Software Effort Estimating Methods
Objectives

• Following this lesson you will be able to
  • Identify major software cost drivers
  • Chose sizing approach using best practice guidelines
  • Select effort estimation method according to
    • Project Maturity
    • Available dataset for regression analysis and benchmarking
  • Acquire knowledge of software development life cycles
Course Outline

• Core Knowledge
  • Software Cost Drivers
  • Estimating Methods

• Summary

• Related and Advanced Topics
  • Software Development Life Cycles
  • Source Lines of Code Sizing Best Practices
Course Outline

• Core Knowledge
  • Software Cost Drivers
    • Estimating Methods
• Summary
• Related and Advanced Topics
  • Software Development Life Cycles
  • Source Lines of Code Sizing Best Practices
Software Cost Drivers

Product Size

• Software Sizing
• Software Reuse
• Requirements Volatility
• Growth

Development Environment

• Process Maturity
• Development Process
• Development Tools
• Language
• Reliability
• Time Constraint

Software Complexity

• Application Domain
• Operating Environment

Developer Capability

• Team Size
• Personnel Capability
• Application Experience
• Multi-site development
• Platform Experience
• Team Cohesion

Effort Estimation

Major Cost Driver based on scholarly literature. Available before contract award

Qualitative. Difficult to collect at early phase. Limited research on their correlation to effort

Categorical
Software Sizing

• Analysts begin a software cost estimate by predicting the sizes of the deliverables that must be developed.

• Software sizing is the process of determining how big the application being developed will be.

• Sizing depends on program’s complexity, functions, safety requirements, reliability constraints.
  • Programs (e.g. aircraft) that are more complex, perform many functions, and have safety requirements, and require high reliability compliance, are typically bigger than simpler programs (e.g. business systems).

Software Size is the #1 Cost Driver according to Scholarly Literature.
Software Sizing - Methods

• **Popular Methods for measuring size include:**

<table>
<thead>
<tr>
<th>Method</th>
<th>Acronym</th>
</tr>
</thead>
<tbody>
<tr>
<td>COSMIC</td>
<td>COSMIC</td>
</tr>
<tr>
<td>Function Points Analysis</td>
<td>FPA</td>
</tr>
<tr>
<td>Source Lines of Code</td>
<td>SLOC</td>
</tr>
<tr>
<td>Story Points</td>
<td>SP</td>
</tr>
<tr>
<td>Reports, Interfaces, Conversions, Extensions, and Forms/Workflows</td>
<td>RICEF/W</td>
</tr>
<tr>
<td>Functional Requirements</td>
<td>REQ</td>
</tr>
</tbody>
</table>
Before choosing a size method one must take into account whether the source documents (and associated artifacts) needed for the size method are available to estimators.

### Source Documents

<table>
<thead>
<tr>
<th>Size Method</th>
<th>Requirements Traceability Matrix</th>
<th>Software Requirements Specification</th>
<th>Software Requirements Document</th>
<th>Software Architecture Design Document</th>
<th>Contractor Software Data Reports</th>
<th>Agile Project Management Tool (e.g. JIRA)</th>
<th>Code Counter (e.g. USC Code Counter)</th>
</tr>
</thead>
<tbody>
<tr>
<td>COSMIC</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td></td>
</tr>
<tr>
<td>Function Points</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td></td>
</tr>
<tr>
<td>SLOC</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td></td>
</tr>
<tr>
<td>Story Points</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td></td>
</tr>
<tr>
<td>RICEF/W</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td></td>
</tr>
<tr>
<td>Use Case Points</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td></td>
</tr>
<tr>
<td>User Requirements</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td></td>
</tr>
</tbody>
</table>

### Available Artifacts

- Initial Functional Requirements
- Revised Functional Requirements, Data Movement
- Interim Functional Requirements, Data Attributes Data Movement
- Final Functional Requirements, Data Attributes Data Movement
- UML Diagrams
- Story Points User Stories
- Sprint Burn-Down Epics
- New/Reused SLOC
- Logical/Physical SLOC

### Document Availability

- Pre-Contract Award
- Post Contract Award
- Preliminary Design Review
- Critical Design Review
- Contract Start through End
- Initial Operational Capability
- Initial Operational Capability
In addition, when choosing a size method one must consider the following questions of applicability:

- Are the rules for the sizing method rigorously defined in a widely accepted format?
- Are statistical data available to support claims for the consistency of counting by certified counters?
- Do you have funding to hire certified counter (or credible expert) for the sizing method?
- Are benchmarking productivity data (e.g. SLOC/hours) for the size method available to the estimators?
- Are parametric cost models (custom built or proprietary) for the sizing method available to the estimators?
- Percent of reuse and modification be differentiated from total count?
# Software Sizing – Comparison

<table>
<thead>
<tr>
<th>Sizing method</th>
<th>Advantage</th>
<th>Disadvantage</th>
<th>Application</th>
</tr>
</thead>
<tbody>
<tr>
<td>COSMIC</td>
<td>• All data movement have same value</td>
<td>• Benchmarking data is limited</td>
<td>• Real Time Embedded</td>
</tr>
<tr>
<td></td>
<td>• Sizing does NOT depend on data attributes</td>
<td>• No automated counting tool</td>
<td>• Automated Information Systems</td>
</tr>
<tr>
<td></td>
<td>• Many types of data sources may be used</td>
<td>• Only be approximated before contract award as data movements are ill-defined</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Accurately estimated at post contract award</td>
<td>• Requires a certified counter</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Sizing is linked to functional requirements</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Function Point</td>
<td>• Benchmarking data is available</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Analysis</td>
<td>• Many types of data sources may be used</td>
<td></td>
<td>• Automated Information Systems</td>
</tr>
<tr>
<td></td>
<td>• Sizing is linked to functional requirements</td>
<td></td>
<td>• Real-Time-Embedded (small)</td>
</tr>
<tr>
<td></td>
<td>• Accurately estimated at construction phase</td>
<td></td>
<td>• COTS software Integration</td>
</tr>
<tr>
<td>SLOC</td>
<td>• Benchmarking data is available</td>
<td>• No automated counting tool</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Accurate counting via automated code counter at contract completion</td>
<td>• Only be approximated before contract award as data attributes not available</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Sizing by analogy appropriate when projects are preceded by similar ones, same developer</td>
<td>• Counting involves subjectivity</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Requires a certified counter</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Underestimate algorithmic-intensive systems</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Very difficult for large scale systems</td>
<td></td>
</tr>
</tbody>
</table>

COSMIC and Function Points have a controlling body for internationally standardizing the counting rules.

SLOC can be automatically counted via Code Count Tools (e.g. USC Code Counter).
# Software Sizing – Comparison

<table>
<thead>
<tr>
<th>Sizing method</th>
<th>Advantage</th>
<th>Disadvantage</th>
<th>Application</th>
</tr>
</thead>
</table>
| RICEF/W Objects     | • Represents those objects within the COTS tool (e.g. SAP, Oracle, PeopleSoft) that requires customization as these did not meet user requirements | • No standards for counting  
• Needs to be adjusted for complexity  
• Benchmarking data is limited  
• Only be approximated before contract award as COTS tool selection is unknown | • Enterprise Resource Planning Only |
| Story Points        | • Sizing can be obtained from Agile Program Management Tools after contract award  
• Sizing by analogy appropriate when projects are proceeded by previous release/sprint by same developer/team | • No standards for counting  
• Counting involves subjectivity  
• Sizing not available before contract award  
• Cannot used benchmarking data from different developers, projects, or teams  
• Not appropriate for large scale systems | • Any application domain using Agile methods (e.g. SCRUM) |
| Functional Requirements | • Many types of data sources may be used  
• Sizing does NOT depend on data attributes  
• Accurately counted via Dynamic Object-Oriented System (DOORS) at contract end  
• Available at early phase | • Often poorly defined and inconsistently reported across programs at early phase.  
• No standards for counting  
• Benchmarking data available  
• All functional requirements have the same value, not adjusted for complexity | • Real Time Embedded  
• Automated Information Systems |

**Functional Requirements available at early phase and applicable to traditional and agile methods**

**Story Points appropriate only when projects are preceded by a previous release/sprint by same team**
Software Reuse

- Actual software reuse incurs additional costs for the integration effort
- Managers typically overestimate the ability to re-use software from a previously completed effort into their new program

![Actual vs Planned Software Reuse](chart.png)

- Analysts should consider using historical data to adjust the expected percent reuse to lessen overestimation

Wilson Rosa, “Software Reuse Trends in DoD”, White Paper for Principal Deputy Assistant Secretary of Navy (FM&C), 28 October 2015
Size Growth

• A recent study revealed defense software projects underestimated size by an average of 78%

• Adjusting 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.

• Understanding that software will usually grow, and accounting for it by using historic data, will result in more accurate software sizing estimates (GAO-09-3SP)
Size Growth – Best Practices

• Adjust your **Estimated Size for Growth** when your parametric formula or productivity benchmark has been derived using Final Size vs Final Effort
• Chose your **Size Growth** adjustment factors according to program’s maturity and application domain
• Growth adjustments should be made before performing cost/risk and uncertainty analysis
Application Domain

- Software Complexity is influenced by its **Application Domain**

- **Application Domains** are groups of software capabilities that are environment independent, technology driven, characterized by:

  1. Required software reliability
  2. Database size
  3. Product complexity
  4. Integration complexity
  5. Real-time operating requirements
  6. Platform volatility
  7. Target system volatility
  8. Special display requirements
  9. Development re-hosting
  10. Quality assurance requirements
  11. Security requirements
  12. Assurance requirements
  13. Required testing level

- **Application Domain List and Definitions vary by source:**

  ![SLIM-Estimate™](SLIM-Estimate.png)
  ![SEER®](SEER.png)
  ![COCOMO](COCOMO.png)
  ![ISBSG](ISBSG.png)

- **When you build your benchmarks by application domain, you have captured most of the variance in effort addressed by popular parametric cost models like COCOMO, SEER-SEM, True S, and SLIM**
## Application Domain Comparison

<table>
<thead>
<tr>
<th>Department of Defense (17)</th>
<th>SEER-SEM (37)</th>
<th>COCOMO II (12)</th>
<th>SLIM (9)</th>
<th>ISBSG (42)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Microcode</td>
<td></td>
<td></td>
<td>• Microcode and firmware</td>
<td></td>
</tr>
<tr>
<td>Vehicle Payload</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Signal Processing</td>
<td>• Signal Processing</td>
<td>• Signal Processing</td>
<td></td>
<td>• Transportation control (includes avionics, signalling)</td>
</tr>
<tr>
<td>Vehicle Control</td>
<td>• Flight Systems</td>
<td>• OS/Executive</td>
<td>• Avionic</td>
<td></td>
</tr>
<tr>
<td>Real-Time Embedded</td>
<td>• Radar</td>
<td>• Embedded Electronics/Appliance</td>
<td>• Real Time</td>
<td>• Robot control</td>
</tr>
<tr>
<td></td>
<td>• Robotics</td>
<td>• GUI (cockpit displays)</td>
<td></td>
<td>• Embedded software for simple device control</td>
</tr>
<tr>
<td>Command/Control</td>
<td>• Command and Control</td>
<td>• Command and Control</td>
<td>• Command and Control</td>
<td>• Command &amp; control system</td>
</tr>
<tr>
<td>Telecommunications</td>
<td>• Communications</td>
<td>• Message Switching</td>
<td>• Communications</td>
<td>• Telecom &amp; network management</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>• Image, video or sound processing</td>
</tr>
<tr>
<td>Test Software</td>
<td>• Diagnostics</td>
<td>• Testing Software</td>
<td>• Diagnostics</td>
<td>• Fault Tolerance</td>
</tr>
<tr>
<td>System</td>
<td>• Device Driver</td>
<td>• System &amp; Device Utilities</td>
<td>• Operating System</td>
<td>• Device or interface driver</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Utilities</td>
<td>• Operating system or software utility</td>
</tr>
<tr>
<td>Process Control</td>
<td>• Process Control</td>
<td>• Process Control</td>
<td>• Process Control</td>
<td>• Complex process control</td>
</tr>
<tr>
<td>Scientific</td>
<td>• Graphics</td>
<td>• Engineering and Science</td>
<td>• Scientific</td>
<td>• Artificial Intelligence</td>
</tr>
<tr>
<td></td>
<td>• Expert System</td>
<td>• Simulation</td>
<td></td>
<td>• Mathematical modelling</td>
</tr>
<tr>
<td></td>
<td>• Artificial Intelligence</td>
<td></td>
<td></td>
<td>• Scientific/ engineering application</td>
</tr>
<tr>
<td></td>
<td>• Math &amp; Complex Algorithms</td>
<td></td>
<td></td>
<td>• Statistical analysis</td>
</tr>
<tr>
<td></td>
<td>• Simulation</td>
<td></td>
<td></td>
<td>• 3D modelling or animation</td>
</tr>
</tbody>
</table>

Table above allows you to crosscheck and compare estimates across parametric cost models
### Application Domain Comparison (cont.)

#### Department of Defense (17)  SEER (37)  COCOMO II (12)  SLIM (9)  ISBSG (42)

<table>
<thead>
<tr>
<th>Department of Defense (17)</th>
<th>SEER (37)</th>
<th>COCOMO II (12)</th>
<th>SLIM (9)</th>
<th>ISBSG (42)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mission Planning</td>
<td>Mission Planning &amp; Analysis</td>
<td>Management Information System</td>
<td>Business</td>
<td></td>
</tr>
<tr>
<td>Custom AIS Software</td>
<td>Database, Data Mining, Data Warehousing, Financial Transactions, GUI Management Information System Multimedia, Relational Database Object-Oriented Database, Transaction Processing Internet Server Applet Report Generation Office Automation</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Enterprise Information System</td>
<td>Business Information System</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Enterprise Service</td>
<td>Business Information System</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Training</td>
<td>Training, CBT, CAI</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Software Tools</td>
<td>Software Development Tools Business Analysis Tool, CAD Software Tools</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Software Tools</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- You may bin your dataset by “super domain” when # projects is limited for productivity benchmarking

---

© ICEAA
Effort Estimation

• Once the software sizing is complete, it can be converted into software development effort using formulas below

**Parametric Formula**

\[ Effort_{DEV} = A \times Size^B \times EM \]

Or

\[ Effort_{DEV} = A \times Size \]

**Simple Formula**

\[ Effort_{DEV} = SIZE \times (PROD)^{-1} \]

**Productivity Benchmark**

\[ PROD = \frac{Output}{Inputs} = \frac{SIZE}{Effort} \]

• Preferred formula for Parametric Method
• Most appropriate when there is enough data to build a regression effort model
• An estimated value of \( B < 1.0 \) indicates an economy of scale. An estimated value of \( B > 1 \) indicates a diseconomy of scale
• Some parametric cost models may include Effort Multipliers (EM) to account for impact of application domain (AD), development environment and/or team capability

• Preferred formula for Analogy, Engineering-Build-Up, and Extrapolation from Actual Methods
• Most appropriate when there is not enough data to build a parametric model or when it is difficult to assess team capability or development environment
• Productivity (PROD) benchmark from actual data or from analogous program(s) by application domain
Example of Parametric Formula 1

Effort = 2.047 x KESLOC^{0.9288} x AD

<table>
<thead>
<tr>
<th>Parametric Formula</th>
<th>N</th>
<th>R² %</th>
<th>CV (%)</th>
<th>Mean</th>
<th>KESLOC Min</th>
<th>KESLOC Max</th>
</tr>
</thead>
<tbody>
<tr>
<td>Effort = 2.047 x KESLOC^{0.9288} x AD</td>
<td>317</td>
<td>89</td>
<td>33</td>
<td>431</td>
<td>1</td>
<td>842</td>
</tr>
</tbody>
</table>

**Effort** =  **Effort in Person Month**

**KESLOC** =  **Thousand Equivalent Source Lines of Code (SLOC)**

**AD** =  **Application Domain multiplier from lookup table below:**

<table>
<thead>
<tr>
<th>Application Domain</th>
<th>Multiplier</th>
<th>T-stat</th>
<th>VIF</th>
</tr>
</thead>
<tbody>
<tr>
<td>Software Tools</td>
<td>1.000</td>
<td>***</td>
<td>***</td>
</tr>
<tr>
<td>Automated Information System</td>
<td>1.917</td>
<td>4.0</td>
<td>3.3</td>
</tr>
<tr>
<td>Mission Planning</td>
<td>2.209</td>
<td>3.1</td>
<td>2.6</td>
</tr>
<tr>
<td>Scientific</td>
<td>3.068</td>
<td>5.6</td>
<td>3.2</td>
</tr>
<tr>
<td>System</td>
<td>3.072</td>
<td>5.8</td>
<td>3.8</td>
</tr>
<tr>
<td>Telecommunication</td>
<td>3.434</td>
<td>6.7</td>
<td>6.0</td>
</tr>
<tr>
<td>Test</td>
<td>4.521</td>
<td>6.8</td>
<td>2.3</td>
</tr>
<tr>
<td>Real Time Embedded</td>
<td>4.801</td>
<td>8.7</td>
<td>6.7</td>
</tr>
<tr>
<td>Command and Control</td>
<td>4.935</td>
<td>8.5</td>
<td>4.7</td>
</tr>
<tr>
<td>Vehicle Control</td>
<td>5.903</td>
<td>9.2</td>
<td>4.0</td>
</tr>
<tr>
<td>Vehicle Payload</td>
<td>7.434</td>
<td>9.8</td>
<td>3.1</td>
</tr>
<tr>
<td>Signal Processing</td>
<td>10.72</td>
<td>12.7</td>
<td>5.0</td>
</tr>
</tbody>
</table>

## Example of Parametric Formula 2

<table>
<thead>
<tr>
<th>Application Domain</th>
<th>Parametric Formula</th>
<th>METRICS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Enterprise Information System</td>
<td>Effort = 922.2 * RICEF/W</td>
<td>R²</td>
</tr>
<tr>
<td></td>
<td></td>
<td>83%</td>
</tr>
</tbody>
</table>

**Effort** = Effort in Labor Hours for configuring SAP/Oracle COTS Tool

**RICEF/W** = Combined number of reports, interfaces, conversions, extensions, forms/workflow configured

### Application:
- Includes supply chain and finance Enterprise Resource Planning (ERP)
- Captures assessment and configuration effort
- Dataset only captures Government Sector ERPs, SAP/ORACLE/MOMENTUM Tools
- Reports may include forms, and extensions may include scripts, workflows, and bolt-on

---


© ICEAA
### Example of Productivity (PROD) Benchmarks

<table>
<thead>
<tr>
<th>Application type</th>
<th>ESLOC/PM</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>25th Percentile</td>
</tr>
<tr>
<td>Mission Planning</td>
<td>207</td>
</tr>
<tr>
<td>Automated Information System</td>
<td>292</td>
</tr>
<tr>
<td>System Software</td>
<td>168</td>
</tr>
<tr>
<td>Scientific</td>
<td>129</td>
</tr>
<tr>
<td>Telecommunications</td>
<td>140</td>
</tr>
<tr>
<td>Real-Time Embedded</td>
<td>84</td>
</tr>
<tr>
<td>Command and Control</td>
<td>103</td>
</tr>
<tr>
<td>Vehicle Control</td>
<td>70</td>
</tr>
<tr>
<td>Vehicle Payload</td>
<td>43</td>
</tr>
<tr>
<td>Signal Processing</td>
<td>40</td>
</tr>
</tbody>
</table>

\[
\text{PROD} = \frac{\text{SIZE}}{\text{Effort}} = \frac{\text{ESLOC}}{\text{PM}}
\]

Where:

- **ESLOC** = Equivalent Source Lines of Code (SLOC)
- **PM** = Effort in Person Month

---

Course Outline

• Core Knowledge
  • Software Cost Drivers
    • Estimating Methods
• Summary
• Related and Advanced Topics
  • Software Development Life Cycles
  • Source Lines of Code Sizing Best Practices
Effort Estimation Methods

- Engineering Build-Up
- Extrapolation from Actual
- Analogy
- Parametric Method – Commercial
- Parametric Method – Custom
Appropriate Estimating Method According to Program Maturity

PDR = preliminary design review; CDR = critical design review; IOC = initial operational capability
Engineering Build-Up Method

- Cost of each component is estimated by individuals, often responsible for developing the software component or very familiar with design.
  - Typically done by System Developers as they have the time, data, and technical expertise.

- Best suited for efforts at Post-Critical Design Review phase when detailed software design documents or artifacts are available.

<table>
<thead>
<tr>
<th>Method</th>
<th>Strength</th>
<th>Weakness</th>
<th>Application</th>
</tr>
</thead>
<tbody>
<tr>
<td>Engineering Build-Up</td>
<td>• Sensitive to labor Rates</td>
<td>• Slow and laborious</td>
<td>• Story Point Sizing</td>
</tr>
<tr>
<td></td>
<td>• Sensitive to complexity</td>
<td>• Tend to underestimate cost due to cognitive biases and organizational pressure.</td>
<td>• Function Points Analysis</td>
</tr>
<tr>
<td></td>
<td>• Estimation errors in the various components have a chance to balance out</td>
<td>• Requires detailed knowledge of the software architecture</td>
<td>• Contract Cost Proposals</td>
</tr>
<tr>
<td></td>
<td>• Easily audited</td>
<td></td>
<td>• COSMIC Sizing</td>
</tr>
</tbody>
</table>

© ICEAA
Extrapolation from Actuals Method

- Best suited for follow-on software upgrades, releases or deliveries; or when you have existing data from the current program

- Guidelines

Compute Productivity from Current Program’s History:

\[ \text{PROD}_{\text{History}} = \frac{\text{SIZE}_{\text{History}}}{\text{PM}_{\text{History}}} \]

Compute Effort for Follow-on Build using Program’s Historical Productivity and the Simple Model:

\[ \text{Effort}_{\text{Follow-on}} = \frac{\text{SIZE}_{\text{Follow-on}}}{\text{PROD}_{\text{History}}} \]

To be considered “Actual”, data must satisfy the following criteria:
- Software component successfully tested and delivered (Post-Development Test)
- Same Program, Contractor, Scope and Application Domain, and similar Size
Extrapolation from Actuals Method

Exercise and Result

• Suppose you are asked to compute the total effort for Project A Build 4, using actual data from Project A Build 3 and the Simple Model

• Given:

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Actual</th>
<th>Estimate</th>
</tr>
</thead>
<tbody>
<tr>
<td>Project Name:</td>
<td>A</td>
<td>A</td>
</tr>
<tr>
<td>Build</td>
<td>3</td>
<td>4</td>
</tr>
<tr>
<td>Company</td>
<td>XYZ</td>
<td>XYZ</td>
</tr>
<tr>
<td>Application Domain</td>
<td>Simulation</td>
<td>Simulation</td>
</tr>
<tr>
<td>Effective SLOC (ESLOC)</td>
<td>45,000</td>
<td>65,000</td>
</tr>
<tr>
<td>Person-Month (PM)</td>
<td>600</td>
<td>??</td>
</tr>
</tbody>
</table>

• Calculation

  • Step 1: Calculate Productivity for Project A Build 3

  \[ PROD_{\text{Build}3} = \frac{\text{ESLOC}_{\text{Build}3}}{\text{PM}_{\text{Build}3}} = \frac{45000}{600} = \frac{75 \cdot \text{ESLOC}}{\text{PM}} \]

  • Step 2: Calculate Effort for Project A Build 4 using Build 3 Productivity and Simple Model

  \[ \text{Effort}_{\text{Build}4} = \frac{\text{SIZE}_{\text{Build}4}}{\text{PROD}_{\text{Build}3}} = \frac{65000}{75} = 867 \text{ PM} \]
Analogy Method

• Best suited for programs in early life cycle phase when:
  • Not enough data exists from a number of similar systems, but you have cost data from a single similar system

• Guidelines:
  Compute Productivity from Analogous System

\[
PROD_{\text{Analogous}} = \frac{\text{SIZE}}{\text{Effort}}
\]

• Compute Effort for New System using Analogous System’s Productivity and the Linear Model

\[
\text{Effort}_{\text{NewSystem}} = \frac{\text{SIZE}_{\text{NewSystem}}}{PROD_{\text{Analogous}}}
\]

Best results are achieved when similarities between old and new systems are high:
  • Platform, Complexity (Productivity Type Mix), Size, Amount of Reuse…
**Analogy Method**

**Exercise - Result**

- **Compute the total effort for New System, using actual data from Analogous System and the Simple Model**

- **Given:**

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Analogous System</th>
<th>New System</th>
</tr>
</thead>
<tbody>
<tr>
<td>Program Name:</td>
<td>Radar 3rd GEN</td>
<td>Radar 4th GEN</td>
</tr>
<tr>
<td>Contractor:</td>
<td>Ktr A</td>
<td>Ktr B</td>
</tr>
<tr>
<td>Productivity Type(s)</td>
<td>SCP, MP, RTE</td>
<td>SCP, MP, RTE</td>
</tr>
<tr>
<td>WBS Element</td>
<td>Communication</td>
<td>Communication</td>
</tr>
<tr>
<td>ESLOC:</td>
<td>186000</td>
<td>200000</td>
</tr>
<tr>
<td>Effort (PM=Person-Months)</td>
<td>4038</td>
<td>??</td>
</tr>
</tbody>
</table>

- **Step 1: Calculate Productivity for Analogous System**

  \[
  \text{PROD}_{\text{Analogy}} = \frac{\text{SIZE}}{\text{Effort}} = \frac{\text{ESLOC}}{\text{PM}} = \frac{186000}{4038} = \frac{46 \times \text{ESLOC}}{\text{PM}}
  \]

- **Step 2: Calculate Effort for New System using Analogous System’s Productivity and the Software Simple Model**

  \[
  \text{Effort}_{\text{New System}} = \frac{\text{SIZE}_{\text{New System}}}{\text{PROD}_{\text{Analogy}}} = \frac{\text{ESLOC}_{\text{New System}}}{\text{PROD}_{\text{Analogy}}} = \frac{200000}{46} = 4342 \text{ PM}
  \]
• Appropriate when you don’t have historical data relevant to the system being estimated

**Parametric Method Commercial**

<table>
<thead>
<tr>
<th>Method</th>
<th>Strength</th>
<th>Weakness</th>
<th>Application</th>
</tr>
</thead>
<tbody>
<tr>
<td>Parametric Method - Commercial</td>
<td>• Some can be used effectively if calibrated properly with real data</td>
<td>• No insight to dataset, calibration and normalization</td>
<td>• Crosscheck</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Secondary Method</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Design-to-cost tradeoffs</td>
</tr>
</tbody>
</table>

**True S® by PRICE Systems**

**SLIM-Estimate™**
**Parametric Method**

**Custom**

- Best suited when you can build your own model from a number of programs relevant to the system being estimated

<table>
<thead>
<tr>
<th>Method</th>
<th>Strength</th>
<th>Weakness</th>
<th>Application</th>
</tr>
</thead>
<tbody>
<tr>
<td>Parametric Method - Commercial</td>
<td>- Statistical inferences can be made with fewer parameters</td>
<td>- Data can be very difficult and time-consuming to collect, normalize, and analyze</td>
<td>- Crosscheck</td>
</tr>
<tr>
<td></td>
<td>- Suitable for early milestone phases</td>
<td></td>
<td>- Primary Method when data is available for regression model</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- Design-to-cost tradeoffs</td>
</tr>
</tbody>
</table>
Course Outline

• Core Knowledge
  • Software Cost Drivers
  • Estimating Methods

• Summary

• Related and Advanced Topics
  • Software Development Life Cycles
  • Source Lines of Code Sizing Best Practices
Summary

• Software size and application domain are the two major contributors to software development cost

• Functional Requirements, COSMIC, and Function Points can be obtained or derived from several data sources at early phase

• Story Points are typical available after completing first delivery

• Estimated software size should be adjusted for growth based on historical data or open-source benchmarks

• Analysts should consider using historical data or benchmarks to adjust the expected percent reuse to lessen overestimation

• Analogy-Based Method best suited for programs in early cycle phase when not enough data exists from similar systems.

• Commercial Parametric Cost Models are appropriate when you don’t have historical data relevant to the system being estimated
Course Outline

• Core Knowledge
  • Software Cost Drivers
  • Estimating Methods
• Summary
• Related and Advanced Topics
  • Software Development Life Cycles
  • Source Lines of Code Sizing Best Practices
Software Life Cycle Models

- Waterfall
- Incremental
- Evolutionary
- Spiral

- COTS Integration
- Reuse
- Rehost/Port
- Refactoring/re-engineering
- Maintenance

All of these are accommodated in the Incremental Commitment Spiral Model (ICSM)

© ICEAA
The Incremental Commitment Spiral Process: Phased View

**Anchor Point Milestones**

**Stage I: Incremental Definition**
- ECR/MDP: Exploration / Needs and Opportunities
- VCR/MDD: Valuation / Materiel Solution Analysis & AoA
- FCR/A: Foundations / Technology Development (TD) & CDD

**Stage II: Incremental Development and Operations**
- DCR₁, DCR₂/B₁, DCR₂/B₂: Development, Engineering and Manufacturing Development (EMD), TD, & CDD
- OCR₁, OCR₂, OCR₃: Operations, Production, Development, EMD, TD & CDD

**Activities**
- Concurrent risk- and opportunity-driven growth of system understanding and definition
  - Initial scoping
  - Concept definition
  - Investment analysis
  - System life-cycle architecture and ops concept
  - Build-to increment plans and specifications
  - NDI, outsource partner selections

**Risk patterns determine life cycle process**
- Adjust scope, priorities, or discontinue

**Synchronize, stabilize concurrency via FEDs**


© ICEAA
Software Development Approaches

- Architectured agile
- Agile/DevOps/Continuous delivery
- Plan-driven
- Formal methods
- COTS/services

Any of these software development approaches can be used with most any lifecycle models presented earlier...

A given lifecycle model can employ more than one software development approach...

Teams may change development approaches based on maturity of architecture or new features to be added to software system...
Ways to “Develop” Code

• Custom written (3GL software)
• Tool-generated software (4GL software)
• Content management frameworks
• Reuse
  • “As is”
  • With technical debt remediation
  • Refactoring/re-engineering
  • Rehost/Port
  • COTS/services
Risk-Driven Scalable Spiral Model: Simple Increment View For Each Level of System-of-Interest

Unforeseeable Change (Adapt)

Rapid Change

Unforeseeable Change (Adapt)

Future Increment Baselines

AGILE

Rebaselining for Future Increments

Deferrals

Short, Stabilized Development of Increment N

Increment N Transition/Operations and Maintenance

Artifacts

Concerns

Verification and Validation (V&V) of Increment N

Future V&V Resources

Increment N Baseline

Stable Development Increments

Rapid Change

Foreseeable Change (Plan)

Short Development Increments

High Assurance

Stable Development Increments

Continuous V&V

Current V&V Resources

Future V&V Resources

Reality for Large/Complex Development: Multiple Subsystems Use Different Development Approaches*

Course Outline

• Core Knowledge
  • Software Cost Drivers
  • Estimating Methods
• Summary
• Related and Advanced Topics
  • Software Development Life Cycles
  • Source Lines of Code Sizing Best Practices
Source Lines of Code (SLOC)

• Considers the volume of code required to develop the software; includes executable instructions and data declarations and normally excludes comments and blanks.

• Estimation is by analogy, engineering expertise, or automated code counters. SLOC sizing is particularly appropriate for projects preceded by similar ones.

• SLOC are logical source statements consisting of executables, data declarations, and compiler directives.

<table>
<thead>
<tr>
<th>Statement Type</th>
<th>Includes</th>
<th>Excludes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Executable</td>
<td></td>
<td>✓</td>
</tr>
<tr>
<td>Nonexecutable</td>
<td>✓</td>
<td></td>
</tr>
<tr>
<td>Declarations</td>
<td>✓</td>
<td></td>
</tr>
<tr>
<td>Compiler directives</td>
<td>✓</td>
<td></td>
</tr>
<tr>
<td>Comments and blank lines</td>
<td></td>
<td>✓</td>
</tr>
</tbody>
</table>
SLOC – Sizing Best Practices

• It is desirable to have consistent size definitions and measurements across different models and estimates

• Generally Accepted Rules for reporting Software Size:
  • Delivered Size must be split by Size Type
  • Size Count must be reported in Logical SLOC
  • Baseline SLOC is converted into Equivalent SLOC (ESLOC)

Mainstream parametric models (COCOMO, SEER, SLIM, PRICE) use these rules
## Size Type

<table>
<thead>
<tr>
<th>Size Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Delivered</td>
<td>Sum of New and Adapted Size</td>
</tr>
<tr>
<td>New</td>
<td>Original software created for the first time.</td>
</tr>
<tr>
<td>Adapted</td>
<td>Sum of Reused, Modified, Generated, and Converted</td>
</tr>
<tr>
<td>Reused</td>
<td>Pre-existing software that is not changed with the adaption parameter settings:</td>
</tr>
<tr>
<td></td>
<td>• Design Modification % (DM) = 0%</td>
</tr>
<tr>
<td></td>
<td>• Code Modification % (CM) = 0%</td>
</tr>
<tr>
<td>Modified</td>
<td>Pre-existing software that is modified for use by making design, code and/or test changes:</td>
</tr>
<tr>
<td></td>
<td>• Code Modification % (CM) &gt; 0%</td>
</tr>
<tr>
<td>Generated</td>
<td>Software created with automated source code generators.</td>
</tr>
<tr>
<td>Converted</td>
<td>Software that is converted between languages using automated translators.</td>
</tr>
<tr>
<td>Commercial Off-The-Shelf Software (COTS)</td>
<td>Pre-built commercially available software components. The source code is not available to application developers. It is not included for equivalent size. However, the effort associated with COTS Integration must not be overlooked. Use Number of COTS components as your primary size measure (see Advanced Topics for guidelines)</td>
</tr>
</tbody>
</table>

**Generally Accepted Best Practices:**
1. When collecting or analyzing data make sure your delivered SLOC is split between New, Reused, Modified, Generated, and Converted.
2. Don’t include COTS software in your Delivered Source Lines of Code
• Equivalent Source Lines of Code (ESLOC)
• An adjustment of total delivered size to reflect the actual degree of work required
• It adjusts reused size? Or software size? relative to the effort of developing it all new
• Adjustment based on % re-design, re-code, re-test
• There are various methods to compute equivalent size, some will be discussed later in this lesson

If given ESLOC as an input, you must find out how it was computed!
ESLOC, General Formula

ESLOC is an adjustment of total size to reflect the actual degree of work involved:

Formula:

\[
ESLOC = \text{New SLOC} + \text{Modified SLOC}^{*}\text{AAF}_M + \text{Reused SLOC}^{*}\text{AAF}_R + \text{Generated SLOC}^{*}\text{AAF}_G + \text{Converted SLOC}^{*}\text{AAF}_C
\]

Where:

\[
\text{AAF}_i = 0.4*\text{DM} + 0.3*\text{CM} + 0.3*\text{IM}
\]

And:

<table>
<thead>
<tr>
<th>AAF</th>
<th>= Adaptation Adjustment Factor</th>
</tr>
</thead>
<tbody>
<tr>
<td>i</td>
<td>Refers to the size type: Modified (M), Reuse (N), Generated (R), Converted (C)</td>
</tr>
<tr>
<td>DM</td>
<td>Design Modified (DM), also known as re-design</td>
</tr>
<tr>
<td>CM</td>
<td>Code Modified (CM), also known as re-code</td>
</tr>
<tr>
<td>IM</td>
<td>Integration Modified (IM), also known as re-test</td>
</tr>
</tbody>
</table>

Formula adapted from COCOMO II with two key differences:
1. Splits Adapted SLOC between Modified, Reuse, Generated, and Converted
2. Specifies an AAF for each size type
## What to include in ESLOC Calculation by Size Type:

<table>
<thead>
<tr>
<th>Size Type</th>
<th>Includes</th>
<th>Excludes</th>
</tr>
</thead>
<tbody>
<tr>
<td>New</td>
<td>✓</td>
<td></td>
</tr>
<tr>
<td>Reused</td>
<td>✓</td>
<td></td>
</tr>
<tr>
<td>Modified</td>
<td>✓</td>
<td></td>
</tr>
<tr>
<td>Generated</td>
<td>✓</td>
<td></td>
</tr>
<tr>
<td>Converted</td>
<td>✓</td>
<td></td>
</tr>
<tr>
<td>COTS</td>
<td></td>
<td>✓</td>
</tr>
</tbody>
</table>

Not counting COTS in ESLOC is considered best practice by Air Force

COTS Integration should be estimated separately using a distinct approach; see Modern Estimation Challenges
**ESLOC Exercise - Result**

- Suppose that the code sizes given are New, Modified, and Reused Code
- Find the ESLOC given:

<table>
<thead>
<tr>
<th>Size Type</th>
<th>SLOC</th>
<th>DM</th>
<th>CM</th>
<th>IM</th>
</tr>
</thead>
<tbody>
<tr>
<td>New SLOC (NSLOC)</td>
<td>3500</td>
<td>n/a</td>
<td>n/a</td>
<td>n/a</td>
</tr>
<tr>
<td>Modified SLOC (MSLOC)</td>
<td>4400</td>
<td>0%</td>
<td>15%</td>
<td>100%</td>
</tr>
<tr>
<td>Reused SLOC (RSLOC)</td>
<td>6500</td>
<td>0%</td>
<td>0%</td>
<td>15%</td>
</tr>
</tbody>
</table>

**Formula:**

\[
\text{ESLOC} = \text{NSLOC} + (\text{AAF}_M \times \text{MSLOC}) + (\text{AAF}_R \times \text{RSLOC})
\]

**Calculations:**

\[
\begin{align*}
\text{NSLOC} & = 3500 \\
\text{AAF}_M \times \text{MSLOC} & = 4,400 \times \left[(0.4\times 0\%) + (0.3\times 15\%) + (0.3\times 100\%)\right] = 1518 \\
\text{AAF}_R \times \text{RSLOC} & = 6,500 \times \left[(0.4\times 0\%) + (0.3\times 0\%) + (0.3\times 15\%)\right] = 292.5
\end{align*}
\]

Total ESLOC = 3500 + 1518 + 292.5 = 5310.5