



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



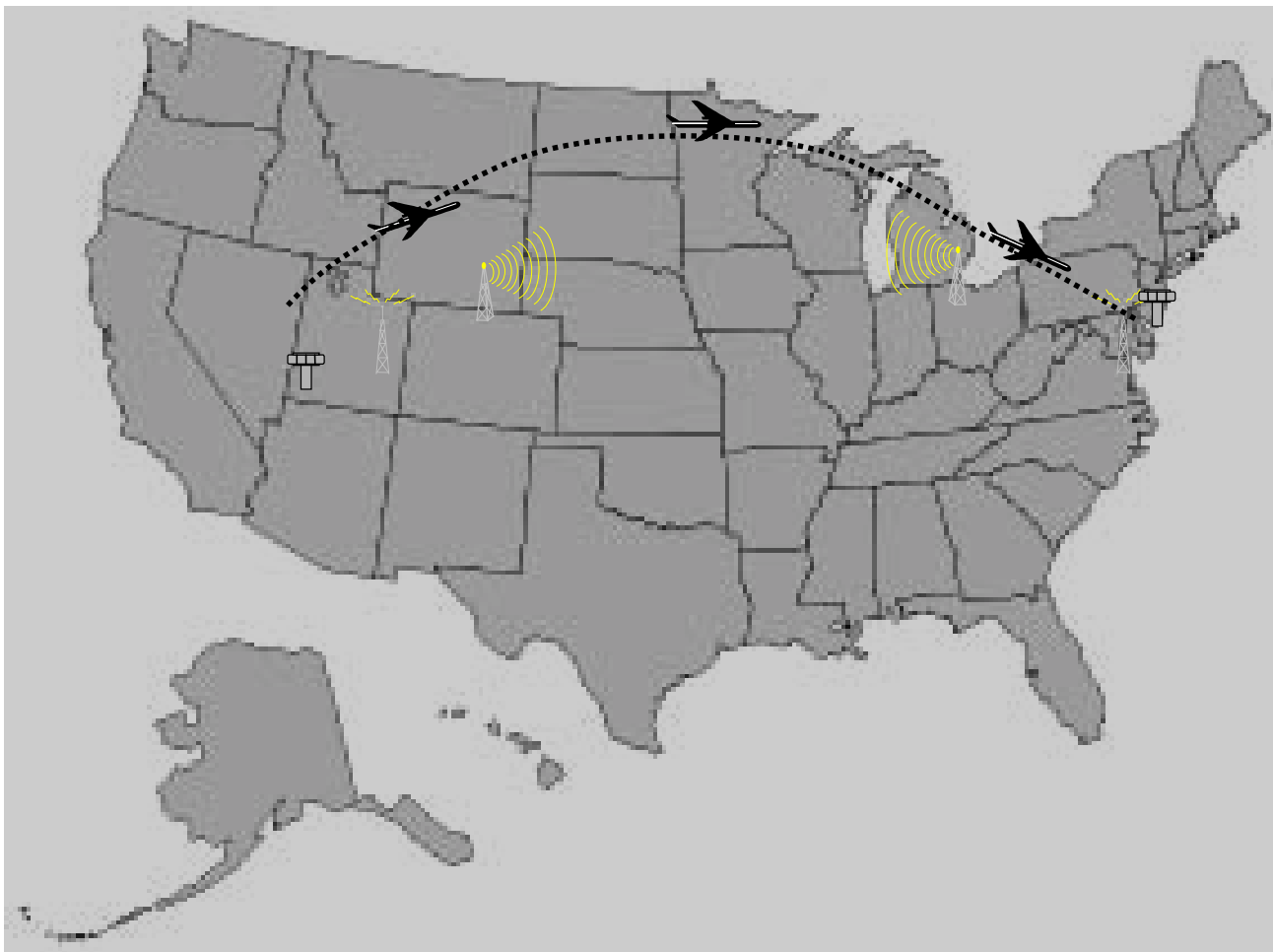
Outline

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



The FAA's Job

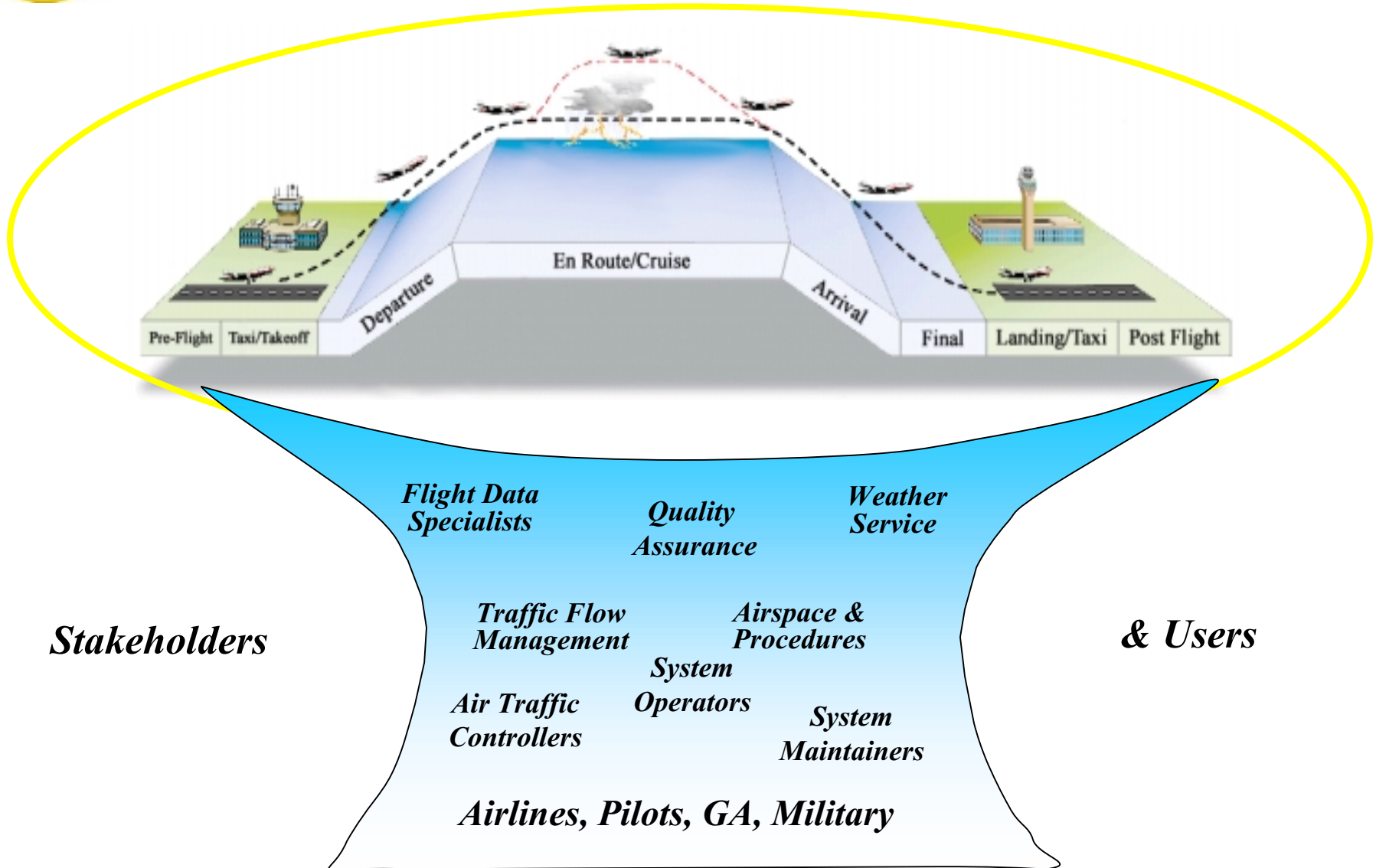
Each day, manage 30,000 commercial flights to safely move 2,000,000 passengers



- ~ 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





En Route Automation Systems

- Consist of a number of software intensive systems

System	Type/Primary Language(s)	Size (SLOC 000s)
NAS Host	Mainframe / Jovial, BAL	300
	Mainframe / Custom OS, BAL	400
	Various Support / Jovial, Bal, REXX	472
Display System	Distributed (UNIX) / Ada83, C	440
	Support Applications/ Ada, C, Fortran	350
User Request Evaluation Tool	Client/Server (UNIX) / Ada95	105
	Display X-Windows/ Ada83, C	64
	Dist. RT (Unix/Posix)/ Ada95, C	229
	Support Applications/ Ada, C, SQL	222
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



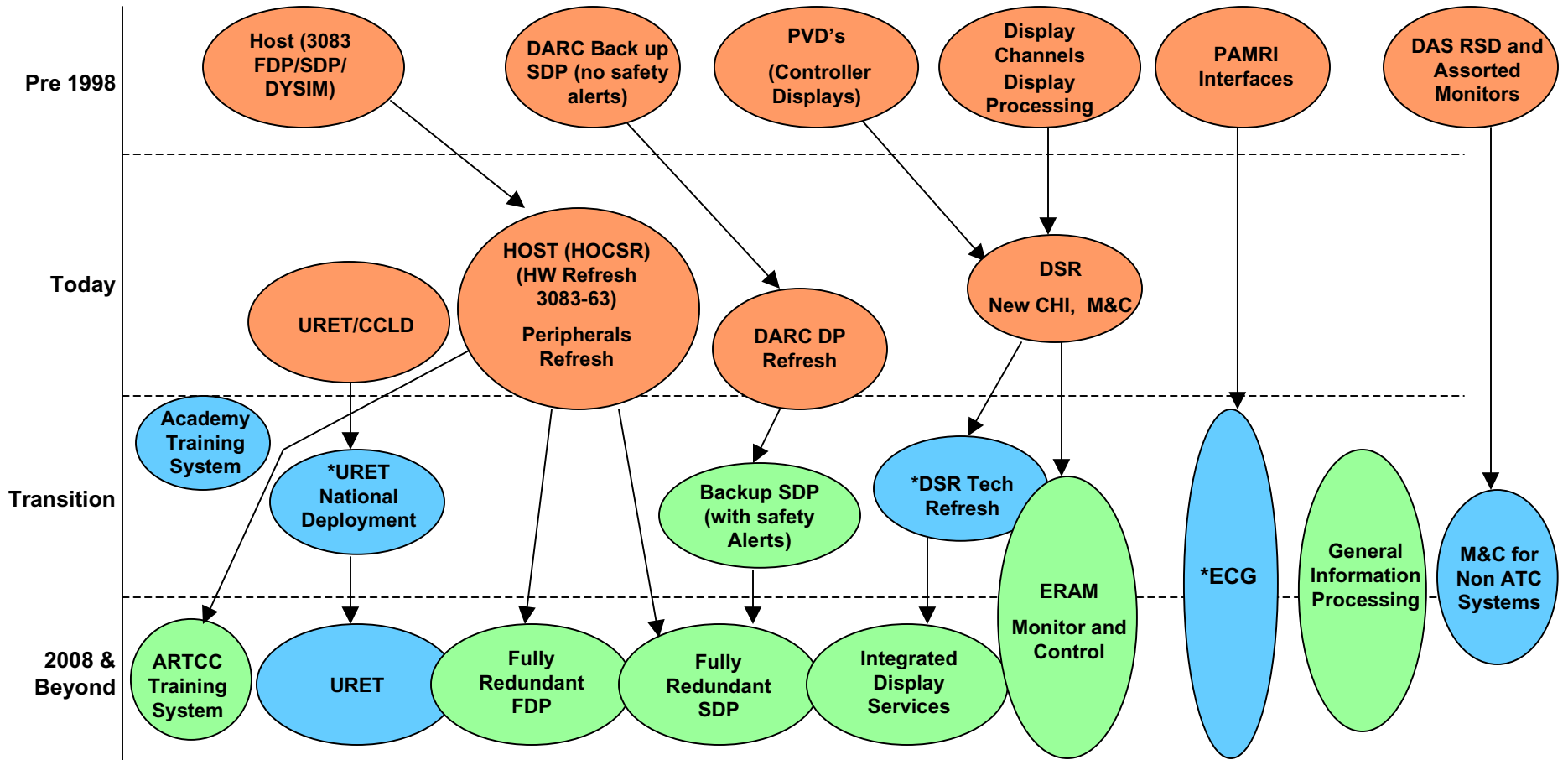
ERAM Infrastructure Acquisition

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



- 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



Estimating Software Size: Impact & Methodologies

Timothy J. Hohmann
Galorath Associates, Inc.
El Segundo, CA

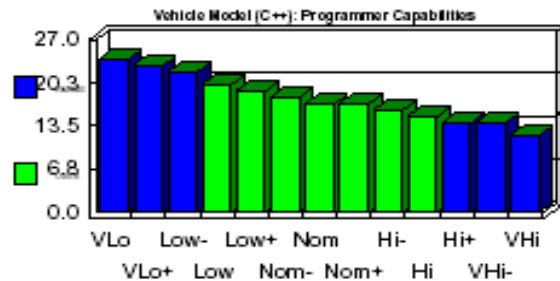


Figure 4. Effort Sensitivity to Programmer Experience

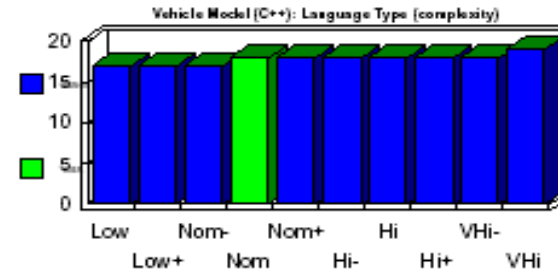


Figure 2. Effort Sensitivity to Language Complexity

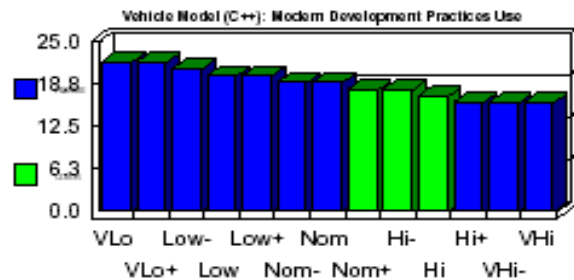


Figure 1. Effort Sensitivity to Modern Development Practices

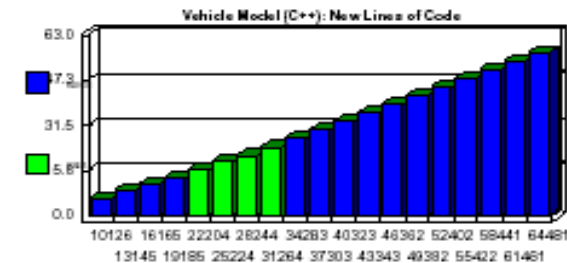
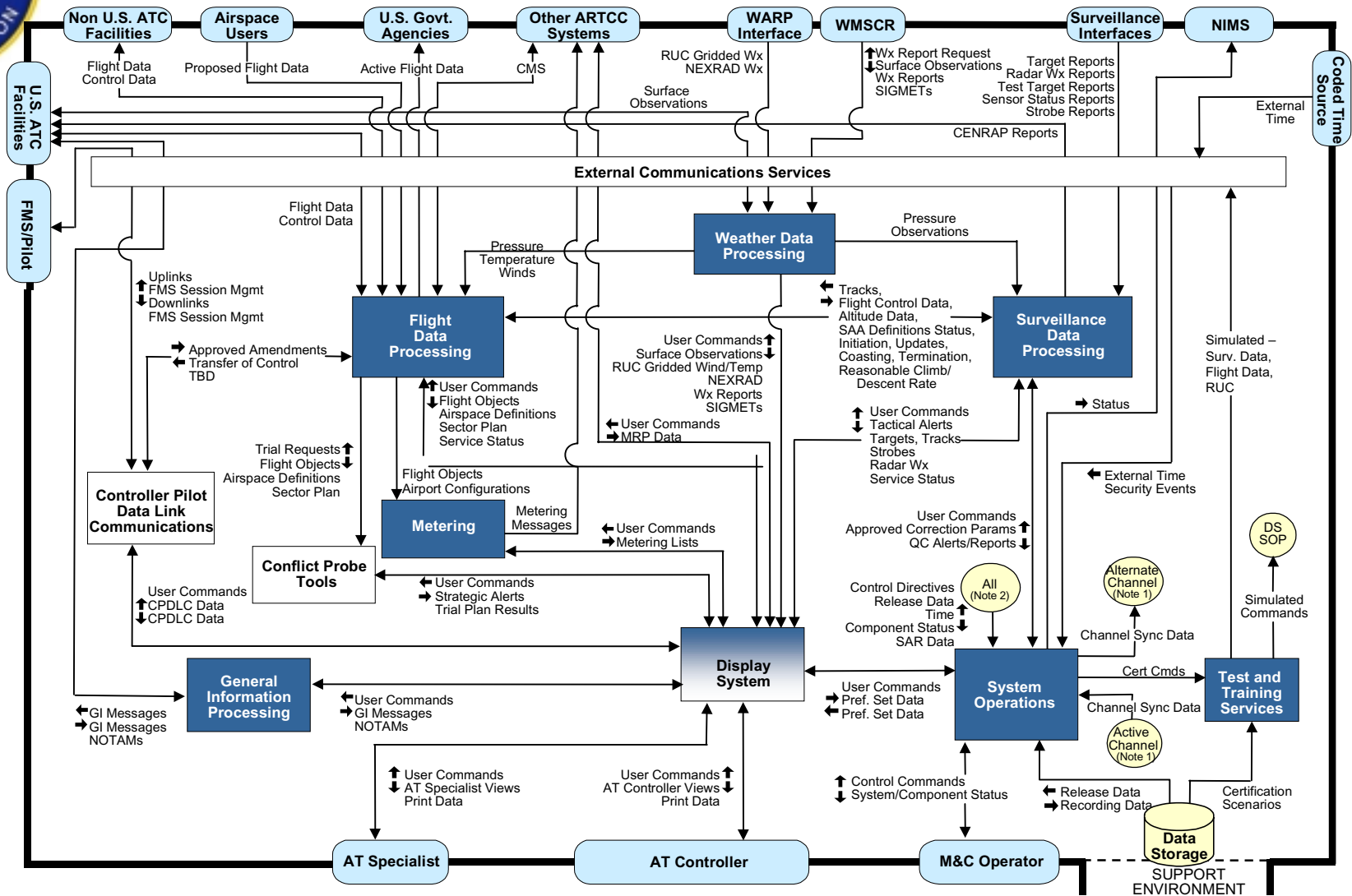


Figure 3. Effort Sensitivity to Size

Mike Kimmel and Lee Fischman (Galorath Associates Inc., El Segundo, CA),
“Estimating Relative Language Productivities”



ARTCC En Route Automation System Operational Functional Architecture



ERAM Subsystem

Note 1: FDP, SDP, DS, SOP, WDP, GIP, MET, CPDLC produce and receive channel synchronization data
 Note 2: FDP, TTS, SDP, DS, CPT, WDP, GIP, MET produce and receive data from SOP

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



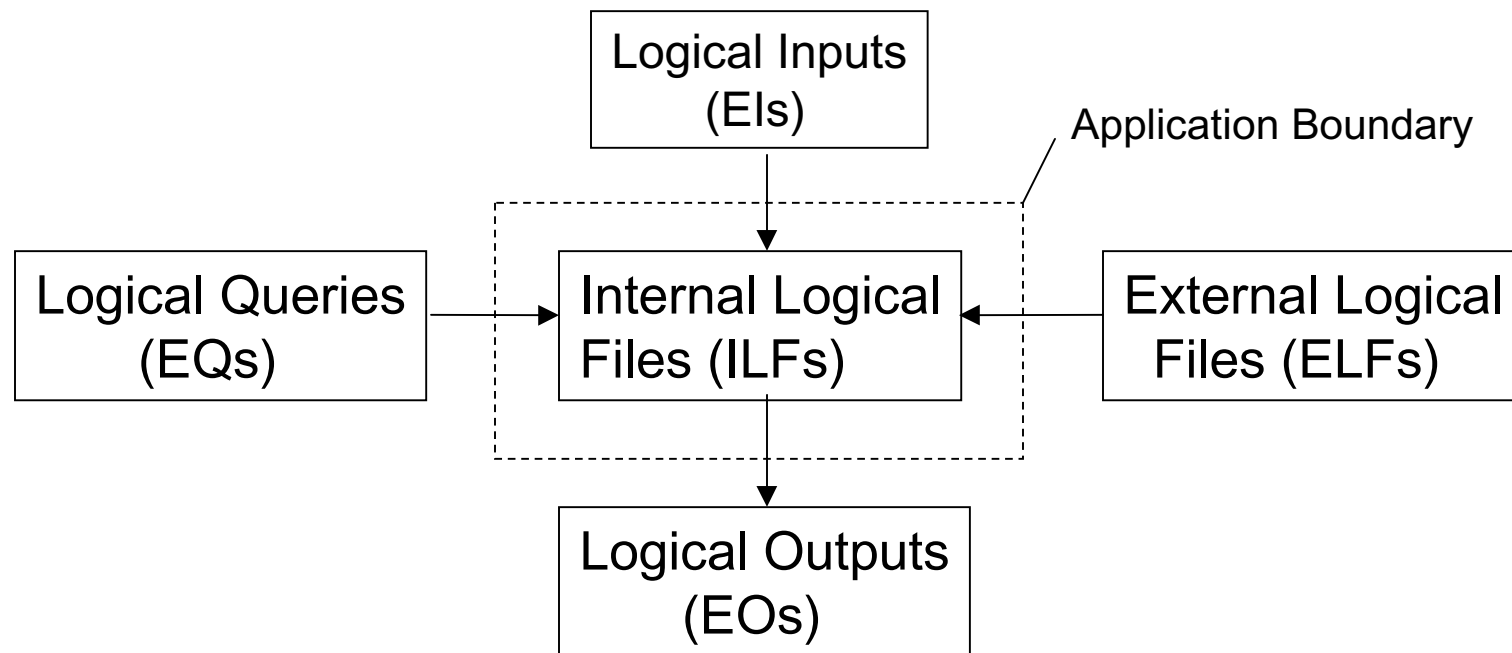
Function Point Methodology

- 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





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)*

*Gartner Research, *Function Points Can Help Measure Application Size*, Research Note SPA-18-0878, 19 November 2002



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

ERAM Display System (NG)		---	---
Display System-SRS		Application	
View:	Count		
Level:	0	(INCLUDES FILE COUNTS)	
Component:	ERAM Display System (NG)		
Selection:	NONE		

Function Type	Complexity Rating	Number of Functions	Function Points Awarded	Total Function Points
EXTERNAL INPUTS	Low	9	3	27
	Average	3	4	12
	High	1	6	6
	Sub-Total	13		45
EXTERNAL OUTPUTS	Low	7	4	28
	Average	3	5	15
	High	0	7	0
	Sub-Total	10		43
EXTERNAL INQUIRIES	Low	3	3	9
	Average	4	4	16
	High	1	6	6
	Sub-Total	8		31
INTERNAL LOGICAL FILES	Low	0	7	0
	Average	11	10	110
	High	0	15	0
	Sub-Total	11		110
EXTERNAL INTERFACE FILES	Low	0	5	0
	Average	28	7	196
	High	0	10	0
	Sub-Total	28		196
Total Number of UN-ADJUSTED Function Points				425
VALUE ADJUSTMENT FACTOR				1.07
Total Number of ADJUSTED Function Points = SUPPORT WORK PRODUCT				455



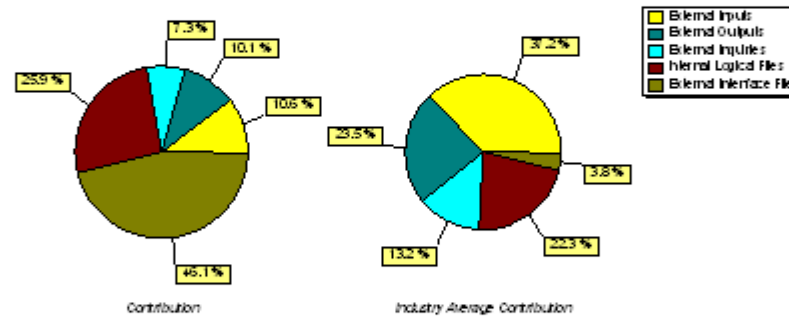
Relative Contribution of Function Types

This report displays the percentage of the functionals contributed by each Function Type within the context. The contribution can be compared with the industry average contribution.

ERAM Display System (M/G)	--	--
Display System-GRS	Application	
View:	Count	
Selection:	NONE	

Function Type	Contribution (%)	Industry Average Contribution ¹ (%)	Function Ratio (Transactions : Files)
1 External Inputs	10.6	37.2	0.3 : 1
2 External Outputs	10.1	23.5	0.3 : 1
3 External Inquiries	7.3	13.2	0.2 : 1
4 Internal Logical Files	25.9	22.3	NA
5 External Interface Files	45.1	3.8	NA

Relative Contribution of Function Types



¹The source of the Industry Average Complexity values is the International Software Benchmarking Standards Group (BSG) Benchmark 5, www.isbs.org.au



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!

*[B. Boehm, et. al., Software Cost Estimation with COCOMO II (Prentice Hall: Upper Saddle River, NJ; 2000), pp 175 – 181.]



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



WARNING: Beyond This Point Be Dragons

When you attempt to do this at a granular level you will likely discover:



- 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



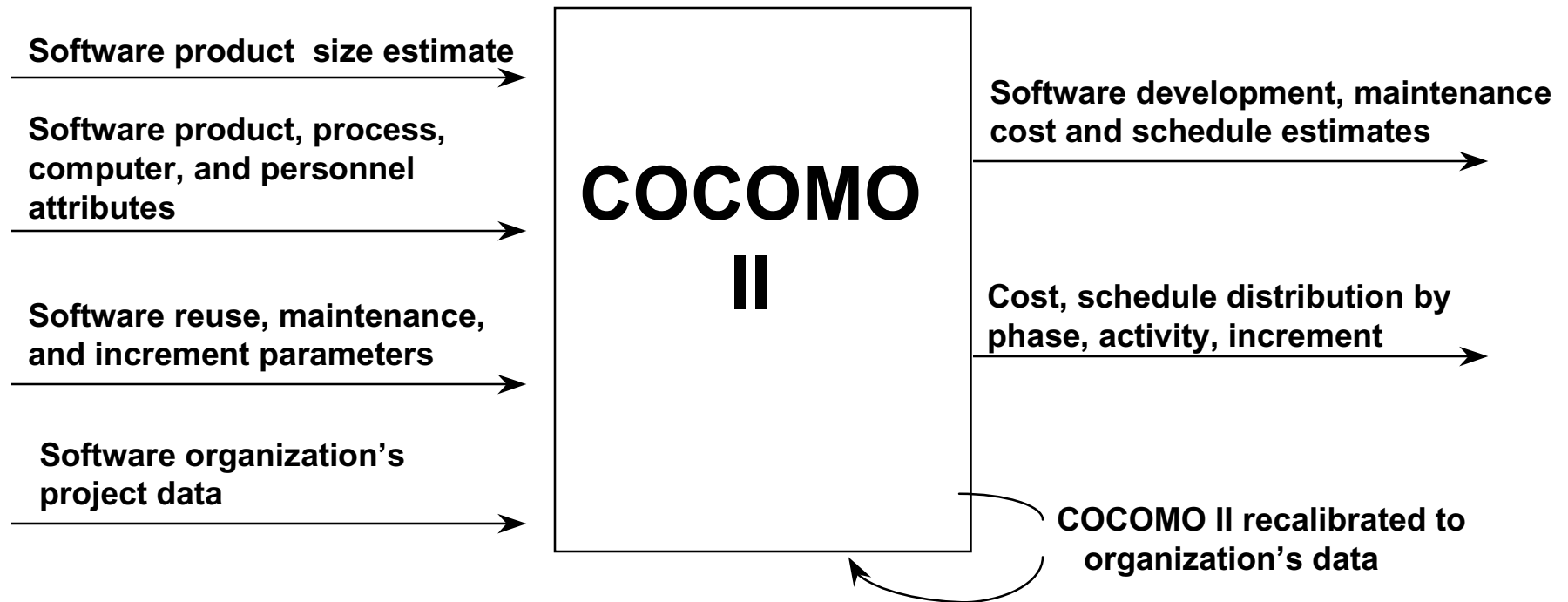
Purpose of COCOMO II

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

©University of Southern California, Center for Software Engineering

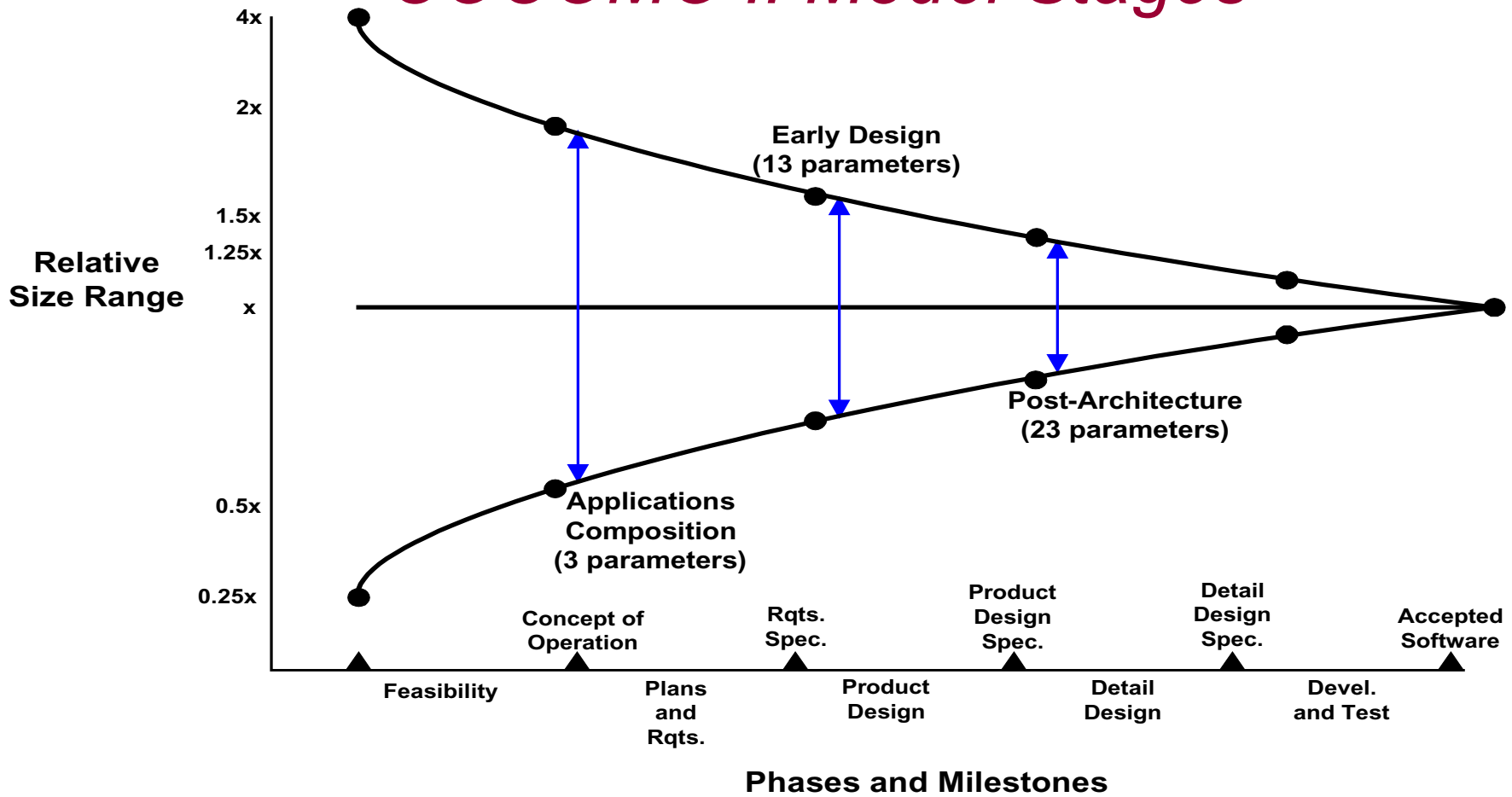


COCOMO II Overview





COCOMO II Model Stages



©University of Southern California, Center for Software Engineering



Early Design and Post-Arch Models

- Effort:

$$PM_{estimated} = A \times (Size)^{(SF)} \times \left\{ \prod_i EM_i \right\}$$

- 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

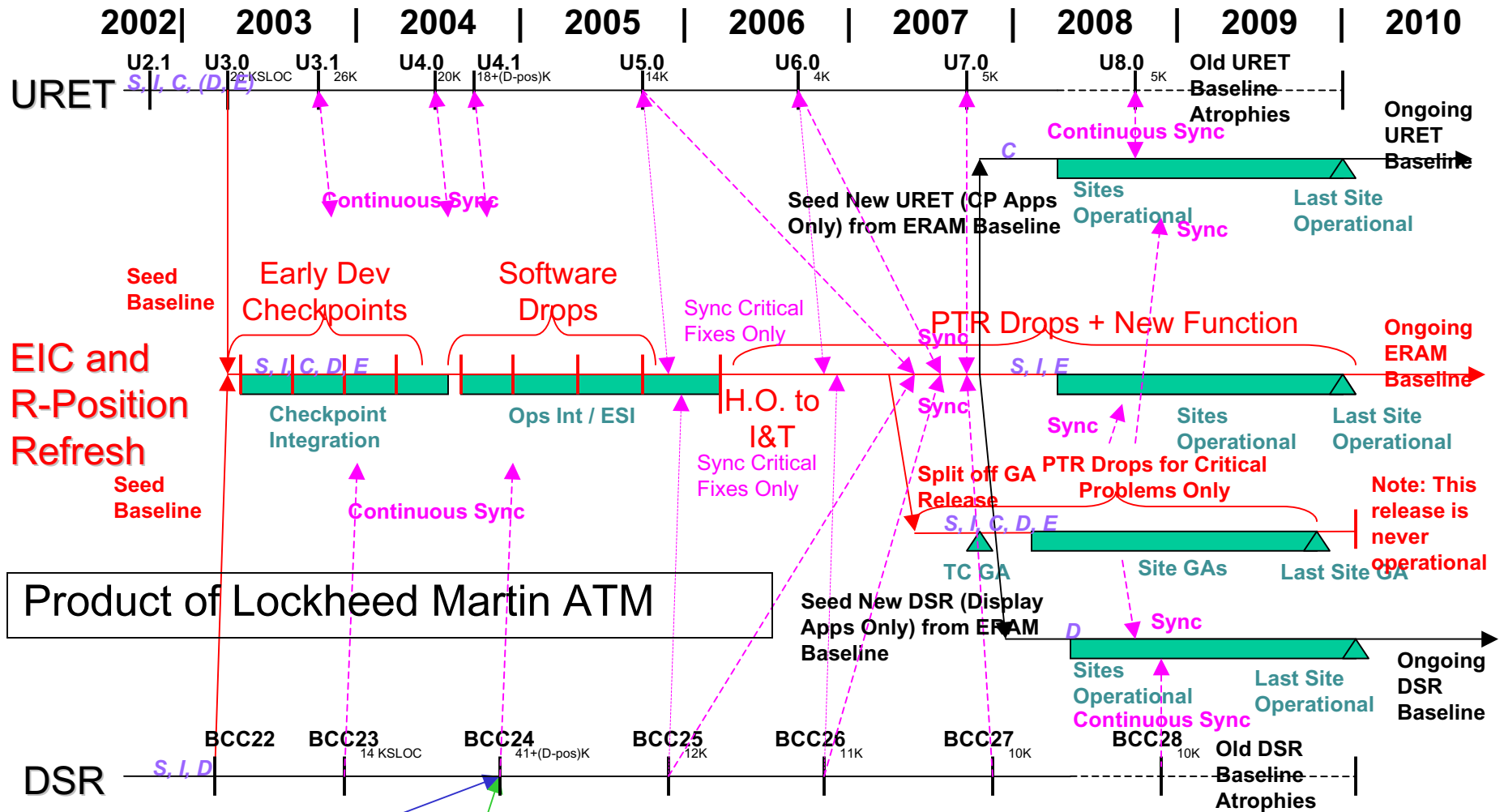


Tailoring and Application of COCOMO II

- 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

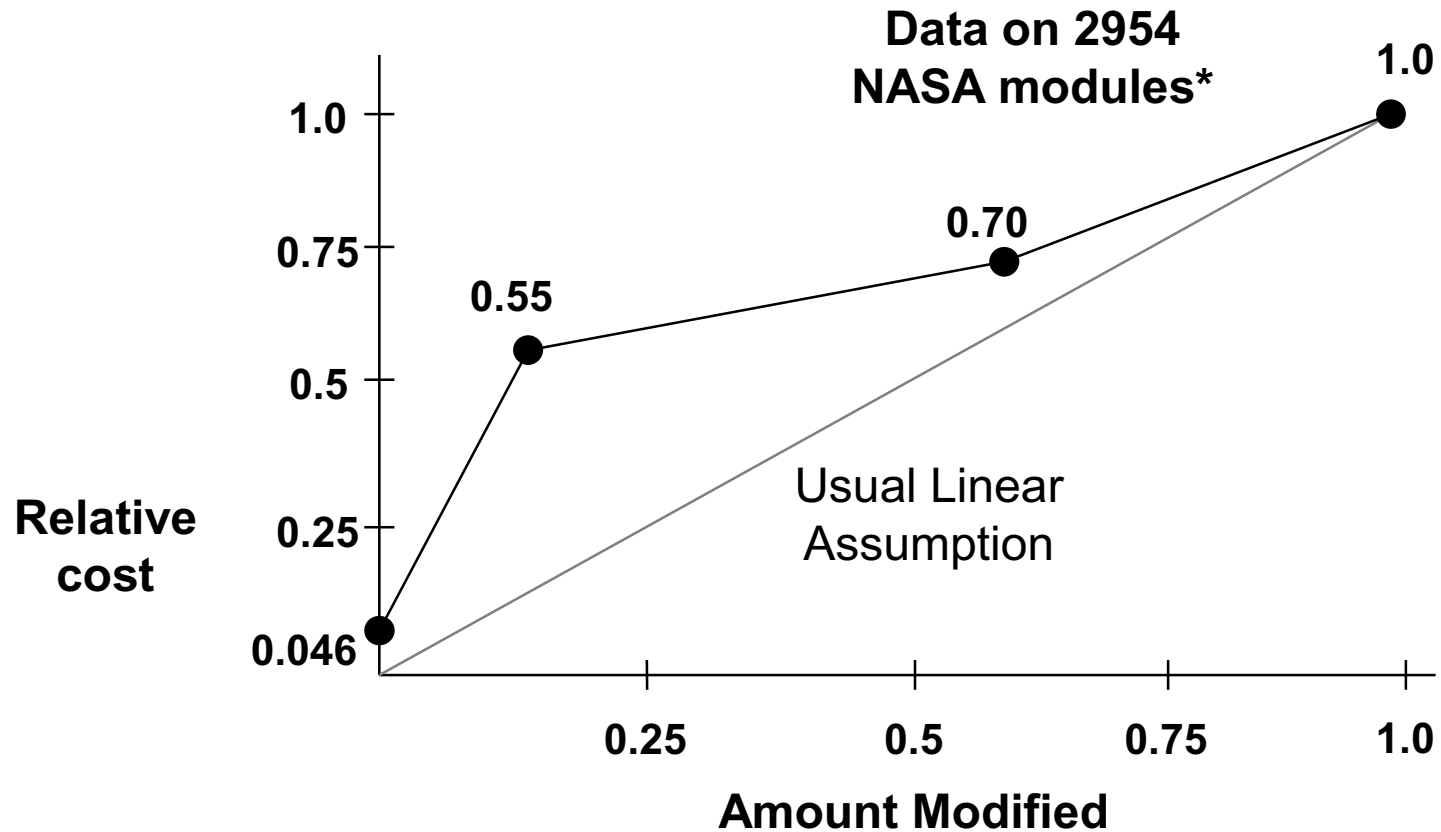


S=Support Infra, I= Ops Infra, C=CP Apps, D=Display Apps, E=ERAM Apps

EBUS
D-Position Refresh



Nonlinear Reuse Effects



©University of Southern California, Center for Software Engineering

* R. Selby, "Empirically Analyzing Software Reuse in a production Environment," in *Software Reuse: Emerging Technology*, IEEE Computer Society Press, 1988., pp 176-189



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)
 - $AAF = 0.4(DM) + 0.3(CM) + 0.3(IM)$
 - $ESLOC = ASLOC[AA+AAF(1+0.02(SU)(UNFM))]$,
 - $AAF < 0.5$
 - $ESLOC = ASLOC[AA+AAF(SU)(UNFM)]$, $AAF > 0.5$



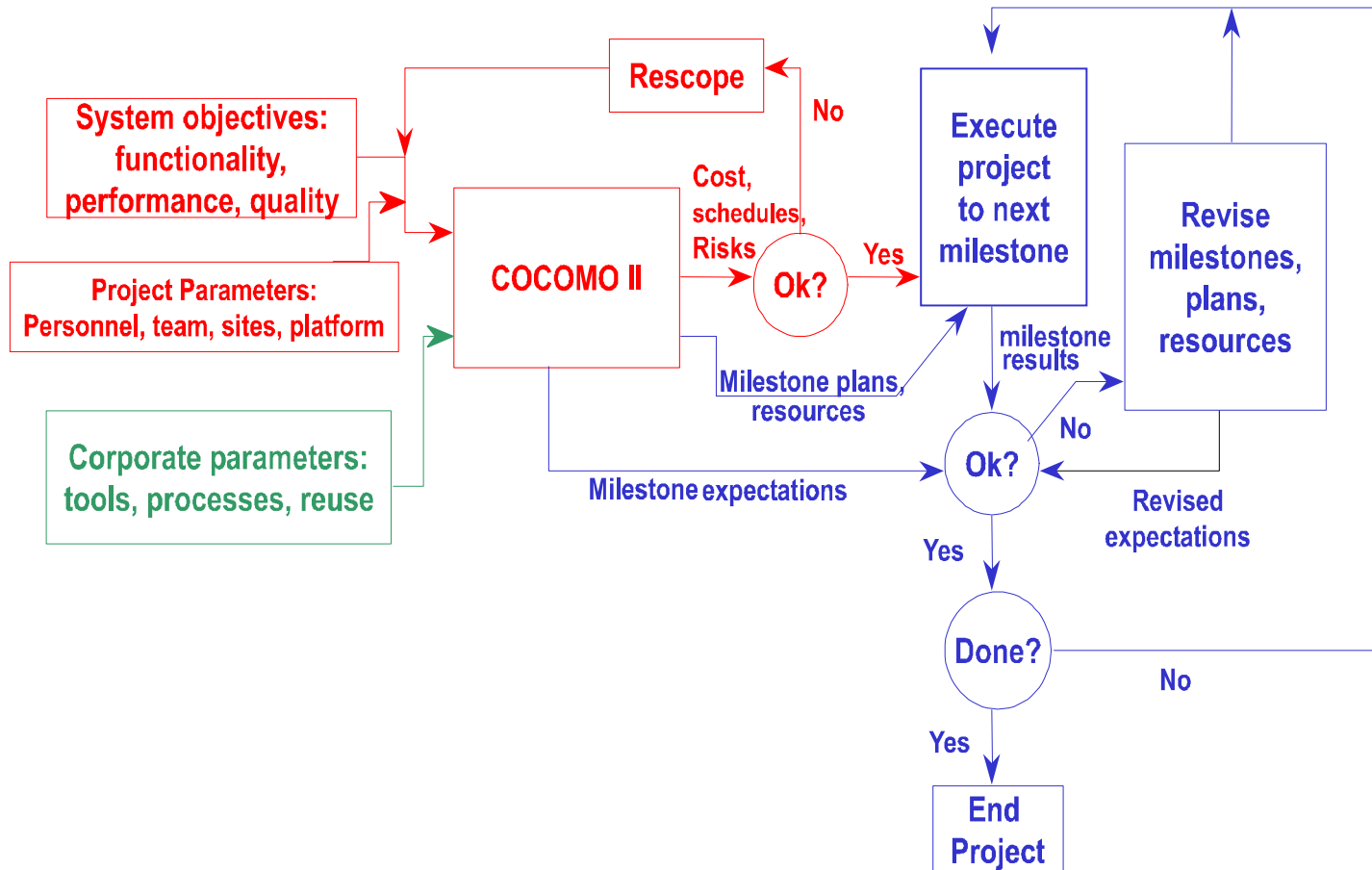
Re-estimation Reasons

- Size Estimate changes: more detailed
- Functionality changes => size changes
- Environment changes => driver changes
- Milestones
- Close-loop control

©University of Southern California, Center for Software Engineering



Using COCOMO II to Cope With Change





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?

??
??
??
??
??
??
??
??
??
??