration (CMMI), or other alternative rating)?</td>
<td>4</td>
<td>27</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1.7.3</td>
<td>Has schedule data been reported in number of months from contract start or as calendar dates?</td>
<td>3</td>
<td>27</td>
<td>2</td>
<td>0</td>
</tr>
<tr>
<td>1.2.13</td>
<td>Are precedents reported and consistent from submission to submission?</td>
<td>1</td>
<td>27</td>
<td>4</td>
<td>0</td>
</tr>
<tr>
<td>1.3.3.4</td>
<td>What precedents or prior builds are identified to give credibility to the upgrade designation?</td>
<td>1</td>
<td>27</td>
<td>3</td>
<td>1</td>
</tr>
<tr>
<td>1.3.3.2</td>
<td>Has the contractor indicated whether the software is an upgrade or new development? If not, why not?</td>
<td>6</td>
<td>26</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1.4.3</td>
<td>Does the dictionary define what the skill level requirements are, and is the contractor adhering to that definition?</td>
<td>6</td>
<td>26</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1.2.15</td>
<td>Is the Process Maturity rating reported with an associated date, and has it changed from a prior submission?</td>
<td>3</td>
<td>26</td>
<td>3</td>
<td>0</td>
</tr>
<tr>
<td>1.2.10</td>
<td>Has the contractor or submitting organization illustrated whether they were the primary or secondary developer?</td>
<td>7</td>
<td>24</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1.6.6</td>
<td>Were common WBS elements/labor categories such as System Engineering (SE), Program Management (PM), Configuration Management (CM), or Quality Management (QM) been broken out separately?</td>
<td>7</td>
<td>24</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1.7.5</td>
<td>Does the report include unique schedule start and end date values? For example, do multiple records have the same schedule data, e.g., same calendar dates for multiple WBS/CSCIs or builds?</td>
<td>7</td>
<td>24</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1.2.3</td>
<td>Is the Prime Mission Product (PMP) name been clearly identified (for example: most current official military designation)?</td>
<td>5</td>
<td>24</td>
<td>3</td>
<td>0</td>
</tr>
</tbody>
</table>
## V&V Questions Most Frequently With “N/A” Response
### All Reviews (Mar - Dec 2016)

<table>
<thead>
<tr>
<th>Question ID</th>
<th>Question from V&amp;V Guide Template</th>
<th>N(No)</th>
<th>N(Yes)</th>
<th>N(N/A)</th>
<th>N(No Resp.)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.5.3.2</td>
<td>If function points have been provided has the submitting organization clearly illustrated the function point count type (For example: Enhancement Project, Application, or Development Project)?</td>
<td>0</td>
<td>0</td>
<td>32</td>
<td>0</td>
</tr>
<tr>
<td>1.5.3.3</td>
<td>Has the submitting organization provided the number of Data Functions and Transactional Functions (For example: Internal Logic Files, External Interface File, External Inquiries, External Inputs, and External Outputs)?</td>
<td>0</td>
<td>0</td>
<td>32</td>
<td>0</td>
</tr>
<tr>
<td>1.5.3.4</td>
<td>Has the submitting organization included the Value Adjustment Factor?</td>
<td>0</td>
<td>0</td>
<td>32</td>
<td>0</td>
</tr>
<tr>
<td>1.5.3.5</td>
<td>If the submitting organization has provided sizing metrics using the Reports, Interfaces, Conversions, Extensions, Forms, and Workflows (RICE-FW) convention, has the complexity of each RICE-FW category been provided?</td>
<td>0</td>
<td>0</td>
<td>32</td>
<td>0</td>
</tr>
<tr>
<td>1.5.4.3</td>
<td>FACH: Has a description been provided that describes which ISO 12207:2008 elements have been included within the provided total?</td>
<td>1</td>
<td>0</td>
<td>31</td>
<td>0</td>
</tr>
<tr>
<td>1.5.4.3</td>
<td>FACH: Do sub-element FAC values sum to the parent FAC total value?</td>
<td>1</td>
<td>0</td>
<td>31</td>
<td>0</td>
</tr>
<tr>
<td>1.5.4.3</td>
<td>If the report is a final report, does the provided ATD total match the provided FAC total?</td>
<td>1</td>
<td>0</td>
<td>31</td>
<td>0</td>
</tr>
<tr>
<td>1.5.4.3</td>
<td>Is the submission compliant with the CSDR Plan, i.e., a comparison of the submission to the plan requirement?</td>
<td>1</td>
<td>0</td>
<td>31</td>
<td>0</td>
</tr>
<tr>
<td>1.5.4.3</td>
<td>Were SLOC counts reported, or were other counting or sizing metrics used (e.g. function points, use cases, rung logic ladders, etc.)? If so, has the submitting organization obtained the appropriate authorization to report non-SLOC based sizing within the corresponding CSDR plan?</td>
<td>0</td>
<td>0</td>
<td>32</td>
<td>0</td>
</tr>
<tr>
<td>1.5.4.3</td>
<td>If subcontractor hours have not been provided, did the reporting organization provide subcontractor dollars?</td>
<td>2</td>
<td>1</td>
<td>29</td>
<td>0</td>
</tr>
<tr>
<td>1.5.4.3</td>
<td>If COTS or GOTS integration or glue code has been included within the submission, does the total seem realistic when compared to the total SLOC included in the CSCI or WBS element (For example: COTS integration code equals 500 KSLOC and the total SLOC for the specific CSCI or WBS element equals 150 KSLOC)?</td>
<td>3</td>
<td>0</td>
<td>28</td>
<td>1</td>
</tr>
<tr>
<td>1.5.4.3</td>
<td>Do the children or lower-level WBS/CSCI elements add up to the parent? If not, is there effort that is only captured at a higher-level WBS/CSCI level that should be allocated to the lower-level WBS/CSCI elements?</td>
<td>5</td>
<td>3</td>
<td>24</td>
<td>0</td>
</tr>
<tr>
<td>1.5.4.3</td>
<td>Has the total contract price been identified?</td>
<td>9</td>
<td>0</td>
<td>23</td>
<td>0</td>
</tr>
<tr>
<td>1.5.4.3</td>
<td>Do the number of requirements trace from the parent to the children in the WBS? If not, this could imply that some portion of the software effort is only captured at higher-level WBS/CSCI elements and should be cross checked.</td>
<td>3</td>
<td>4</td>
<td>23</td>
<td>2</td>
</tr>
<tr>
<td>1.5.4.3</td>
<td>For a Final report does the size look realistic? For example: is all of the code rounded to the nearest 1000 lines, or does the dictionary indicate that they had difficulty counting code that may have come from a subcontractor?</td>
<td>1</td>
<td>9</td>
<td>22</td>
<td>0</td>
</tr>
<tr>
<td>1.5.4.3</td>
<td>If there are prior submissions, is this submission an update to a prior submission or a new event? If the submission is an update to an existing submission, does the latest submission clearly describe what report the prior submission is linked to?</td>
<td>2</td>
<td>8</td>
<td>21</td>
<td>1</td>
</tr>
<tr>
<td>1.5.4.3</td>
<td>If effort was outsourced, has the outsourced organization been provided?</td>
<td>4</td>
<td>7</td>
<td>21</td>
<td>0</td>
</tr>
<tr>
<td>1.5.4.3</td>
<td>Is there an explanation of missing activities included within the supporting SRDR data dictionary?</td>
<td>7</td>
<td>4</td>
<td>21</td>
<td>0</td>
</tr>
<tr>
<td>1.5.4.3</td>
<td>If a prior submissions exists, is the information that has changed readily identifiable and a reason for the change provided (either in the data dictionary or comments section)?</td>
<td>5</td>
<td>6</td>
<td>20</td>
<td>1</td>
</tr>
<tr>
<td>1.5.4.3</td>
<td>Does the skill mix make sense relative to the complexity of the code (unusual amount of very low or very high skill mix, for example)?</td>
<td>0</td>
<td>12</td>
<td>20</td>
<td>0</td>
</tr>
<tr>
<td>1.5.4.3</td>
<td>If the report is an interim or final submission, has the number of Discovered, Removed, and Deferred defects changed from the previous submission? If significant changes have occurred, does the supporting comments section and/or data dictionary provide details regarding what drove the significant change in product quality metrics?</td>
<td>10</td>
<td>2</td>
<td>20</td>
<td>0</td>
</tr>
<tr>
<td>1.5.4.3</td>
<td>Does the submission include unique values for each of the lower-level CSCI or WBS elements? For example, do multiple related records have the same effort data (i.e. activity effort is repeated or total effort is repeated)?</td>
<td>6</td>
<td>6</td>
<td>20</td>
<td>0</td>
</tr>
<tr>
<td>1.5.4.3</td>
<td>If there was a prior submission, has the skill mix changed dramatically and, if so, is there an explanation why? Conversely, did it remain unchanged? If so, why?</td>
<td>8</td>
<td>4</td>
<td>19</td>
<td>1</td>
</tr>
<tr>
<td>1.5.4.3</td>
<td>When subcontractor code is present, is it segregated from the prime contractor effort, and does it meet the same criteria for quality as the prime’s code count?</td>
<td>6</td>
<td>7</td>
<td>19</td>
<td>0</td>
</tr>
</tbody>
</table>
• V&V comments are generated when SURF members answer a question with “No”

• Common trends for “No” responses:
  – Reports not including all types of requirement counts (e.g., new, modified, inherited, deleted, cybersecurity, etc.)
  – Reports not including COTS/GOTS “glue code” Software Lines of Code (SLOC) totals
  – Reports not including SLOC counts using USC Unified Code Count (UCC) tool
  – Reports not including software defect counts
  – Reports not including a subset of required metadata (e.g., TD, EMD, Prototype, Service, Milestone, Contract Type, etc.)

• Common trends for “Yes” responses:
  – Reports include a subset of metadata (e.g., contractor, contract number, report type, report POC, program name, etc.)
  – Reports typically have SLOC counts broken out by new, modified, reuse, auto, and deleted
  – Reports typically include primary language type designation
  – Reports typically include “Effort” broken out by activity

• Common trends for “N/A” responses:
  – Reports typically do not include “Forecast At Completion (FAC)” values
  – Reports typically do not include non-SLOC sizing metrics (Function Points, RICE-FW, etc.)
  – SURF analyst typically does not have access to corresponding CSDR plan (Working with CADE to develop SURF portal)
Currently, SURF members are updating or creating draft question lists to account for new DIDs for Development, Maintenance, and ERP
  Updates to the development question lists include improvements to the list from lessons learned over the previous year

Draft Question lists will then be sent out to a larger SRDR-focused team members to ensure questions list are reasonable and that they address quality data concerns
  Important to keep question lists to a reasonable size for continued SURF success

V&V guide and question templates to be updated to incorporate new questions as well as other lessons learned

Updated V&V to larger SRDR working group and senior management for final comments/feedback

Send Updated V&V guide to OSD for final PAO approval and posting to CADE
SURF Summary

- SURF is focused on improving data quality and helping support robust Government review process
- We would like to thank all of the DoD and Non-DoD individuals who have commented, participated, and provided feedback throughout the past few years
- Please feel free to use the contact information below if you would like more information regarding SURF, the SRDR V&V Guide, or checklist

Marc Russo
Naval Center for Cost Analysis (NCCA)
NIPR: Marc.russo1@navy.mil

Nicholas Lanham
Naval Center for Cost Analysis (NCCA)
NIPR: Nicholas.lanham@navy.mil

Ron Cipressi
Air Force Cost Analysis Agency
NIPR: Ronald.p.cipressi.civ@mail.mil

Dan Strickland
Missile Defense Agency (MDA)
NIPR: Daniel.strickland@mda.mil
Expanding Our Estimation Tool Set: Formalizing analogy based cost estimation

Jairus Hihn
Jet Propulsion Laboratory, California Institute of Technology

August 22-24, 2017
Software and IT-CAST
First Things First - The Team

- JPL/California Institute of Technology
  Dr. Jairus Hihn
  Elinor Huntington
  Alex Lumnah
  Michael Saing
  Tom Youmans
- NASA Strategic Investment Division
  James Johnson
- North Carolina State University (Original Research Team)
  Dr. Tim Menzies
  George Mathew

Contact:
Dr. Jairus Hihn, jairus.m.hihn@jpl.nasa.gov
James K. Johnson, james.k.johnson@nasa.gov
Acknowledgements

• Many individuals have contributed or assisted with this work:
  – JHU APL: Nicole Powers-Krepps, Sally Whitley, Meagan Hahn, Christian Patton
  – NASA GRC: Elizabeth (Betsy) Turnbull, Chris Blake, Tom Parkey, Bob Sefcik
  – NASA HQ: Cris Guidi, Charley Hunt, Doug Comstock, Eric Plumer
  – NASA GSFC: Stephen Shinn, Tamra Goldstein
  – NASA ARC: Tommy Paine
  – Special thanks to Julie McAffee and Mike Blandford of ONCE team
Isn’t software maintenance free? It was free at the university research programs!

- Program Office Manager

But we are just cloning the last mission so flight software budget is basically ZERO, right! (Oh and all the instruments/sensors have been changed)

- A Different Program Office Manager

My project is special and I do not need to follow the standard WBS. By the way can we use Mission X data to help us cost my mission.

- Project Manager
Why explore alternative modeling methods?

- For most of our history the cost community has relied upon regression based modeling methods
- Sometimes regression breaks down
  - Regression methods have the underlying assumption of clean and complete data with large sample sizes
- Guess what - Most cost data suffers from sparseness, noise, and small sample sizes
- The point is we need more tools in our toolkit
Example of Classic Breakdown with Regression
Anscombe’s Quartet
Anscombe’s Quartet

All four of the displayed plots have virtually identical statistics
- Means, Medians, Variances
- Regression line, $R^2$, F and T tests
- But visual inspection clearly shows they are very different

MRE can distinguish between the models

- Plotting the absolute values of the relative error it is easily seen that Model 3 fits its data best just as intuition would indicate
  - MRE = Magnitude of Relative Error, \( \frac{\text{abs(Predicted} - \text{Actual})}{\text{Actual}} \)
Formal Analogy and Bayesian Models are a Natural Next Step in the Evolution Cost Modeling and Analysis.
What We Learned from Methodology

- There are a variety of models whose performance are hard to distinguish (given currently available data) but some models are better than others.
- If one has sufficient data to run a parametric model such as COCOMO then the best model has repeatedly been found to be the parametric model.
- When insufficient information exists then a model using only system parameters can be used to estimate software costs with ‘acceptable’ reduction in accuracy. The main weakness is the possibility of occasional very large estimation errors which the parametric model does not exhibit.
- A major strength of the nearest neighbor and clustering methods is the ability to work with a combination of symbolic and numerical data.
- While a nearest neighbor model performs as well or better as clustering based on MMRE, clustering handles outliers better and provides a structured model that supports cost analysis and not just prediction.
Comparing Model Performance

- To compare models we use MRE metrics from leave one out validation
- COCOMO II out of the box performs well against parametric and non-parametric models
- Even performs well against local calibration
- If you have enough information run a parametric model!!

<table>
<thead>
<tr>
<th>Estimation Model</th>
<th>Median MRE (MMRE)</th>
<th>25&lt;sup&gt;th&lt;/sup&gt; Percentile</th>
<th>75&lt;sup&gt;th&lt;/sup&gt; Percentile</th>
</tr>
</thead>
<tbody>
<tr>
<td>Knn1 (Nearest Neighbor)</td>
<td>32%</td>
<td>14%</td>
<td>80%</td>
</tr>
<tr>
<td>PEEKING2 (Spectral Clustering)</td>
<td>32%</td>
<td>16%</td>
<td>97%</td>
</tr>
<tr>
<td>COCOMO2</td>
<td>36%</td>
<td>22%</td>
<td>55%</td>
</tr>
<tr>
<td>Mission Type Summary Table</td>
<td>38%</td>
<td>14%</td>
<td>106%</td>
</tr>
<tr>
<td>COCONUT</td>
<td>44%</td>
<td>32%</td>
<td>62%</td>
</tr>
</tbody>
</table>

Negative results for software effort Estimation, Empirical Software Engineering, Nov 2016
Menzies, Yang, Mathew, Boehm, Hihn
ASCoT is the NASA Analogy Software Cost Tool

The purpose of ASCoT is to

- Supplement current estimation capabilities
- Be effective in the very early lifecycle when our knowledge is fuzzy
  - uses high level systems information
  - Usable by Cost Estimators, Software Engineers and Systems Engineers

Methodology handles

- small sample sizes
- noisy and sparse data
- Can also handle large data sets

Previous research approach and activities are widely published

- ICEAA 2014, 2015
- Numerous research publications in IEEE SW, TSE, ASE, Empirical Software Engineering by Professor Tim Menzies et.al.
We Are Estimating With minimum Inputs

Cluster and KNN algorithms use

- Spacecraft Type
- Destination
- Number of Instruments
- Number of Deployables
- Software Inheritance Categories
- Mission Size ($) Categories

Regression Model uses

- Spacecraft Development Costs
- Number of Instruments
Rm: HST (Hubble, 1830), Near (48)

Large Outer Planets

Rovers

This!
Model MRE Performance

Model Estimation Error, based on MRE, is steadily improving

MRE Comparison Based on Test Cases

<table>
<thead>
<tr>
<th>Test Case MRE</th>
<th>ASCoT Prototype</th>
<th>ASCoT Beta</th>
<th>ASCoT</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0%</td>
<td>1%</td>
<td>2%</td>
</tr>
<tr>
<td>1</td>
<td>1%</td>
<td>3%</td>
<td>3%</td>
</tr>
<tr>
<td>2</td>
<td>3%</td>
<td>3%</td>
<td>7%</td>
</tr>
<tr>
<td>3</td>
<td>4%</td>
<td>22%</td>
<td>15%</td>
</tr>
<tr>
<td>4</td>
<td>35%</td>
<td>23%</td>
<td>27%</td>
</tr>
<tr>
<td>5</td>
<td>45%</td>
<td>29%</td>
<td>32%</td>
</tr>
<tr>
<td>6</td>
<td>4%</td>
<td>35%</td>
<td>35%</td>
</tr>
<tr>
<td>7</td>
<td>79%</td>
<td>37%</td>
<td>37%</td>
</tr>
<tr>
<td>8</td>
<td>101%</td>
<td>37%</td>
<td>37%</td>
</tr>
<tr>
<td>9</td>
<td>102%</td>
<td>51%</td>
<td>51%</td>
</tr>
<tr>
<td>10</td>
<td>192%</td>
<td>54%</td>
<td>54%</td>
</tr>
<tr>
<td>11</td>
<td>506%</td>
<td>175%</td>
<td>107%</td>
</tr>
</tbody>
</table>

Median MRE: 40% | 26% | 30%
Average MRE: 89% | 37% | 32%
By gradually increasing the granularity of our clusters, while maintaining robustness to avoid overfitting, we were able to find logical separation between groupings of missions.
Cluster Parameter Variation
Conclusion: Put It In A Tool

NASA
Analogy Software Estimation Tool
“ASCoT” Key Analysis Components

Cluster Analysis
- Clustering
- Development Effort Estimate

Knn Analysis
- Nearest Neighbor
- Development Effort and SLOC Estimate

Regression Analysis
- Linear Regression
- Development Cost Estimate

COCOMO II
- Verified Reproduction
- Cost/Effort

- Cluster & Regression Analysis components listed rely on high level Mission Descriptors such as # of Instruments and Mission Type

- COCOMO II is a reproduction and uses traditional inputs
Data Sources

• Where the data came from
  – NASA CADRe (When it exists and is usable)
    • Cost Analysis Data Requirements archived in ONCE database
  – Contributed Center level data
  – NASA software inventory
  – Project websites and other sources for system level information if not available in CADRe
Why Does Software Cost So Much?
Toward a Causal Model

23 August 2017

Mike Konrad
Robert Stoddard
David Zubrow

Software Engineering Institute
Carnegie Mellon University
Pittsburgh, PA 15213
Outline

- What is causal learning and modeling, and why do we care about it?
- Our technical approach
- Initial Results
- Conclusions
Our Project: Bottom Line Up Front

Goal
- Demonstrate the benefit of causal modeling to the software cost domain
- Identify and quantify a causal network of factors that drive software effort and schedule

Actionable intelligence
- Enhance program control of software cost throughout the development and sustainment lifecycles
- Inform “could/should cost” analysis and price negotiations
- Improve contract incentives for software intensive programs
- Increase competition using effective criteria related to software cost

If you are interested in this approach, let’s work together.
Why do we care about causal modeling?

Proactively controlling software costs requires knowing which of our “independent factors” actually cause outcomes to change in a predictable manner.

Just as correlation may be fooled by spurious association, so can regression.

We must move beyond correlation to causation, if we want to make use of cause and effect relationships.

Today, we can garner evidence of causation without the expense and challenge of conducting a controlled experiment.

Establishing causation with observational data remains a vital need and a key technical challenge, but is becoming more feasible and practical.
Significant Progress Toward Practicality

Sewall Wright Path Models (1920’s)
Structural Equation Models (1930’s)
  Social Science Path Models (1960’s)
  Bayesian Networks (1980’s)

Pearl’s Probabilistic Reasoning (1988)

Pearl’s 2nd Edition Book on Causality (2009)
Morgan Counterfactuals & Causality (2007)
Morgan Counterfactuals & Causality (2014)
Morgan Handbook Social Science Causal Inference (2014)

TETRAD – An Open Source Tool for Causal Learning
Carnegie Mellon University
http://www.phil.cmu.edu/tetrad/
University of Pittsburgh
http://www.ccd.pitt.edu/

For video tutorials from 2016 summer short course:
http://www.ccd.pitt.edu/training/presentation-videos/
Basic Technical Approach

Causal Discovery
using Tetrad, which implements a variety of algorithms

Causal Model (DAG)

Formulate Hypotheses
using domain knowledge and prior scholar publication

Estimated Model (SEM)
Integrating Models

Tetrad Learning

- COCOMO Data
- Vendor 1 Data
- Vendor 2 Data
- Vendor 3 Data
- SRDR Data
- TSP/PSP Data
- CSIAC Data

~ 60 unique cost factors
15+ cost relationships to evaluate

Actionable Causal Models

Module Effort = f(factor1, factor2, factor3)
Module Post-Development Quality = g(factor1, factor4, factor5)
High-Reliability Module Cost = h(factor4, factor6, factor7)
A familiar example of a causal model

Hard to find data sources to actually estimate the entire model

Consequently harder to empirically establish the causal relationships

Causal modeling methods allow for the integration of partial models

Opportunity for empirical support and refinement
Explaining Final Effort and Duration (Initial Results)

181 pairs of matched initial-final SRDR reports reduced to 134 (complete Req...INT data).
Explaining Final Effort and Duration
Both this chart and previous analyzed with PC algorithm with Alpha set to .001.
What Do These Initial Results Suggest?

Effort estimates for Req, Arch, Code, INT directly influence effort actuals.

- Not so for Duration

There are other cases where estimates of an attribute do not directly influence actuals for that attribute, suggesting challenges to estimation.

Total effort actual

- may be directly influenced by Req effort and Code effort actuals
- not directly influenced by Arch effort actual
- directly influences INT effort actual (after accounting for influence of initial INT effort estimate). Evidence of effort compression?

Cautions

- Double-headed edges suggest unmeasured confounders (factors that are a common cause of factors connected by the edge).
- Undirected edges suggest insufficient data.
Explaining Size and Defect Density – Need to Drill Deeper?
Data from 975 programmers during PSP training

**Historical Programmer Capabilities**

**Effort**
- E:Plan
- E:Design
- E:Code
- E:Compile
- E:Test
- E:PM

**Defect Injection**
- I:Plan
- I:Design
- I:DLDI
- I:Code
- I:CR
- I:Compile
- I:Test

**Defect Removal**
- R:Plan
- R:Design
- R:DLDI
- R:Code
- R:CR
- R:Compile
- R:Test

**Outcomes**
- LOC
- DD-Test
Conclusions

Causal learning:
• has come of age from both a theoretical and practical tooling standpoint
• may be performed on data whether it be derived from experimentation or passive observation

Causal models:
• help separate true causes from spuriously-correlated factors
• help identify when unknown causes may likely exist
• lend themselves to actionable intelligence better than models based on correlation

We welcome collaborators interested in using these methods and tools.
QUESTIONS?
Contact Information

Points of Contact

Robert Stoddard
rws@sei.cmu.edu

Mike Konrad
mdk@sei.cmu.edu

Dave Zubrow
dz@sei.cmu.edu

William Nichols
wrn@sei.cmu.edu

David Danks
ddanks@cmu.edu

Kun Zhang
kunz1@cmu.edu

U.S. Mail
Software Engineering Institute
Customer Relations
4500 Fifth Avenue
Pittsburgh, PA 15213-2612, USA

Web
www.sei.cmu.edu
www.sei.cmu.edu/contact.cfm

Customer Relations
Email: info@sei.cmu.edu
Telephone: +1 412-268-5800
SEI Phone: +1 412-268-5800
SEI Fax: +1 412-268-6257
Reliable Non-Design, Code, Test, and Integration Cost Relationships

Jeremy Goucher  |  Brittany Staley

August 2017
Agenda

• Introduction
• Data
• Generalized Methodology
• Program Management
• System Engineering
• Integrated Logistics
• System Integration
• Modeling and Simulation
• Training and Test Sites
• NDCTI vs DCTI Phasing
• Conclusion
Introduction

• Non-Design, Code, Test, and Integration (NDCTI) costs can make up more than 50% of the total cost estimate.

• NDCTI elements are typically estimating using cost relationships (CRs) derived by parametric methods as functions of DCTI cost:
  – Increase in DCTI implies larger team, increased complexity, increased funding, increased contracting actions, all translating to increases in many NDCTI elements.

• New CRs are needed on a routine basis to ensure models are consistent with current trends.

• NDCTI costs are grouped into six major categories:
  - Project Management (PM)
  - System Integration (SI)
  - System Engineering (SE)
  - Modeling and Simulation (MS)
  - Integrated Logistics Support (IL)
  - Training and Test Sites (Sites)
Data

- 12 years of data ending in 2016
- All data normalized to BY17$s
- 3 completed projects, 1 project 75% complete
  - All results based on 3 completed projects unless otherwise noted
  - Fourth project presented anecdotally
- Software sizes range from 200k to 1.4M equivalent source lines of code (ESLOC)
- Field testing, which can have a wide variety of requirements, are not included in analysis
  - Cost estimates for field tests are based on unique test requirements for each test event
Generalized Methodology

\[ \text{Factor} = \frac{NDCTI \$}{DCTI \$} \]

- Insufficient data points for regression analysis
  - 2 degrees of freedom
- All means, standard deviation, and coefficient of variation (CV) based on three completed programs
- Fourth program assumptions
  - 100% DCTI completed in first 75% of program schedule
  - 85% of PM, SE, ILS cost incurred in first 75% of program schedule
  - SI, MS, Sites ETC minimally analyzed
Program Management

- PM Includes
  - business and financial management
  - quality assurance standards and adherence
  - data and configuration management
  - program planning
  - program evaluation

\[ \mu = 15.31\% \]
\[ CV = 5.7\% \]

- Fourth project currently 13.14% of DCTI; program 75% complete
  - PM costs continuing to accrue; DCTI complete
  - Estimated PM CR at completion: 15.46%
Program Management Phasing

- Phasing Analysis
  - PM phasing shows long ramp up, some level of effort for a short duration (if at all), and steep drop at the end
  - Does not reflect markers of fixed, or level of effort, type cost

![PM Cost by Program](image-url)
System Engineering

• SE comprises engineering oversight and support functions including:
  − system level coordination
  − planning and integration
  − special projects

\[ \mu = 36.48\% \]
\[ CV = 8.0\% \]

• Fourth project currently 31.89% of DCTI; program 75% complete
  − SE costs continuing to accrue; DCTI complete
  − Estimated SE CR at completion: 37.52%
Phasing Analysis:
- Shows markers consistent with variable cost phasing
- Ramp up, peak, and ramp down more consistent with DCTI phasing
Integrated Logistics

• IL includes:
  − oversight and coordination of IL requirements and processes
  − management of supply chain and spares
  − development of technical manuals
  − training support

\[
\mu = 2.85\%
\]
\[
CV = 62.2\%
\]

• Fourth project currently 0.77% of DCTI; program 75% complete
  − IL costs continuing to accrue; DCTI complete
  − Estimated IL CR at completion: 0.91%

• Evaluation of CV
  − IL is very small portion of total cost
  − Assuming NDCTI represents 50% of total cost, ILS error likely represents error in estimate between 0.5% and 2%
Integrated Logistics Phasing

- **Phasing Analysis:**
  - Variable costs with no common spend pattern
  - Possibly includes “on-demand” or schedule based services or products

### IL Cost by Program

![IL Cost by Program Chart](chart.png)

- **P1**
- **P2**
- **P3**
- **P4 (75%)**

**Normalized Cost**

**Time**

Years: Y0, Y1, Y2, Y3, Y4, Y5, Y6, Y7, Y8, Y9

Normalized Cost:
- Y0: $0
- Y1: $1
- Y2: $2
- Y3: $3
- Y4: $4
- Y5: $3
- Y6: $2
- Y7: $1
- Y8: $0
- Y9: $0
System Integration

• SI includes:
  – System level requirements
  – multi-element integration and test
  – test plans and procedures
  – integration oversight

\[
\mu = 20.91\%
\]
\[
CV = 91.9\%
\]

• Evaluation of CV:
  – DCTI cost is not a very good basis of estimate for SI cost
  – High productivity reduces DCTI cost, but likely has no impact on the effort to integrate the various elements into a single program
• SI measured as a function of ESLOC
  – Removes DCTI productivity from the equation
  – Large ESLOC may relate to large integration efforts

\[
SI = f(ESLOC)
\]

• Findings
  – To avoid potential disclosure of proprietary information, results of SI as a function of ESLOC cannot be shown
  – Two of three completed programs have very similar $ / ESLOC cost
  – Incomplete program on track to be similar to the two programs with similar $ / ESLOC ratios

• Other considerations:
  – Possible SI could be analyzed in groups based on similar technical specifications
System Integration Phasing

- Phasing Analysis
  - SI phasing displays ramp up/ramp down with peaks and valleys
  - More cost in the second half than the first half

![SI Cost by Program](image)
• M&S comprises the effort to develop simulated environments within which a computer program can be tested

\[ \mu = 11.83\% \]
\[ CV = 59.3\% \]

• Fourth project currently **11.08%** of DCTI; program 75% complete

• Evaluation of CV
  - DCTI may not be a good BOE for MS
  - MS effort involves developing a synthetic environment within which the primary program can be operated and tested
  - Likely requires ESLOC inputs and unique DCTI type calculations
Modeling & Simulation Phasing

- Phasing Analysis
  - The effort to development simulated environments within which a computer program can be tested
  - No obvious common pattern
  - May require unique phasing based on program requirements

![M&S Cost by Program Graph](image)
Sites comprises the effort to integrate, install, and test the designed system at both training and test site facilities.

\[ \mu = 6.32\% \]
\[ CV = 19.49\% \]

Fourth project currently **1.29%** of DCTI; program 75% complete
- Sites costs typically incurred near the end of the program
- Expect fourth project final Sites cost to be in line with completed projects
• Phasing Analysis:
  – Sites cost phasing shows peaks and valleys
  – Schedule based phasing best approach
NDCTI vs DCTI Phasing

- More likely variable and/or schedule based phasing than fixed, or level of effort, type phasing
- As a composite, NDCTI cost tracks closely to DCTI cost
- Variable cost phasing may be due to corporate strategy to develop functional teams
Conclusion

- Five of six NDCTI CRs recommended for general use
- SI requires additional analysis
- Cost phasing shows variable patterns rather than fixed, or level of effort, type phasing
- Larger data set would likely improve results

<table>
<thead>
<tr>
<th></th>
<th>Mean</th>
<th>St. Dev.</th>
<th>CV</th>
</tr>
</thead>
<tbody>
<tr>
<td>Program Management</td>
<td>15.31%</td>
<td>0.87%</td>
<td>5.7%</td>
</tr>
<tr>
<td>System Engineering</td>
<td>36.48%</td>
<td>2.91%</td>
<td>8.0%</td>
</tr>
<tr>
<td>Integrated Logistics</td>
<td>2.85%</td>
<td>1.77%</td>
<td>62.2%</td>
</tr>
<tr>
<td>Modeling and Simulation</td>
<td>11.83%</td>
<td>7.02%</td>
<td>59.3%</td>
</tr>
<tr>
<td>System Integration*</td>
<td>20.91%</td>
<td>19.21%</td>
<td>91.9%</td>
</tr>
<tr>
<td>Training and Test Sites</td>
<td>6.32%</td>
<td>1.23%</td>
<td>19.4%</td>
</tr>
</tbody>
</table>

*Not a recommended cost relationship*
Stay Connected
linkedin.com/company/herren-associates-inc

Authors

Jeremy Goucher
Phone: (202) 802 5683
jeremy.goucher@jlha.com

Brittany Staley
Phone: (831) 236 1875
brittany.staley@jlha.com

About Herren

Founded in 1989, Herren Associates is an engineering and management consulting firm with a proven record of maximizing the value of every taxpayer dollar. As trusted advisors to federal executives, we partner with clients to drive operational improvements and manage performance - maximizing efficiency and cost effectiveness.
Introduction to Software Obsolescence Cost Analysis Framework

Sanathanan Rajagopal
Principal Cost Consultant

23 August 2017
Software and IT-CAST Meeting
QinetiQ Businesses

Air and Space

Maritime, land and weapons

North America

Cyber, Information & training

OptaSense®

International
QinetiQ, International Business, Advisory Services

- Over 150 highly skilled and experienced subject matter experts
- 70%+ Professional Accreditation
- ACostE, SCAF, APM, Prince2, MSP, ICEAA
- Based across 5 UK Sites
- Deployed internationally
- Average experience of 10 years
- Over 40% PhD / MSc qualified
Agenda

1. Research Aim
2. Definitions
3. Introductions
4. Cognitive Case Studies
5. Software Obsolescence Cost Analysis Framework
6. Software Obsolescence and Software Maintenance
7. Summary
8. Conclusion
Software Obsolescence

Research Aim
“To develop a cost analysis framework to estimate the cost of Software Obsolescence Resolution of a bespoke real-time software in defence and aerospace”
Software Obsolescence

Definitions
Software Obsolescence is defined as “what happens when the original and authorised third party ceases to provide support with regular update, upgrade, fixes or due to the changes in target or operating environment, systems or hardware which makes the software unusable”

-S Rajagopal et al; (2014)
Software Maintenance

Software maintenance is the process of managing software regularly by patching, bug fixing, updates and undertaking major upgrade during the productive lifecycle.
Other Definitions

Mitigation Strategy

- Action performed in order to reduce the risk or the potential impact of obsolescence issues.

Resolution Approach

- Action carried out once obsolescence issues arises and needs to be addressed.
## Software Obsolescence vs Software Maintenance

<table>
<thead>
<tr>
<th>Software Maintenance</th>
<th>Software Obsolescence</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bug fixes</td>
<td>Replacement of entire application if need be to a new one</td>
</tr>
<tr>
<td>To address fault/Failures, security patches etc.</td>
<td>To address the issues with the application in totality</td>
</tr>
<tr>
<td>Maintenance is the review of the stored files to ensure they are still useable</td>
<td>Solves unavailability of fixes, licenses, permission and upgrades</td>
</tr>
<tr>
<td>Software maintenance takes care of the current versions to ensure that its up and running and meeting the requirements</td>
<td>Software Obsolescence management looks forward the industry standards and other software to continue supportability of the software</td>
</tr>
<tr>
<td>Maintenance deals with the upgrading the software to enhance capability</td>
<td>Obsolescence management deals with enforced changes in the environment</td>
</tr>
</tbody>
</table>
Software Obsolescence

Introduction
Introduction

The need for a Software Obsolescence Cost Analysis Framework

- Long Support Contract
- High Dependency on Software in Defence and Aerospace
- Software is a key (Cost and Schedule) Programme Driver
- Software Obsolescence is unavoidable
- Constant changes in Hardware (target and Operating Environment)

Need for a Software Development Cost Analysis Framework
Following process was undertaken to develop the Framework:

1. Case Studies
2. Online Survey
3. Cognitive Case Studies
4. Literature Search
5. SME Interviews

Latest Development
Software Obsolescence

Cognitive Case Study
Cognitive Case Study- Aim

The aim of the cognitive case study is to “identify how software developers select technologies to mitigate the effect of software obsolescence during software development, which could then be used to inform the required resolution strategies”
Cognitive Case Study – Methodology

• This study methodology employs the “think aloud” technique to capture the cognitive actions of software developers.

• This requires the participant to literally say aloud everything that they thinks or does during a controlled experiment.

• Everything that is said will be recorded (video and audio), transcribed and then described as a “verbal protocol”.

• Anything that is written down by the participant during the experiment is collected and analysed as a “written protocol”.
Demography of participants
Name of the companies cannot be disclosed due to confidentiality agreement

<table>
<thead>
<tr>
<th>Participants</th>
<th>Company</th>
<th>Level of Experience</th>
<th>Years of Experience</th>
</tr>
</thead>
<tbody>
<tr>
<td>Participant 1</td>
<td>Company A</td>
<td>Novice</td>
<td>2</td>
</tr>
<tr>
<td>Participant 2</td>
<td>Company B</td>
<td>Expert</td>
<td>15</td>
</tr>
<tr>
<td>Participant 3</td>
<td>Company C</td>
<td>Expert</td>
<td>20</td>
</tr>
<tr>
<td>Participant 4</td>
<td>Company B</td>
<td>Practitioner</td>
<td>8</td>
</tr>
<tr>
<td>Participant 5</td>
<td>Company D</td>
<td>Practitioner</td>
<td>5</td>
</tr>
</tbody>
</table>
Cognitive Case Study - Analysis

- Analysing Software Obsolescence problem
- Consulting information on Software
- Evaluating Software Obsolescence problem

PERCENTAGE OF TIME

MICRO STRATEGIES

- Top Down Selection of Technology
- Confirmation of the Technology
- Re-evaluating the technology selections

MACRO STRATEGY

- Explaining effect of technology selection

PERCENTAGE OF THE TIME

MICRO STRATEGIES

- Referring to technology knowledge
- Referring to technology selection
- Referring to an software obsolescence strategy
- Explaining effect of technology selection

Unclassified - QinetiQ Proprietary
Participant 1  “I will use this technology so that it is readily available for the maintainers to use it in future”

Participant 2  “I am using this approach (Technology selection) in order to reduce the obsolescence as this technology is independent of changes in hardware”

Participant 3  “I am using this technology because I am certain that in next 10 years there will not be a change in the hardware or system in Mod that will make this technology selection an obsolete one”
Process undertaken by Participants during Cognitive Experiment

Cognitive Case Study Model

Micro Strategy

Start → Analyse the Software Obsolescence problem → Consulting Information on Software Obsolescence Problem → Technology Selection based on the requirement → Reviewing the solution (selection of technology and its implications)

Macro Strategy

Deployment of Solution → Testing the solutions for obsolescence

Problem Areas

Deployment of Solution → End
Cognitive Case Study- Findings

• There seems to be a link between the technical selection and the resolution approach.

• Requirement should be stable to reduce the effect of Software Obsolescence.

• Stable operating and target environment reduces the risk of Software Obsolescence.

• Establishment of good support system reduces the Software Obsolescence risks.

• Maintaining the software in house and building an in-house capability will reduce the risk of software obsolescence, however this would be expensive.
Software Obsolescence Cost Analysis Framework
Software Obsolescence Cost Analysis Framework

The framework has the following attributes

– This framework is in its third iterations.
– This framework’s foundation is based on the Literature Searches, Case Studies, Online Survey results, SME Interviews and Cognitive Case Studies.
– This framework has several attributes that can be mapped across from and to, to the software estimating principals.
– This framework looks at the Cost Risk and Uncertainty which is at its development stage.
Software Obsolescence Cost Analysis Framework

- Project and System Information
  - Project Data
  - Project Schedule Data
  - Software Application Data
  - Software Sizing Matrix
  - System Details
  - Software Development Data

- Historic Data (Maintenance Data)
  - ISBSG Data

- Obsolescence Management Level
  - 1.As Is-State
  - 2.Resolution Strategy

- Resolution Approach
  - Identification of Resolution Techniques

- Software Obsolescence Complexity Level
- Software Obsolescence Resolution Profile
- Software Obsolescence Key Cost Drivers
- Risk and Uncertainty Modelling

- Software Obsolescence Management Systems

- Software Obsolescence Resolution Cost Modelling

- Cost of Software Obsolescence Resolutions

- CER based on Historic Data (ISBSG)

- Representation
- Approach

- Estimates
Software Obsolescence Management Level

• Software Obsolescence Management Level determines the level the current software program/Project/team is at in managing software obsolescence.

• It provides information on the as-is state for managing Software Obsolescence

• From this process, the following should be determined
  – Obsolescence Management Strategy
  – Software Obsolescence Management Strategy
  – Project teams approach towards software obsolescence risks
  – Capacity and capability to deal with software Obsolescence
  – Ability to deploy software obsolescence monitoring systems/tools if any
  – Understanding obsolescence resolution strategy
  – Capacity and capability to monitor software supply chain
Software Obsolescence Management Level

1. Deal with S/W Obsolescence Reactively
2. No Obsolescence Management Strategy
3. Freeze and do nothing
4. CMMI Level 1
5. Low TOMCAT Score

1. No S/W Obsolescence Management Strategy
2. Reactive but dealing with software obsolescence by reverse engineering and code conversions
3. CMMI Level 2
4. Low TOMCAT Score

1. Deploy Software Obsolescence Monitoring process or tool, if available
2. Monitoring Software Supply Chain
3. Monitoring Skills and Technology Insertions
4. Deploy Software Obsolescence Professional
5. Monitor Software Obsolescence Proactively
6. CMMI Level 3
7. Medium TOMCAT Score

1. Deploy S/W Obsolescence Mgt Strategy
2. Proactive Mgt of Obsolescence and effective Mgt of S/W Obsolescence Management Strategy
3. Continuous Monitoring of S/W Obsolescence
4. Management of Software Obsolescence is Business as Usual
5. Considering Software Obsolescence at the design and development stages.
Software Obsolescence Management Level

• Software Obsolescence Management Level will help the Organization/Project Team understand what they need to do get to an optimum level of Software Obsolescence Management.

• The organisations /Project team at the lowest level in Software Obsolescence Management level deals with software obsolescence reactively and the organisations/project team at the highest level deals with the software obsolescence proactively.

• However the organisations/project team at higher level may have a higher overhead which is suitable for large organisations but very expensive to small to medium size organizations.

• Due to these reason organisations should undertake a sequential trade off to get optimum benefits out of the software Obsolescence management level.
Software Obsolescence Resolution Approach

• Software Obsolescence Resolution Approach helps to tease out the resolution strategy.

• The resolution approach will help in identification of the best possible resolution techniques.
  – Resolution techniques are determined based on the Project and Systems Parameters and Software Management Level
  – Resolution techniques are identified for individual software component rather than software program as a whole.

• Software Obsolescence Resolution Approach helps the project team to compile an appropriate software obsolescence strategy.
Software Obsolescence Resolution Approach

• Software Obsolescence Resolution Approach will help to identify the key cost drivers.

• Software Obsolescence Resolution Approach will help to identify the key risk/uncertainties around the selection of appropriate resolution approach.

• Three major types of resolution approach are identified
  – Technical Resolutions
  – Logistical Resolutions
  – Functional Resolutions
Software Obsolescence Resolution Approach (Adapted from Bartel et al)
Software Obsolescence Vs Software Maintenance
Breakdown of Technological Resolutions Approach

Technological Resolutions

Support

Technical

Maintenance

Purchase

Update

Upgrade

Bug/Patch fixes

First/Second/Third Line support

Perfective

Preventive

Corrective

Adaptive
### Software Maintenance – Definitions (Adapted from ISBSG)

<table>
<thead>
<tr>
<th>Perfective Maintenance</th>
<th>Preventative Maintenance</th>
<th>Corrective Maintenance</th>
<th>Adaptive Maintenance</th>
</tr>
</thead>
<tbody>
<tr>
<td>Perfective maintenance is the modification of a software application, after delivery, to improve performance or maintainability</td>
<td>The modification of a software application after delivery to detect and correct latent faults in the software product before they become effective faults</td>
<td>The reactive modification of a software product performed after delivery to correct discovered problems.</td>
<td>Enhancements necessary to accommodate changes in the environment in which a software product must operate</td>
</tr>
</tbody>
</table>
Software Maintenance and Software Obsolescence Relationship

- Corrective Maintenance
- Adaptive Maintenance
- Perfective Maintenance
- Preventative Maintenance

Reactive Management

Proactive Management

High Risk

Low Risk
Case Study

Six case studies were undertaken to establish a relationship between software maintenance and software obsolescence. Below is the data point used for one of these case studies.

• Data from ISBSG

• Data points from
  – Financial Industry
  – Government
  – Electronics and Computers
  – Communications

• Number of applications: 201-500

• Y = No of Applications

• X1 = Total Maintenance Hours

• X2 = Perfective Maintenance Hours

• X3 = Adaptive Maintenance Hours
Software Maintenance and Software Obsolescence Relationship

<table>
<thead>
<tr>
<th>Observations</th>
<th>Name</th>
<th>Y</th>
<th>X1</th>
<th>X2</th>
<th>X3</th>
</tr>
</thead>
<tbody>
<tr>
<td>Include</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>y</td>
<td>299</td>
<td>230</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>y</td>
<td>299</td>
<td>4332</td>
<td>217</td>
<td></td>
<td>0</td>
</tr>
<tr>
<td>y</td>
<td>299</td>
<td>2733</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>y</td>
<td>299</td>
<td>1358</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>y</td>
<td>351</td>
<td>823</td>
<td>32</td>
<td></td>
<td>0</td>
</tr>
<tr>
<td>y</td>
<td>299</td>
<td>3941</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>y</td>
<td>416</td>
<td>174</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>y</td>
<td>351</td>
<td>238</td>
<td>11</td>
<td></td>
<td>0</td>
</tr>
<tr>
<td>y</td>
<td>351</td>
<td>888</td>
<td>65</td>
<td></td>
<td>0</td>
</tr>
<tr>
<td>y</td>
<td>351</td>
<td>266</td>
<td>79</td>
<td></td>
<td>0</td>
</tr>
<tr>
<td>y</td>
<td>299</td>
<td>634</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>y</td>
<td>299</td>
<td>908</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>y</td>
<td>299</td>
<td>251</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>y</td>
<td>416</td>
<td>3939</td>
<td>188</td>
<td>188</td>
<td></td>
</tr>
<tr>
<td>y</td>
<td>299</td>
<td>335</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>y</td>
<td>299</td>
<td>181</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>y</td>
<td>299</td>
<td>1251</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>y</td>
<td>299</td>
<td>1237</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>y</td>
<td>299</td>
<td>1830</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>y</td>
<td>299</td>
<td>403</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>y</td>
<td>299</td>
<td>2142</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>y</td>
<td>299</td>
<td>2293</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>y</td>
<td>299</td>
<td>386</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>y</td>
<td>299</td>
<td>228</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>y</td>
<td>299</td>
<td>3768</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>y</td>
<td>299</td>
<td>1172</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>y</td>
<td>299</td>
<td>7769</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>y</td>
<td>299</td>
<td>1020</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>y</td>
<td>299</td>
<td>6073</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Significance F

- Statistically significant - good correlation between variables
- OK - reasonable correlation between variables
- Statistically insignificant - poor correlation between variables
- P-value
  - Statistically significant - keep variable in regression model
  - Statistically insignificant - remove variable from regression model

Predicted Values for X2

Regression Statistics

- Multiple R: 0.71552391
- Adjusted R Square: 0.49454499
- Standard Error: 23.89668586
- Observations: 30

Output

- Y: 315.248

S-curve for X2 (X1=X3=0)
Case Study- Findings

• On Software with larger applications, about 95% of the time is spent on corrective maintenance.

• This indicates that more time is spent on reactive management of the software.

• In order to reduce software obsolescence, more time should be spend on preventive and perfective maintenance.
Summary

Software Obsolescence is an emerging issue and it is important to understand how much SW/Obs is going to cost at a very early stage of development life cycle. In order to do so, we need to:

- Define what Software obsolescence is
- Understand the difference between Software Maintenance and Obsolescence
- Identify how Software Obsolescence is triggered
- Have a framework to manage software obsolescence proactively
- Identify the key Software Obsolescence Resolution approaches
Conclusions

• Software plays an important role in defence. Almost every project in defence has software elements with various degrees of complexity and dependencies.

• In order to understand and see the bigger picture and challenges; software developers and the customers need to foresee the following issues that drive the whole life cost and should be in a position to develop innovative means to mitigate these issues by:

  – Anticipation of the Software Obsolescence at a very early stage of projects.
  – Understanding the technology insertion, technology update requirement.
  – Understanding the relationship between Software Maintenance and Software Obsolescence.
  – Anticipation of future capability integration to the existing platforms taking into account systems of systems, software to software and software to hardware integrations.
  – Formulation and evaluation of alternative architectural framework to inform the software designers that recognises the key market and cost drivers.
CADE is the authoritative data source for estimating, analyzing, and managing Major Defense Programs.

- **Cost**: Browse/Export Prime and Subcontractor Cost Data
- **Earned Value (EVM)**: Browse/Export Data on ACAT I Prime Contracts, Quick-look Visualization Tools
- **Acquisition**: Program Information, SAR/MAR Annual Funding, SAR/MAR Schedule Events, CARDs*
- **Technical**: Software Database, Electronic CARDs (eCARDs)*, Technical Data Reports*
- **Library**: Cost Estimates, Funding Memos, Program Briefings, Research Studies

**Coming Soon**
- CARD ingestion into CADE coming Fall 2017
- Technical Data reports currently being put into policy
- eCARD currently in development

**Statistics**
- 349 Programs
- 53,875 CSDR Submissions
- 3,648 EV Submissions

**URL**: http://cade.osd.mil
CADE Vision and Major Initiatives

What is the DCARC?

Plan
› Facilitates the development of plans with the CWIPT

Validate
› Locates errors and communicates with contractor to correct

Monitor
› Issues quarterly compliance criteria for delinquent programs

Analyze
› Further examination of program events

- Develops CSDR Policy
- Communicates with Program Offices, Service Cost Centers, CAPE and contractors
- Administers CAPE rating for DAES
- Trains community on CSDR policies and procedures

URL: http://cade.osd.mil
CADE Vision and Major Initiatives

**Why is CADE Important?**

CADE enables analysts and the Department to do more with less

CADE is both an online data platform and the cost community’s strategic initiatives to improve data collection.

- Provide decision makers with analyses
  - Allows for better acquisition strategies and execution
  - Shift from reactive to proactive holistic analysis
  - Informs lifecycle program decisions

- Quality and transparency of source data
  - Consistency – where data comes from, what we know about it
  - Enterprise data stewardship – enterprise agreement and accountability for what data means and how it is used
  - Reporting compliance improvement

- Data properly secured both at rest and in motion

---

**PDF Files**
- Paper printouts

**Excel Files**
- Electronic files

**DACIMS**
- Searchable database

**CADE**
- Improved UI Co-Plans
- Data visualization
- Strategic data planning

**CADE 2018**
- Comprehensive data
- Technical data
- Maintenance data
- Software data
- Flexfiles, eCard, CSDR+
- Integrated dashboard
- Guided workflows
- Better IT functionality
Analogous program CADE data is critical early in a program’s lifecycle. As the program progresses, its own actual data becomes invaluable in budget formulations, contract negotiations, and source selections.
CADE 2018 Architecture - Summary

CSDR+
- enhance CSDR data

FlexFile (-Q)
- automated detailed cost data

eCARD
- consumable program information

Guided Workflow
- intuitive online program planning

Integrated Dashboard
- customizable personalized information

API
- improved database foundations

CADE Vision and Major Initiatives
CADE Vision and Major Initiatives

CADE Architecture - Summary

1. CSDR+ enhance CSDR data

Enhanced data utility of CSDRs enabled by

› Contextual tagging
› Standardized metadata

Improved online user experience of CSDR data through

› Modern search/filtering
› Updated interactive analytics and visuals
CADE Vision and Major Initiatives
CADE Architecture - Summary

2 FlexFile (-Q)
automated detailed cost data

Search/Query Visualizations Prototype

Ability to create and validate client side FlexFile effort

*Ability to submit and validate a FlexFile is being developed under core CADE
CADE Vision and Major Initiatives

CADE Architecture - Summary

3 eCARD

consumable program information

Online site capabilities to view and consume CARD data
CADE Vision and Major Initiatives

CADE Architecture - Summary

4 Guided Workflow
intuitive online program planning

Will provide user a single, comprehensive, online, guided, and modern planning module
› Replaces planning functions within cPet Desktop, cPet Web, and PPM
› Incorporates Co-Plans

Future efforts include the entire submission/review workflow
CADE Vision and Major Initiatives
CADE Architecture - Summary

5 Integrated Dashboard
*customizable personalized information*

Centralized, customizable dashboard for personalized informational and data from multiple areas of CADE

Views and see the important data in one place, making use of CADE more efficient
CADE Vision and Major Initiatives

CADE Architecture - Summary

6 API
improved database foundations

Being applied to other capabilities such as CSDR+, eCARD, etc.

Automated ability for outside systems to consumer non-proprietary data from CADE
CADE Vision and Major Initiatives

Major Initiatives

Cost:

1921-3 improved ways of reporting business unit data

Bill of Materials
Standardized collection of parts and supplier pricing data

Technical Reporting:

Technical Data (1921-T) programmatic and technical descriptions analysts need to build estimates

Maintenance and Repairs (M/R) collection of information related to each maintenance event such as the specific system being repaired and reason for failure
MAIS Reporting Update
Software Resource Data Reporting (SRDR)
Overview

SRDR

Development:

Introduction of Agile Measures
› Supports capturing the metrics and effort associated with this SW dev methodology

Maintenance:

Collection of Information Assurance and Vulnerability Assessment (IAVA) data
› Ability to distinguish IAVA-related releases
› Clarified SW change count definitions to include IAVA

Updated SW Maintenance Effort definitions
› SW License Management is a PM activity
<table>
<thead>
<tr>
<th>Formats</th>
<th>Development</th>
<th>Maintenance</th>
<th>ERP</th>
</tr>
</thead>
<tbody>
<tr>
<td>Technical Data</td>
<td>SW size, context, technical information</td>
<td>SW size, context, technical information</td>
<td>SW product, context, object sizing and implementation</td>
</tr>
<tr>
<td></td>
<td>Release level and computer SW configuration item (CSCI) level sections</td>
<td>Top level and release level sections</td>
<td>Project, Sizing and Implementation sections captured at the release level</td>
</tr>
<tr>
<td>Effort Data</td>
<td>Reports SW efforts associated with each reported release and CSCI</td>
<td>Reports the to-date SW maintenance efforts for each in-progress and completed release(s), and total maintenance activities</td>
<td>Project resource and schedule information at the release level</td>
</tr>
</tbody>
</table>
CADE Vision and Major Initiatives
CADE Community

Cost

- **FlexFile:** Daron Fullwood, CAPE
- **CSDR/EVM Co-Plan:** John McGregor, AT&L PARCA/EVM
- **1921-3:** Mike Biver and Carol Moore, CAPE
- **Sustainment:** Tom Henry, CAPE; Lisa Mably, AFCAA
- **Materials:** Praful Patel, NCCA

Office Collaboration

- AFCAA CEM joint CADE effort, commodity leads, Contract Databases, Software & Technical Data, CARDS, SAR database
- FlexFile, JCARD (NAVAIR), Ships WG, CCRL, CER Handbook, SAR database
- MDA-DCARC alignment, CCRG

- USMC BOM/CER Effort
- JIAT, ACDB/WTW/Missile prototype, TACOM WTW CIPT, Historical Data Migration
- EVM-CR, DAVE (DAMIR, AIR, Kaleidoscope)
- DDR&E/SE tech data; LM&R CARD input, DCMA, DPAP, DAU, Big Data initiative, CSDR/EVM Co-Plans

Technical

- **SRDR:** Ranae Woods, AFCAA
- **CARD:** Curt Khol, CAPE
- **Tech Data:** Greg Hogan, AFCAA
- **MAIS:** Richard Mabe, AFCAA
- **Maintenance & Repair:** Lisa Mably, AFCAA

Commodity Study Joint Effort

- Aircraft
- Missiles
- Radar, C2 Center, C41
- ICBM
- O&S
- Ships
- Space
- WTV
- MAIS
- UAV

Service Cost Agency Leads

- David Henningsen
- Katherine McCormack
- Duncan Thomas
- Justin Moul
- Ranae Woods
- Greg Hogan

Industry

- CSDR Focus Group, Joint Training, NDIA,
  **FlexFile Pilot Leads:** LMCO, Boeing, NGC, BAE, GDLS, HII, Ball Aerospace
- **CIPTS:** Aviation, JSOC, O&S, Software and IT, WTV
CADE Vision and Major Initiatives

Points of contact

Director, Cost Assessment Data Enterprise (CADE)
Bess Dopkeen
(703) 695-7282 Office
bess.r.dopkeen.civ@mail.mil

Director, Defense Cost and Resource Center (DCARC)
Daron Fullwood
571-372-4267 Office
Daron.D.Fullwood.civ@mail.mil

CADE Help Desk
(253) 564-1979
cadesupport@tecolote.com

CADE Training
Torri Preston
571-372-4270 Office
Torri.R.Preston.ctr@mail.mil
## Major Initiatives

### Cost

<table>
<thead>
<tr>
<th>Cost/Quantity Reporting</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Cost Data (CCDRs/1921s)</strong> – contains most of what analysts need to build an estimate – dollars, hours, quantities, and descriptive tagging</td>
</tr>
<tr>
<td><strong>FlexFiles</strong> – new generation of cost reporting, government data reporting</td>
</tr>
<tr>
<td><strong>Quantity Report (1921-Q)</strong> – provides actual account of physical units completed in a streamlined submission process</td>
</tr>
<tr>
<td><strong>1921-3</strong> – improved ways of reporting business unit data</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Bill of Materials</th>
</tr>
</thead>
<tbody>
<tr>
<td>Standardized collection of parts and supplier pricing data</td>
</tr>
</tbody>
</table>

### Technical

<table>
<thead>
<tr>
<th>Technical Reporting</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Cost Analysis Requirements Description (CARD) / Technical Data (1921-T)</strong> – programmatic and technical descriptions analysts need to build estimates</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Software</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>SRDRs</strong> – software effort, size, and schedule estimating approaches including analogy, parametric, and commercial models</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Maintenance &amp; Repairs</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>1921-M/R</strong> – collection of information related to each maintenance event such as the specific system being repaired and reason for failure</td>
</tr>
</tbody>
</table>

### Strategic Planning

<table>
<thead>
<tr>
<th>Co-Planning</th>
</tr>
</thead>
<tbody>
<tr>
<td>Reporting strategy that aligns CSDR &amp; EVM requirements</td>
</tr>
<tr>
<td>Cooperative planning leads to better data, lower costs, and improved program management</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Institutional Knowledge/Community Support</th>
</tr>
</thead>
<tbody>
<tr>
<td>What analysts need to know about the data</td>
</tr>
<tr>
<td>Additional contextual information on programs</td>
</tr>
</tbody>
</table>

Cost analysts will have all of this data and institutional knowledge at their fingertips. It will be the exception – not the rule – to go back to industry to do our estimates.
COCOMO III Workshop: Implementing a New Driver for Software Security

Brad Clark, PhD
Software and IT-CAST Meeting
August 22, 2017
Abstract

• Making software applications secure from intrusion, corruption, attack, denial of service and other things is challenging. Does it really cost that much more to make software secure?
• This workshop will discuss what it means to make software secure and where it might cost more to implement security measures.
• The COCOMO III model needs to consider the costs associated with building secure software.
Topics

• Software Security Overview
  – Why Software Security
  – Supply Chain Management Impact
  – Examples of Software Weaknesses
  – Software Component Security Requirements
  – Software Development Security Requirements

• COCOMO III Model Overview

• Discussion on Implementing a New Driver
Why is Software Security Important?

• There have been dramatic increases in business and mission risks attributable to exploitable software

• Software vulnerabilities jeopardize
  – intellectual property
  – consumer trust
  – business operations and services
  – broad spectrum of critical infrastructures (including everything from process control systems to commercial software products)

• Recent examples:
  – Recent NSA ransom ware attacks
  – Foreign hackers 'may have hit voter site days before referendum’
  – US child hacker launches cyber attack on Brussels Airport
  – Penthouse and Adult Friend Finder hack leaves over 412 million exposed… Oops
Is It Worth It?

• How much additional effort (cost) does it take to develop secure software considering the impact of:
  – Security requirements for software
    • Impacted by levels of security
  – Implementation expertise
  – Testing independence
  – Process and tool support
  – Platform constraints and configurations (volatility)

• Two cost aspects:
  – Software component security requirements
  – Management of a secure development lifecycle process
Non-Functional Requirement Tensions

- Functional requirements specify the work for which the system is intended
- Non-Functional requirements pertain to the functions of the system
- There is a tradeoff between Security and other Non-Functional req’ts

- Availability
- Performance
- Modifiability
- Interoperability
- Reliability
- Scalability
- Security
- Usability
- Testability
- Safety
- Portability
Application Development Context

Asset Owner

Operational and Maintenance capabilities (policies & procedures)

System Integrator

Automated Solution

Product Supplier

Product

- Subsystem-1
- Subsystem-2
- Complementary HW & SW

- Software Applications
- Embedded Devices
- Network Components
- Host Devices

Source: ISA-62443-4-1 Secure Product Development Lifecycle Requirements
Development Supply Chain Context

What security was implemented in the software down-line from the acquired product?

Defense in Depth is a design concept that attempts to address this issue.

Source: adapted from DHS, “Software Assurance in Acquisition and Contract Language”
Examples of Weaknesses Introduced During Design

- Acceptance of Extraneous Untrusted Data With Trusted Data
- Access to Critical Private Variable via Public Method
- Addition of Data Structure Sentinels, e.g. null character at the end of strings
- Algorithmic Complexity
- Allocation of File Descriptors or Handles Without Limits or Throttling
- Allocation of Resources Without Limits or Throttling
- Incorrect Control Flow Implementation
- Apple '.DS_Store'
- Argument Injection or Modification
- ASP.NET Misconfiguration: Not Using Input Validation Framework
- Asymmetric Resource Consumption (consume more resources than the access level permits)

Source: Mitre-CWE, Common Weakness Enumeration A Community-Developed List of Software Weakness Types
Weaknesses in the 2011 CWE/SANS Top 25 Most Dangerous Software Errors Examples

• Cross-Site Request Forgery (CSRF)
  – Improper Neutralization of Input During Web Page Generation (‘Cross-site Scripting’)  
  – Improper Neutralization of Special Elements used in an OS Command (‘OS Command Injection’)  
  – Improper Neutralization of Special Elements used in an SQL Command (‘SQL Injection’)  

• Porous Defenses
  – Execution with Unnecessary Privileges
  – Improper Restriction of Excessive Authentication Attempts
  – Incorrect Authorization
  – Incorrect Permission Assignment for Critical Resource

Source: Mitre-CWE, Common Weakness Enumeration A Community-Developed List of Software Weakness Types
Examples of Weaknesses in SW Written in C++

- Access of Resource Using Incompatible Type ('Type Confusion')
- Access to Critical Private Variable via Public Method
- Base Addition of Data Structure Sentinel
- Assignment of a Fixed Address to a Pointer
- Buffer Access with Incorrect Length Value
- Base Buffer Copy without Checking Size of Input ('Classic Buffer Overflow')
- Buffer Underwrite ('Buffer Underflow')
- Cloneable Class Containing Sensitive Information
- Compiler Optimization Removal or Modification of Security-critical Code

Source: Mitre-CWE, Common Weakness Enumeration A Community-Developed List of Software Weakness Types
Software Security Requirements

• What are examples of security requirements for software?
• Many security resources discuss security policy
• In this presentation, one set of requirements was selected to provide insight:
  – ISA-62443-4-2 Security for Industrial Automation And Control Systems Technical Security Requirements for IACS Components

  – ISA: International Society of Automation
  – IACS: Industrial automation and control system
# Security Requirements for Software Components - 1

## 1. Identification and authentication control

- Human user identification and authentication
- Software process and device identification and authentication
- Account management
- Identifier management
- Authentication management …

## 2. Use control

- Authorization enforcement
- Wireless control
- Use control for portable and mobile devices
- Session lock
- Remote session termination …

Source: ISA-62443-4-2 Technical Security Requirements for IACS Components

22 August 2017  Brad Clark - Software & IT-CAST - ©2017 USC-CSSE
Security Requirements for Software Components - 2

3. System integrity
- Communication integrity
- Malicious code protection
- Software and information integrity
- Input validation
- Error handling ...

4. Data confidentially
- Information confidentiality
- Information persistence
- Use of cryptography

5. Restricted data flow
- Network segmentation
- Zone boundary protection
- Person-to-Person communication restrictions

22 August 2017
Security Requirements for Software Components - 3

6. Timely response to events

- Audit log accessibility
- Continuous monitoring

7. Resource availability

- Denial of service protection
- Resource management
- Control system backup, recovery and reconstitution
- Network and security configuration settings
- Least functionality
Component Security Levels

- The seven security requirements shown previously have four Security Levels (SL).
- Identify and authenticate all users (humans, software processes and devices) by mechanisms that

  **SL-1** – Protect against *casual or coincidental access* by unauthenticated entities.

  **SL-2** – Protect against intentional unauthenticated access by entities using *simple means with low resources, generic skills and low motivation*.

  **SL-3** – Protect against intentional unauthenticated access by entities using *sophisticated means with moderate resources*.

  **SL-4** - Protect against intentional unauthenticated access by entities using *sophisticated means with extended resources*.

Source: ISA-62443-4-2 Technical Security Requirements for IACS Components
Impact of Component Security Requirements on Development Effort

• More requirements affect software effort (cost) by increasing the functionality (or size) to be implemented in the software

• The four security levels shown previously increase the amount of functionality (and size) and therefore effort

• The amount of effort required, directly related to the amount of functionality, is influenced by other factors such as
  – Product Factors (e.g. complexity, reliability)
  – Personnel Factors (e.g. capabilities, experience)
  – Platform Factors (e.g. constraints, volatility)
  – Project Factors (e.g. precededness, risk resolution, process capability, development flexibility, tools)

• These are addressed next
Secure Development Lifecycle - 1

• Security management
  – Identification of responsibilities
  – Security expertise
  – Code signing
  – Development environment security
  – 3rd party embedded component security
  – Process verification

• Specification of security requirements
  – Product security requirements (authentication, authorization, encryption, auditing and other security capabilities)
  – Product security context (product’s intended operating environment including physical environment)
  – Threat model (analysis that identifies potential security issues and how they will be addressed)
  – Security requirements review

Source: ISA-62443-4-1 Secure Product Development Lifecycle Requirements
Secure Development Lifecycle -2

• Secure by design
  – Secure design principles
  – Defense in depth design (layers of security)
  – Security design review
  – Assessing & addressing security-related issues

• Secure implementation
  – Security implementation review
  – Assessing & addressing security-related issues

• Security verification and validation testing
  – Security requirements testing
  – Threat mitigation testing
  – General vulnerability testing
  – Penetration testing
Secure Development Lifecycle -3

• Security defect management
  – Receiving notifications of security-related issues
  – Reviewing security-related issues
  – Assessing & addressing security-related issues
  – Disclosing security-related issues

• Security update management
  – Dependent component or operating system security update documentation
  – Security update delivery
  – Timely delivery of security patches
Conclusions

• Software component security requirements affect the amount of functionality

• Software development security requirements affect the productivity of the work

• Security Levels affect both the
  – Amount of functionality, e.g. more software to be developed
  – Amount of development tasks, e.g. increased reviews, testing, audits
Topics

• Software Security Overview
  – Why Software Security
  – Supply Chain Management Impact
  – Examples of Software Weaknesses
  – Software Component Security Requirements
  – Software Development Security Requirements

• COCOMO III Model Overview

• Discussion on Implementing a New Driver
COCOMO III Model

Software product size estimate
Software product, platform, personnel & project attributes
Software reuse, maintenance, and increment parameters
Defect removal profile levels

Software development and maintenance estimates for:
- Effort
- Cost & Schedule distributed by:
  - Phase
  - Activity
  - Increment
- Quality

Local calibration to organization’s data

COCOMO is an open and free model
COCOMO III Model Concept

- Software product size estimate
- Software product, platform, personal & project attributes
- Labor Rates
- Defect removal profile levels

- Defect Introduction Model
- Effort Model
- Schedule Model

- Number of est. non-trivial defects for Requirements, Design, & Code
- Number of est. residual defects and the residual defect density
- Schedule (Months)
- Effort (Person Months)
- Costs ($$)
- Staffing Levels

- 22 August 2017
- Brad Clark - Software & IT-CAST - ©2017 USC-CSSE
COCOMO III Effort & Schedule Estimation Model

Effort (PM) = A * Size\(^E\) * Product(14 Cost Drivers)

\[ E = B + \text{Sum}(5 \text{ Cost Drivers}) \]

Schedule (M) = C * PM\(^F\) * SCED\%/100

\[ F = D + 0.2(E-B) \]

Where:
A, B, C, D are constants determined by calibration
E represents (dis)economies of scale and project-wide scale factors
Defect Introduction (DI) = A * Size^E * Product(DI Drivers)

E = Initially set to 1.0

Residual Defects = C * DI * Product(1 – DRF)

DRF: Defect Removal Fraction from 3 profiles:
1. Automated Analysis
2. People Reviews
3. Execution Testing
COCOMO III Cost Drivers -1

- **Product Attributes**
  - Impact of Software Failure (FAIL) (formerly RELY)
  - Product Complexity (CPLX)
  - Developed for Reusability (RUSE)
  - Required Software Security (SECU) - *New*
  - **Dropped:**
    - Documentation Match to Lifecycle Needs
    - Database Size

- **Platform Attributes**
  - Platform Constraints (PLAT) – *New*
  - Platform Volatility (PVOL)
COCOMO III Cost Drivers -2

• Personnel Attributes
  – Analyst Capability (ACAP)
  – Programmer Capability (PCAP)
  – Personnel Continuity (PCON)
  – Applications Experience (APEX)
  – Language and Tool Experience (LTEX)
  – Platform Experience (PLEX)
COCOMO III Cost Drivers -3

• Project Attributes
  – Precedentedness (PREC)
  – Development Flexibility (FLEX)
  – Opportunity and Risk Resolution (RESL)
  – Stakeholder Team Cohesion (TEAM)
  – Process Capability & Usage (PCUS) (formerly PMAT)
  – Use of Software Tools (TOOL)
  – Multisite Development (SITE)

• Defect Removal Profile
  – Automated Analysis
  – People Reviews
  – Execution Testing and Tools
Topics

• Software Security Overview
  – Why Software Security
  – Supply Chain Management Impact
  – Examples of Software Weaknesses
  – Software Component Security Requirements
  – Software Development Security Requirements

• COCOMO III Model Overview

• Discussion on Implementing a New Driver
COCOMO III Workshop

• This activity focuses on which COCOMO III Cost Drivers are impacted by software component and development requirements.
• You are asked to examine the requirements on the following pages and identify the applicable Cost Driver that addresses that requirement
• Refer to the handout
Glossary

• CWE: Common Weakness Enumeration
• ISA: International Society of Automation
• IACS: Industrial Automation And Control System
• Vulnerability: A vulnerability is a software weakness that can be exploited by an attacker. Bugs and flaws collectively form the basis of most software vulnerabilities.
• Weakness: A weakness is an underlying condition or construct existing in a software system that has the potential for negatively impacting the security of the system.
Resources

- Lots of papers
  - [https://www.us-cert.gov/security-publications](https://www.us-cert.gov/security-publications)
For more information, requests or questions, please contact

Brad Clark
Software Metrics, Inc.
brad@software-metrics.com
703-402-3576