Definition of Cost Metrics

PSM Users Group Conference

25 June 2009

Orlando, Florida

Agenda

- Overview of AFCAA sponsored work on the Software Cost Metrics Manual
- Present current definitions for application domains, size, effort, and milestones
- Workshop will focus on obtaining input on the following cost measurement issues:
  - Application domain examples
  - Sizing pre-existing code using reuse parameters

PROBLEM STATEMENT

- **Data Integrity**
  - **Despite increased interest in software data collection, it is surprising that so little effort has been committed to reconciling Data Quality and Standardization Issues…**
    - No reporting of Equivalent Size Inputs – CM, DM, IM, SU, UNFM, Type
    - No common SLOC reporting – logical, physical, etc.
    - No standard definitions – Application Domain, Build, Increment, Spiral,…
    - No common effort reporting – analysis, design, code, test, CM, QA,…
    - No common code counting tool
    - Product size only reported in lines of code
    - No reporting of quality measures – defect density, defect containment, etc.

PROBLEM STATEMENT (cont.)

- **Limited Empirical Research within DoD**
  - **Other contributors to Productivity besides effort and size, are being ignored by most analysts**
    - Operating Environment, Application Domain, and Product Complexity
    - Personnel Capability
    - Required Reliability
    - Quality – Defect Density, Defect Containment
    - Integrating code from previous deliveries – Builds, Spirals, Increments, etc.
    - Requirements Volatility
  - **Converting to Equivalent SLOC**
    - Categories like Modified, Reused, Adopted, Managed, and Used add no value unless they translate into single or unique narrow ranges of DM, CM, and IM parameter values. We have seen no empirical evidence that they do…

PROBLEM STATEMENT (cont.)

- **Software Model Calibration**
  - **Most program offices and support contractors rely heavily on software cost models**
  - **May have not been calibrated with most recent DoD data**
  - **Calibration with recent data (2002-Present) will help increase program office estimating accuracy**

SEER®

SLIM-Estimate™

TruePlanning® by PRICE Systems

PROPOSED SOLUTION

AFCAA in conjunction with USC and cost agencies, will publish a manual to help analysts develop quick software estimates using <u>empirical metrics</u> from recent programs

Team Members

AFCCA Sponsor	Wilson Rosa
USC Team	Barry Boehm (Lead)
	Don Reifer (Domains & Productivity)
	Ray Madachy (Sizing)
	Brad Clark (Data Analysis)

Seeking new collaborators

Special Features

- **Augment NCCA/AFCAA Software Cost Handbook:**
  - **Default Equivalent Size Inputs (DM, CM, IM, SU, UNFM); Requirements Volatility**
  - **Productivity Benchmarks by Operating Environment, Application Domain, and Software Size**
  - **Empirical Code, Effort, and Schedule Growth Measures derived from SRDRs**
  - **Empirically Based Cost Risk and Uncertainty Analysis Metrics**
  - **Calibrated SLIM-Estimate™ using most recent SRDR data**
  - **Mapping between COCOMO, SEER, True S cost drivers**
  - **Empirical Dataset for COCOMO, True S, and SEER Calibration**
  - **Software Maintenance Parameters**

Special Features (cont.)

- **Guidelines for reconciling inconsistent data**
- **Standard Definitions (Application Domain, SLOC, etc.)**
- **Address issues related to incremental development (overlaps, early-increment breakage, integration complexity growth, deleted software, relations to maintenance) and version management (a form of product line development and evolution).**
- **Impact of Next Generation Paradigms – Model Driven Architecture, Net-Centricity, Systems of Systems, etc.**

Table of Contents

Chapter 1: Software Estimation 101

Chapter 2: Software Sizing

Chapter 3: Software Growth

Chapter 4: Equivalent SLOC

Chapter 5: Productivity Benchmarks

Chapter 6: Model Calibration

Chapter 7: Calibrated SLIM-ESTIMATE

Chapter 8: Cost Risk and Uncertainty Metrics

Chapter 9: Data Normalization

Chapter 10: Software Resource Data Report

Chapter 11: Software Maintenance

Chapter 12: Lessons Learned

Completed by Sep 09

Agenda

- Overview of AFCAA sponsored work on the Software Cost Metrics Manual
- Present current definitions for application domains, size, effort, and milestones
- Workshop on the following cost measurement issues:
  - Application domain examples
  - Sizing pre-existing code using reuse parameters

Cost Metrics

- **Data used for cost estimation**
  - **<u>Size</u> input represents the amount of work to be done**
    - **Functional Size**
    - **Product Size**
  - **<u>Effort</u> output represents cost**
  - **<u>Duration</u> output represents time to complete work**

Functionality ≈ Size ≈ Effort ≈ Duration

When using Source Lines of Code (SLOC), estimation practice is to use “New SLOC” as the size input

Cost Metrics

- Estimation Context
  - Operating environment and type of application being developed
    - Some applications are more difficult to develop than other applications
  - Lifecycle used to develop application, e.g. overlapping iterations
  - Amount of reuse or automatically generated functionality
    - Using pre-existing artifacts can impact the amount of effort
    - Auto-generated functionality requires some effort
  - Project characteristics, e.g.
    - Product, platform, personnel, project

Operating Environments

Operating Environments
1. Avionics
2. Business
3. Logistics
4. Manned Space
5. Manned Ground
6. Military Mobile
7. Missile and Unmanned Airborne
8. Shipboard
9. Telecommunications
10. Unmanned Space
11. Web

Comparing results from an Avionics development with that of a Business project makes no sense what-so-ever

Application Domains

Application Domains	
1. Bus	13. Payload
2. Command & Control	14. Platform
3. Communications	15. Process Control
4. Controls & Displays	16. Radar
5. Database	17. Signal Processing
6. Electronic Warfare	18. Simulation & Modeling
7. Executive	19. Situation Awareness
8. Information Assurance	20. Sonar
9. Logistics	21. Test & Evaluation
10. Maintenance & Diagnostics	22. Tool and Tool Systems
11. Mission Management	23. Training
12. Mission Planning	24. Weapons Delivery

Size Definitions and Issues

- **Desire consistent size definitions and measurements across different models and programming languages**
- **Using Source Lines of Code (SLOC) as common measure**
  - **Logical source statements consisting of data declarations executables**
  - **Rules for considering statement type, how produced, origin, build, etc.**
  - **Providing automated code counting tools adhering to definition , including initial modified-code counting**
  - **Providing conversion guidelines for physical statements**
- **Addressing other size units such as requirements, use cases, etc.**

Effort

- Software development involves much more activity than just coding
  - It includes the work involved in developing requirements, designs and tests
  - It involves documentation and reviews, configuration management, and quality assurance
  - It can be done using different life cycles (agile, incremental, spiral, waterfall, etc.) and different ways of organizing the work (matrix, product lines, etc.)

Effort - Included

- Software Requirements Analysis (including any prototyping activities)
- Software Architecture and Detailed Design
- Software Coding and Unit Test
- Software Integration and System/Software Integration
- Software Qualification Testing
- Software Development Test and Evaluation
- Other Direct Software Engineering Development support

Effort – Not Included

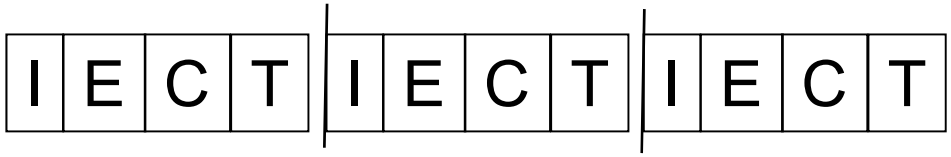
- System Conceptualization
- Operational Concept Development
- Systems Requirements Development (including equations engineering and allocation to hardware and software)
- Hardware/Software Integration and Test
- System Test and Evaluation
- Operational Test and Evaluation
- Production

New Software Lifecycle Activities

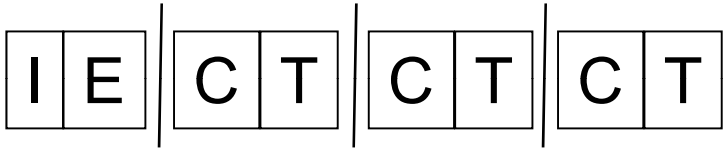
Inception (I)	Elaboration (E)	Construction (C)	Transition (T)
Operational Concept Development	Software Requirements Analysis	Software Detailed Design	To Hardware & Software Integration
Systems Requirements Development	Software Architecture Development	Software Coding and Unit Test	
Software Requirements Development	Software Test Planning	Software Integration	
Requirement Prototyping	Design Prototyping	Software Qualification Testing	
		Software Test & Evaluation	

Lifecycle Challenges to Data Collection

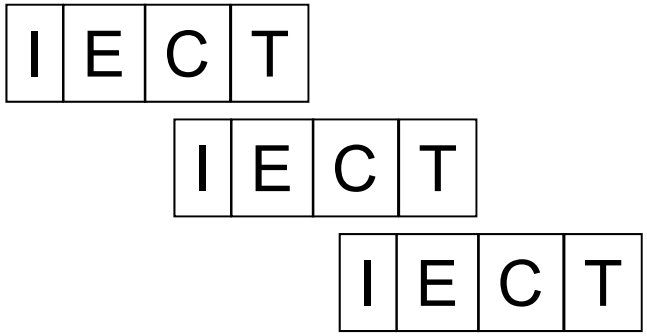
Evolutionary Sequential



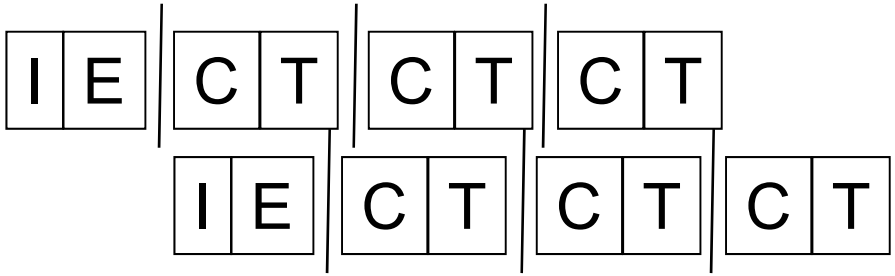
Pre-specified Sequential



Overlapped Increments



Overlapped Rebaselining



Agenda

- Overview of AFCAA sponsored work on the Software Cost Metrics Manual
- Present current definitions for application domains, size, effort, and milestones
- Workshop on the following cost measurement issues:
  - Application domain examples
  - Sizing pre-existing code using reuse parameters

Application Domains

Application Domains (24)	
Bus	Software used to manage infrastructure services (attitude and articulation control, commanding, etc.) for manned or unmanned space applications.
Command and Control	Software used for network monitoring, network control and switching, sensor control, signal/telemetry processing, message processing, data reduction/analysis, mission control and command processing in any of the operating environments described above.
Communications	Software used for transmission and receipt of voice, data, digital and video information within or to/from operating environments described above.
Controls and Displays	Software that provides the interface between the user and system within any of the operational environments described above. For example, software that controls the Heads-Up Display (HUD) in an aircraft provides the pilot with a virtual environment during engagement.
Database	Software used to collect, manage and search information stored as files or databases in any of the operating environments described above.

Application Domains

Electronics Warfare	Software used to deny the opponent an advantage in the electromagnetic spectrum through its attack, protection and support components.
Executive	Systems software used to control the hardware and operating environment within which software applications execute within any of the operating environments described above. Executive software is typically specially developed to control the environment because the operating system or standard platform software cannot accommodate specialized run-time requirements.
Information Assurance	Software used to protect and defend information systems to ensure confidentiality, integrity, authentication, availability and non-repudiation within any of the operating environments described above.
Logistics	Software used to manage the flow of goods, information and other resources involved in maintaining and sustaining systems operational in the field.

Application Domains

Maintenance and Diagnostics	Software used to perform maintenance functions including diagnostics and diagnosis of problems.
Mission Management	Software used to support mission management activities in any of the operating environments described above such as the command and control of systems aboard an aircraft.
Mission Planning	Software used to support mission planning activities in any of the operating environments described above including those for scenario generation, feasibility analysis, route planning, and image/map manipulation.
Payload	Software used to manage payload functions (experiment control, sensor management, etc.) for manned or unmanned space applications.
Platform	Software used to bind, load, schedule and run applications on the selected computer hardware. Typically such software includes the operating system, linker/loader, middleware and utilities required operationally for any of the operating environments described above. In some cases, such platforms provide services like database management and security as part of the platform offerings.

Application Domains

Process Control	Software that provides closed-loop feedback controls for applications like attitude control, propulsion, etc. that run in real-time in industrial and military environments.
Radar	Software used to automate radar functions in any of the operating environments described above.
Signal Processing	Software used to enhance, transform, filter, convert or compress signal data typically in real-time in any of the operating environments described above.
Simulation and Modeling	Software used to evaluate scenarios and assess empirical relationships that exist between models of physical processes, complex systems or other phenomena that may have complex relationships in any of the operating environments described above.
Situation Awareness	Software used to provide decision-makers with a perception of their environment within a volume of time and space within any of the complex, dynamic operating environments described above.

Application Domains

Sonar	Software use to automate sonar functions in any of the operating environments described above.
Test and Evaluation	Software used to support the test and evaluation of software within software, hardware/software and pseudo-operational environment like test labs within any of the operating environments described above. Activities performed include test case generation, test data recording, test data reduction/analysis and test driver/tool development.
Tool and Tool Systems	Software packages and/or integrated development environment that are used for analysis, design, construction and test of operational software for the operating environments described above.
Training	Software used to support education and training of users within any of the operating environments described above. Activities performed include computer-based training and computer-aided instruction within the operational and/or dedicated training systems.
Weapons Delivery	Software used to target, control and command weapons within any of the operating environments described above.

Operating Environments

Operating Environments (11)	
Avionics	Software developed for aircraft avionics systems like those found on aircraft, helicopters, etc.
Business	Software developed for systems that process business information like financial and management information systems.
Manned Space	Software developed for manned space applications like the space shuttle or station.
Manned Ground	Software developed for ground-based systems like artillery and tactical information centers.
Military Mobile	Software developed for ground-based mobile systems like mobile command stations (in trucks or vans), tanks and transports of any kind (trucks, trains, etc.).

Operating Environments

Missile and Unmanned Airborne	Software developed for missiles or unmanned airborne systems like an air-to-air missile or remotely piloted vehicle.
Shipboard	Software developed for sea based systems like aircraft carriers, destroyers, freighters, submarines and tankers.
Telecommunications	Software developed for communications systems that transmit voice, data, video or other information.
Unmanned Ground	Software developed for unmanned ground-based systems like reconnaissance vehicles.
Unmanned Space	Software developed for unmanned space systems used for planetary exploration, imagery or weather prediction.
Web	Software developed for applications that run and utilize the Internet.

Productivity Groups

Application Domains	Operating Environments					
	Avionics	Business	Manned Space	Manned Ground	Military Mobile	…
Bus						
Command & Control						
Communications						
Controls & Displays						
Database						
Electronic Warfare						
Executive						
Information Assurance						
…						

Each group might have suggestions for:
- Equivalent sizing
- Productivities
- Cost model parametric inputs

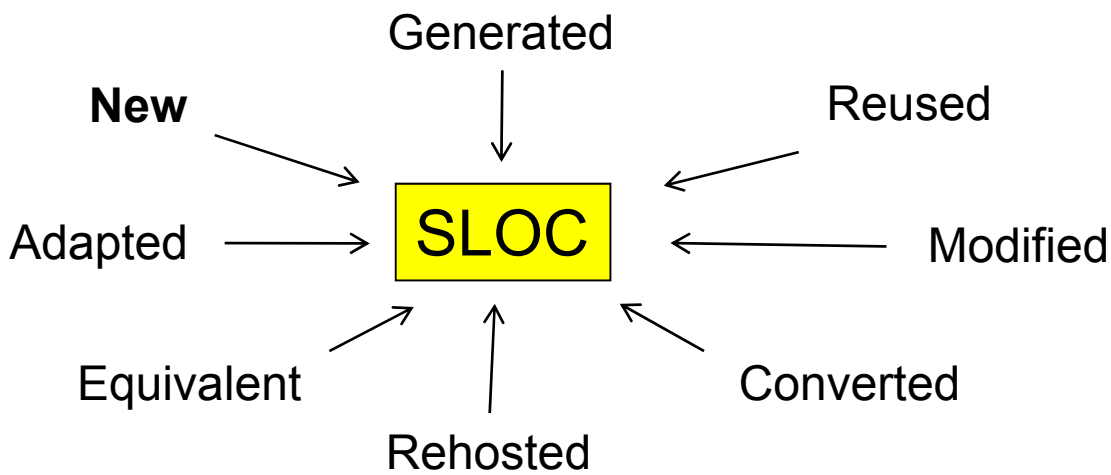
Application Domain Validation by Example Exercise

Size Input

Functionality ≈ Size ≈ Effort ≈ Duration

When using Source Lines of Code (SLOC), estimation practice is to use “New SLOC” as the basis of size input

---



SLOC Size Type Definitions

Size Type	Description
New	Software developed entirely from scratch using no existing software artifacts.
Adapted	Pre-existing software that is used as-is (Reused) or changed (Modified).
Reused	Pre-existing software that is not changed with the adaption parameter settings: • Design Modification % (DM) = 0% • Code Modification % (CM) = 0%
Modified	Pre-existing software that is modified for use by making design, code and/or test changes: • Code Modification % (CM) > 0%
Equivalent	A relative measure of the work done to produce software compared to the code-counted size of the delivered software. It adjusts the size of software relative to developing it all new. This is also sometimes called *Effective* size.
Generated	Software created with automated source code generators. The code to include for equivalent size may consist of the generator statements directly produced by the programmer, or the 3GL generated statements produced by the automated tools.
Converted	Software that is converted between languages using automated translators.

SLOC Conversion Parameters

How do you convert different types of SLOC to a "New" SLOC

Adaptation Parameter	Description
DM (Design Modified %)	The percentage of the adapted software's design which is modified in order to adapt it to the new objectives and environment.
CM (Code Modified %)	The percentage of the adapted software's code which is modified in order to adapt it to the new objectives and environment.
IM (Integration % Required)	The percentage of effort required to integrate the adapted software into an overall product and to test the resulting product as compared to the normal amount of integration and test effort for software of comparable size.

Adaption Adjustment Factor (AAF) = 0.4*DM + 0.3*CM + 0.3*IM

SLOC Conversion Parameters

Software Understanding

	Very Low	Low	Nominal	High	Very High
Structure	Very low cohesion, high coupling, spaghetti code.	Moderately low cohesion, high coupling.	Reasonably well-structured; some weak areas.	High cohesion, low coupling.	Strong modularity, information hiding in data / control structures.
Application Clarity	No match between program and application world-views.	Some correlation between program and application.	Moderate correlation between program and application.	Good correlation between program and application.	Clear match between program and application world-views.
Self-Descriptive -ness	Obscure code; documentation missing, obscure or obsolete	Some code commentary and headers; some useful documentation .	Moderate level of code commentary, headers, documentation.	Good code commentary and headers; useful documentation; some weak areas.	Self-descriptiv code; documentatior up-to-date, well-organized with design rationale.
SU Increment to ESLOC	50	40	30	20	10

SLOC Conversion Parameters

Programmer Unfamiliarity

UNFM Increment	Level of Unfamiliarity
0.0	Completely familiar
0.2	Mostly familiar
0.4	Somewhat familiar
0.6	Considerably familiar
0.8	Mostly unfamiliar
1.0	Completely unfamiliar

SLOC Conversion Parameters

AAM = AAF * [1 + (.02 * SU * UNFM)] , whenAAF ≤ 50%
AAM = AAF + [SU * UNFM] , when AAF > 50%

Total Equivalent Size = New Size + [AAM * Adapted Size]

Adaption Parameter
“Guidance”
Exercise

Example – Draft Table

	DM	CM	IM	Resulting AAF	SU	UNFM	Resulting AAM
Modified							
Major Mod	50	50	75	57.5	30	0.1	0.63
Minor Mod	10	20	50	25	20	0.1	0.28
Reused							
As-is	0	0	30	9	0	0	0.09
Tailored	0	10	40	15	20	0.2	0.16
Generated							
As-is	0	0	50	15	20	0.2	0.16
Optimized	10	10	100	37	20	0.2	0.40
Converted							
As-is	0	0	100	30	20	0.2	0.32
Optimized	20	20	100	44	30	0.2	0.49
Rehosted							
Major	30	30	100	51	30	0.2	0.59
Minor	10	10	50	22	20	0.1	0.25