Refining Software Development Estimation Techniques for the Federal Aviation Administration En Route Systems Acquisition

Presenter:
Jeffrey O’Leary
En Route Software Acquisition and Development Manager
Federal Aviation Administration
(202) 366-5942
jeff.oleary@faa.gov

January 28, 2003
Refining Software Development Estimation Techniques

• Jeffrey O’Leary, Federal Aviation Administration
• A. Winsor Brown, USC Center for Software Engineering
• Mike Liggan, MITRE CAASD
• Martin Merlo, Northrop Grumman
• Alok Srivastava, Northrop Grumman
• Robert Leonard, Northrop Grumman
Goals of Presentation

• Our approach to “Refining Software Development Estimation Techniques”
• Overview of FAA SIS, the “En Route” Air Traffic Control Domain and En Route Automation Modernization (ERAM) program
• FAA software estimation problems and needs
• Function Point Methodology and our efforts to adopt and apply it
• Our experience developing an historical database and metrics
• Current tailoring, application and plans for use of COCOMO II
Outline

• Overview
• Function Point Methodology
• Developing Historical Database
• Application of COCOMO II
• Concluding Remarks
The FAA’s Job

Each day, manage 30,000 commercial flights to safely move 2,000,000 passengers

- ~ 500 FAA Managed Air Traffic Control Towers
- ~ 180 Terminal Radar Control Centers
- 20 Enroute Centers
- ~ 60 Flight Service Stations
- ~ 40,000 Radars, NAVAIDs, Radios, etc.
Domestic Air Traffic Control

Stakeholders & Users

Flight Data Specialists

Quality Assurance

Weather Service

Traffic Flow Management

Airspace & Procedures

System Operators

System Maintainers

Airlines, Pilots, GA, Military
En Route Automation Systems

- Consist of a number of software intensive systems

<table>
<thead>
<tr>
<th>System</th>
<th>Type/Primary Language(s)</th>
<th>Size (SLOC 000s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>NAS Host</td>
<td>Mainframe / Jovial, BAL</td>
<td>300</td>
</tr>
<tr>
<td></td>
<td>Mainframe / Custom OS, BAL</td>
<td>400</td>
</tr>
<tr>
<td></td>
<td>Various Support / Jovial, Bal, REXX</td>
<td>472</td>
</tr>
<tr>
<td>Display System</td>
<td>Distributed (UNIX) / Ada83, C</td>
<td>440</td>
</tr>
<tr>
<td></td>
<td>Support Applications/ Ada, C, Fortran</td>
<td>350</td>
</tr>
<tr>
<td>User Request Evaluation Tool</td>
<td>Client/Server (UNIX) / Ada95</td>
<td>105</td>
</tr>
<tr>
<td></td>
<td>Display X-Windows/ Ada83, C</td>
<td>64</td>
</tr>
<tr>
<td></td>
<td>Dist. RT (Unix/Posix)/ Ada95, C</td>
<td>229</td>
</tr>
<tr>
<td></td>
<td>Support Applications/ Ada, C, SQL</td>
<td>222</td>
</tr>
</tbody>
</table>
| DARC, CPDLC, PAMRI                  | Various / C, C++, Jovial, Assy                | 500 +
What is the Problem with the Current Infrastructure Architecture?

• The HOST computer software is 30 years-old
  – Outmoded design for limited memory & processing
  – Rigid structure – closed architecture
  – Failure modes require operator intervention
  – Non-ATC software integrated with core capability
  – Existing code written in obscure languages – JOVIAL/BAL
  – Adaptation evolved to address shortcomings

• Hardware obsolescence is on the horizon
  – Most recent replacement was for Y2K
  – Maintenance on processors currently ends in 2008
  – Porting existing code to new machines does not resolve design limitations
ERAM Infrastructure Acquisition

Modernizes en route automation and infrastructure to provide an open-standards based system that will be the basis for future capabilities and enhancements

- Replaces:
  - Host computer system software/hardware
  - Direct Access Radar Channel (DARC) software/hardware
  - Other associated interfaces, communications and support infrastructure

- Provides:
  - New automation architecture allows future growth and capabilities
  - New capabilities to support flexible routing, new surveillance types and sensors, full capability including safety alerts on back up system

- Attributes:
  - Leverages recent and ongoing SIS developments and deployments – product line evolution
  - Initial Size Est. 1.1 to 1.3 MSLOC with 45 to 55% from NDI/Reuse opportunities (that estimate was derived largely by analogy with “comparable” FAA and non-FAA systems
En Route Domain Evolution

Pre 1998
- Host (3083 FDP/SDP/DYSIM)
- DARC Back up SDP (no safety alerts)
- PVD’s (Controller Displays)
- Display Channels Display Processing
- PAMRI Interfaces
- DAS RSD and Assorted Monitors

Today
- HOST (HOCSR) (HW Refresh 3083-63)
- Peripherals Refresh
- URET/CCLD
- DARC DP Refresh
- DSR New CHI, M&C

Transition
- Academy Training System
- *URET National Deployment
- *URET
- Backup SDP (with safety Alerts)
- *DSR Tech Refresh

2008 & Beyond
- DARC Back up SDP (no safety alerts)
- URET
- Fully Redundant FDP
- Fully Redundant SDP
- Integrated Display Services
- ERAM Monitor and Control
- *ECG
- General Information Processing
- M&C for Non ATC Systems

Legacy
Non ERAM
ERAM

* = Program dependency
Software Estimation Challenge

**Problem:** Federal Agencies are not very good at software estimating!

**Contributors:**

- We are primarily acquirers not developers
  - Lack of resources and SIS estimating experience
- Estimation by analogy is hazardous
  - Language, technology and environment differences
  - Analogous systems, often legacy, are not good matches
- **TOOLs:** COCOMO, SLIM™, SEER™, etc. dependencies
  - Good independent size estimate
  - Valid historical data from comparable systems
  - Best for “Neat” systems not Systems of Systems with COTS, Reuse, and system interdependencies
Product of Lockheed Martin ATM
Software Estimation Challenge

Taking up the challenge – attacking the problem!

• Develop capability in Function Point (FP) Methodology to become proficient at “sizing” the system
• Develop historical database and metrics to calibrate and validate the estimates
• Tailoring and application of COCOMO II and related estimating techniques to develop the estimate
Outline

- Overview
  - Function Point Methodology
    - Developing Historical Database
    - Application of COCOMO II
    - Concluding Remarks
Function Point Methodology

• Developed by Allan J. Albrecht of IBM in the late 1970’s
  – Adopted by the International Standards Organization (ISO) in 1999
  – Researched, refined and managed as a Standard by the International Function Point Users Group (IFPUG), current version 4.1
Function Point Methodology

• How does FPA Work?
  – Calculates the functional size of a system by assigning a weight to each individual function – The sum of weights is called the Unadjusted Function Points (UFP)
  – At the level of the complete system determines a Value Adjustment Factor (VAF) from application characteristics such as processing complexity and transaction rate.
  – The Adjusted Function Point metric (AFP) is the product of the UFP and VAF
  – Can proceed with estimates of effort and schedule directly with various FP tool algorithms or “back-fire” for estimation models using LOC metrics
Function Point Methodology

- Function point estimation is based on the data to be managed by an application and the processes that access and manipulate the data.

![Diagram showing the relationship between Logical Inputs (EIs), Logical Queries (EQs), Internal Logical Files (ILFs), External Logical Files (ELFs), and Logical Outputs (EOs).]
Function Point Methodology

• FPA Function Types
  – Internal Logical File (ILF) – Logically related data/control information maintained within application’s boundary (e.g., data tables, database files).
  – External Logical File (ELF) – Logically related data/control information maintained within the boundary of another application (e.g., shared data files, reference data, fixed messages)
  – External Input (EI) – Process of data/control information from outside the application boundary (e.g., input screens, interactive inputs, batch input streams, HW inputs)
  – External Output (EO) – Sends data/control information outside the application boundary through processing logic other than/in addition to data retrieval (e.g., output screens, batch outputs, printed reports, HW & SW outputs)
  – External Inquiry (EQ) -- Sends data/control information outside the application boundary strictly through data retrieval (e.g., menus, context-sensitive help, embedded inquiries)
Why Use Function Points?

• Function Points are a well-established method to measure the functionality of a system from a user’s perspective
  – Consistent with desire for user orientation
  – Consistent with the level of abstraction in typical source requirements documentation
  – Function points avoid design-specific perspectives that complicate sizing efforts in FAA’s heterogeneous environment
Why Use Function Points?

• Function Points are recognized to be the best overall sizing approach
  – Through 2005, function points will remain the most appropriate measure of application size (0.8 probability)*

FPM Adoption Strategy

- Formed a 10-member Software Estimation and Analysis Team (SEAT)
  - Five days dedicated training in FPM (David Consulting Group)
  - Selected and trained on a counting tool with potential to meet SIS/System of Systems challenges (CHARISMATEK Function Point WORKBENCH)
  - Defined and clarified project scope and defined subsystem application boundaries to partition problem
The Hot-SEAT: Into the Fire

• Assign application count responsibilities
• Perform application counts with System Specification Document, emerging System Segment Specification, other NDI/Reused specifications and documentation
• Review and adjust application counts
• Integrate application counts into a project count
• Iterate based on project evolution and perform selected validation of estimate basis as SRS documents are developed
# Example Component FP Report

## Function Point Summary

<table>
<thead>
<tr>
<th>Function Type</th>
<th>Complexity Rating</th>
<th>Number of Functions</th>
<th>Function Points Awarded</th>
<th>Total Function Points</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>EXTERNAL INPUTS</strong></td>
<td>Low</td>
<td>9</td>
<td>3</td>
<td>27</td>
</tr>
<tr>
<td></td>
<td>Average</td>
<td>3</td>
<td>4</td>
<td>12</td>
</tr>
<tr>
<td></td>
<td>High</td>
<td>1</td>
<td>6</td>
<td>6</td>
</tr>
<tr>
<td></td>
<td><strong>Sub-Total</strong></td>
<td></td>
<td></td>
<td><strong>45</strong></td>
</tr>
<tr>
<td><strong>EXTERNAL OUTPUTS</strong></td>
<td>Low</td>
<td>7</td>
<td>4</td>
<td>28</td>
</tr>
<tr>
<td></td>
<td>Average</td>
<td>3</td>
<td>5</td>
<td>15</td>
</tr>
<tr>
<td></td>
<td>High</td>
<td>0</td>
<td>7</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td><strong>Sub-Total</strong></td>
<td></td>
<td></td>
<td><strong>43</strong></td>
</tr>
<tr>
<td><strong>EXTERNAL INQUIRIES</strong></td>
<td>Low</td>
<td>3</td>
<td>3</td>
<td>9</td>
</tr>
<tr>
<td></td>
<td>Average</td>
<td>4</td>
<td>4</td>
<td>16</td>
</tr>
<tr>
<td></td>
<td>High</td>
<td>1</td>
<td>6</td>
<td>6</td>
</tr>
<tr>
<td></td>
<td><strong>Sub-Total</strong></td>
<td></td>
<td></td>
<td><strong>31</strong></td>
</tr>
<tr>
<td><strong>INTERNAL LOGICAL FILES</strong></td>
<td>Low</td>
<td>0</td>
<td>7</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>Average</td>
<td>11</td>
<td>10</td>
<td>110</td>
</tr>
<tr>
<td></td>
<td>High</td>
<td>0</td>
<td>15</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td><strong>Sub-Total</strong></td>
<td></td>
<td></td>
<td><strong>110</strong></td>
</tr>
<tr>
<td><strong>EXTERNAL INTERFACE FILES</strong></td>
<td>Low</td>
<td>0</td>
<td>5</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>Average</td>
<td>28</td>
<td>7</td>
<td>196</td>
</tr>
<tr>
<td></td>
<td>High</td>
<td>0</td>
<td>10</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td><strong>Sub-Total</strong></td>
<td></td>
<td></td>
<td><strong>196</strong></td>
</tr>
</tbody>
</table>

Total Number of UN-ADJUSTED Function Points: 425

Value Adjustment Factor: 1.07

Total Number of ADJUSTED Function Points = SUPPORT WORK PRODUCT: 455
Relative Contribution of Function Types

This report displays the percentage of the functional size contributed by each Function Type within the count. The contributions can be compared with the industry average contribution.

<table>
<thead>
<tr>
<th>Function Type</th>
<th>Contribution (%)</th>
<th>Industry Average Contribution (%)</th>
<th>Function Ratio (Transactions : Files)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 External Inputs</td>
<td>15.6</td>
<td>31.2</td>
<td>0.3 : 1</td>
</tr>
<tr>
<td>2 External Outputs</td>
<td>16.3</td>
<td>23.5</td>
<td>0.3 : 1</td>
</tr>
<tr>
<td>3 External Logical Files</td>
<td>7.3</td>
<td>13.2</td>
<td>0.2 : 1</td>
</tr>
<tr>
<td>4 External Interface Files</td>
<td>26.9</td>
<td>22.3</td>
<td>NA</td>
</tr>
<tr>
<td>5 External Methods</td>
<td>46.0</td>
<td>3.0</td>
<td>NA</td>
</tr>
</tbody>
</table>

Relative Contribution of Function Types

The source for the Industry Average Complexity values is the International Software Benchmarking Standards Group (ISBSG) Benchmark 7. www.isbsg.org

CHARMATEK 5 of 5
Outline

• Overview
• Function Point Methodology
  • Developing Historical Database
• Application of COCOMO II
• Concluding Remarks
Software Estimation Challenge

**Problem**: Federal Agencies are not very good at collecting detailed software experience data to calibrate and validate new software estimates

**Contributors**:

- We are primarily organized around projects and do not consistently collect, analyze and archive relevant data for corporate and organizational benefit
- Data is often inaccessible, inconsistent, missing, rolled up or in other hard to use form (e.g. labor cost not labor months, multiple languages map to one WBS, cannot distinguish new development from effort on reuse, modifications, COTs integration, etc.)
- Lack of repository to collect and organize detailed project data into useful analytical information
Software Estimation Challenge

Why is this a problem?

• Studies with COCOMO II show significant improvement in both schedule and effort predictive accuracy when calibrating the multiplicative constant and exponential constant with “local” data *

• Backfiring introduces several sources of potential error, but in the end we will need to see “apples” to compare to vendor “apples”

• We cannot improve until we set the bar!

Software Estimation Challenge

Taking up the challenge – attacking the problem!

• Developing a Software Historical Estimation Database (SHED)
• Started with two recent projects (DS and URET) to locate, analyze and tabulate actual versus estimated LOC, FP, effort, requirement count by mode, type, language, etc.
• Object: determine most useful (informational) data relationships and develop database to make data as accessible, consistent, complete and granular as practical
• Iterate for new and perhaps other recent complete SIS developments
You do not have the data you thought you had
The data does not mean what you thought
Some of the “data” is just plain erroneous
No one ever thought you would need “that” data
Outline

• Overview
• Function Point Methodology
• Developing Historical Database
• Application of COCOMO II
• Concluding Remarks
Purpose of COCOMO II

To help people reason about the cost and schedule implications of their software decisions

©University of Southern California, Center for Software Engineering
COCOMO II Overview

- Software product size estimate
- Software product, process, computer, and personnel attributes
- Software reuse, maintenance, and increment parameters
- Software organization’s project data

Software development, maintenance cost and schedule estimates
Cost, schedule distribution by phase, activity, increment
COCOMO II recalibrated to organization’s data

©University of Southern California, Center for Software Engineering
COCOMO II Model Stages

Relative Size Range

Phases and Milestones

©University of Southern California, Center for Software Engineering
Early Design and Post-Arch Models

• Effort:

\[ PM_{\text{estimated}} = A \times (\text{Size})^{(SF)} \times \prod_{i} EM_i \]

• Size
  – KSLOC (Thousands of Source Lines of Code)
  – UFP (Unadjusted Function Points) * KSLOC/UFP
    • KSLOC/UFP factor varies by language
  – EKSLOC (Equivalent KSLOC) used for adaptation

• SF: Scale Factors (5)

• EM: Effort Multipliers (7 for ED, 17 for PA)
Tailoring and Application of COCOMO II

- COCOMO II is an industry calibrated parametric model
- Equations convert the driver values (including) size to effort and schedule based on data on multiple projects (161 in the current database)
  - 17 effort adjustment factor (multipliers) cover 4 kinds of parameters:
    - Product, Platform, Project and Personnel
  - 5 [economy/diseconomy of] "scale" factors
- Local calibration possible if enough data available: Common errors within an organization can be factored out
  - by adjusting base multiplier using linear regression (LR)
  - by adjusting base exponent using LR in log space
Tailoring and Application of COCOMO II

- Incremental Builds requires adaptation of model, especially those that are Spiral Model based with multiple development spirals
  - Often trade development cost for earlier capability availability (i.e. schedule)
  - Less overall "breakage" due to working kinks in high-priority capabilities out early; but higher breakage [potential] in/for deferred capabilities

- System of Systems presents new challenges: accumulating cost; but with dependencies across systems; ...need to gather new historical data
ERAM Conceptual Plan

Product of Lockheed Martin ATM

---

**URET**

- 2002: U2.1
- 2003: U3.0
- 2004: U3.1
- 2005: U4.0
- 2006: U4.1
- 2007: U5.0
- 2008: U6.0
- 2009: U7.0
- 2010: U8.0

**DSR**

- BCC22
- BCC23
- BCC24
- BCC25
- BCC26
- BCC27
- BCC28

**EIC and R-Position Refresh**

- Early Dev Checkpoints
- Software Drops
- H.O. to I&T

---

**Sync**

- Sites Operational
- Last Site Operational

---

**Note:** This release is never operational

---

**S=Support Infra, I=Ops Infra, C=CP Apps, D=Display Apps, E=ERAM Apps**
Nonlinear Reuse Effects

Data on 2954 NASA modules*

©University of Southern California, Center for Software Engineering

Reuse and Reengineering Effects

• Add Assessment & Assimilation increment (AA)
  – Similar to conversion planning increment

• Add software understanding increment (SU)
  – To cover nonlinear software understanding effects
  – Coupled with software unfamiliarity level (UNFM)
  – Apply only if reused software is modified

• Results in revised Equivalent Source Lines of Code (ESLOC)
  – AAF = 0.4(DM) + 0.3 (CM) + 0.3 (IM)
  – ESLOC = ASLOC[AA+AAF(1+0.02(SU)(UNFM))], AAF < 0.5
  – ESLOC = ASLOC[AA+AAF(SU)(UNFM))], AAF > 0.5
Re-estimation Reasons

• Size Estimate changes: more detailed
• Functionality changes => size changes
• Environment changes => driver changes
• Milestones
• Close-loop control
Using COCOMO II to Cope With Change

COCOMO II

System objectives: functionality, performance, quality

Project Parameters: Personnel, team, sites, platform

Corporate parameters: tools, processes, reuse

Rescope

Cost, schedules, Risks

Execute project to next milestone

Yes

No

Milestone plans, resources

Revise milestones, plans, resources

Milestone expectations

Yes

No

Revised expectations

Done?

Yes

End Project

Milestone results

No

©University of Southern California, Center for Software Engineering
Outline

• Overview
• Function Point Methodology
• Developing Historical Database
• Application of COCOMO II
• Concluding Remarks
Program Control of Estimate Risk

• Traditional estimate approach used in parallel
• New estimate approach using FP & COCOMO II
• Separate Investment Analysis Team develops estimate with SEER™ (Galorath's Suite of Analysis Tools)
• Vendors proposal estimate and supporting documentation
Initial Findings

• It takes a lot of dedicated time and effort by the Software Acquisition specialists to do this
• Preliminary results of FPM are encouraging; we still need to address counts attributable to NDI/Reuse
• We probably will have to keep working on historical data base and perhaps use it as part of estimate revisions
• Be prepared: You may have to slay some dragons along the way
Questions?