Developing and Integrating Software/System Product, Process, Property, and Success Models

Barry Boehm, USC-CSE
IBM T. J. Watson Presentation
August 13, 1999

Boehm@sunset.usc.edu
http://sunset.usc.edu
Outline

• Developing Software/System Models at USC-CSE
  – Process Models
  – Product Models
  – Property Models
  – Success Models
• Integrating Software/System Models
• Conclusions
• Further information
Process Model Research at USC-CSE

• Spiral Model Extensions
  – WinWin spiral model
  – Life cycle anchor points

• WinWin Negotiation Model
  – IPT, JAD guidelines
  – Groupware support tools

• Object-oriented processes with COTS, reuse

• RAD-SAIV process model

• System dynamics models
Spiral Model Experience

- Where do objectives, constraints, alternatives come from?
  - Win Win extensions

- Lack of intermediate milestones
  - Anchor Points: LCO, LCA, IOC
  - Concurrent-engineering spirals between anchor points

The WinWin Spiral Model

1. Identify next-level Stakeholders
2. Identify Stakeholders’ win conditions
3. Reconcile win conditions. Establish next level objectives, constraints, alternatives
4. Evaluate product and process alternatives. Resolve Risks
5. Define next level of product and process - including partitions
6. Validate product and process definitions
7. Review, commitment
Life Cycle Anchor Points

• Common System/Software stakeholder commitment points
  – Defined in concert with Government, industry affiliates
  – Coordinated with Rational’s Unified Software Development Process

• Life Cycle Objectives (LCO)
  – Stakeholders’ commitment to support system architecting
  – Like getting engaged

• Life Cycle Architecture (LCA)
  – Stakeholders’ commitment to support full life cycle
  – Like getting married

• Initial Operational Capability (IOC)
  – Stakeholders’ commitment to support operations
  – Like having your first child
**Win Win Spiral Anchor Points**

(Risk-driven level of detail for each element)

<table>
<thead>
<tr>
<th>Milestone Element</th>
<th>Life Cycle Objectives (LCO)</th>
<th>Life Cycle Architecture (LCA)</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Definition of Operational Concept</strong></td>
<td>• Top-level system objectives and scope</td>
<td>• Elaboration of system objectives and scope of increment</td>
</tr>
<tr>
<td></td>
<td>- System boundary</td>
<td>• Elaboration of operational concept by increment</td>
</tr>
<tr>
<td></td>
<td>- Environment parameters and assumptions</td>
<td></td>
</tr>
<tr>
<td></td>
<td>- Evolution parameters</td>
<td></td>
</tr>
<tr>
<td></td>
<td>- Operational concept</td>
<td></td>
</tr>
<tr>
<td></td>
<td>- Operations and maintenance scenarios and parameters</td>
<td></td>
</tr>
<tr>
<td></td>
<td>- Organizational life-cycle responsibilities (stakeholders)</td>
<td></td>
</tr>
<tr>
<td><strong>System Prototype(s)</strong></td>
<td>• Exercise key usage scenarios</td>
<td>• Exercise range of usage scenarios</td>
</tr>
<tr>
<td></td>
<td>• Resolve critical risks</td>
<td>• Resolve major outstanding risks</td>
</tr>
<tr>
<td><strong>Definition of System Requirements</strong></td>
<td>• Top-level functions, interfaces, quality attribute levels,</td>
<td>• Elaboration of functions, interfaces, quality attributes, and prototypes by increment</td>
</tr>
<tr>
<td></td>
<td>including:</td>
<td>- Identification of TBD’s (to-be-determined items)</td>
</tr>
<tr>
<td></td>
<td>- Growth vectors and priorities</td>
<td>• Stakeholders’ concurrence on their priority concerns</td>
</tr>
<tr>
<td></td>
<td>- Prototypes</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Stakeholders’ concurrence on essentials</td>
<td></td>
</tr>
<tr>
<td><strong>Definition of System and Software Architecture</strong></td>
<td>• Top-level definition of at least one feasible architecture</td>
<td>• Choice of architecture and elaboration by increment</td>
</tr>
<tr>
<td></td>
<td>- Physical and logical elements and relationships</td>
<td>- Physical and logical components, connectors, configurations, constraints</td>
</tr>
<tr>
<td></td>
<td>- Choices of COTS and reusable software elements</td>
<td>- COTS, reuse choices</td>
</tr>
<tr>
<td></td>
<td>- Identification of infeasible architecture options</td>
<td>- Domain-architecture and architectural style choices</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Architecture evolution parameters</td>
</tr>
<tr>
<td><strong>Definition of Life-Cycle Plan</strong></td>
<td>• Identification of life-cycle stakeholders</td>
<td>• Elaboration of WWWWWHH* for Initial Operational Capability (IOC)</td>
</tr>
<tr>
<td></td>
<td>- Users, customers, developers, maintainers, interoperators, general public, others</td>
<td>- Partial elaboration, identification of key TBD’s for later increments</td>
</tr>
<tr>
<td></td>
<td>- Identification of life-cycle process model</td>
<td></td>
</tr>
<tr>
<td></td>
<td>- Top-level stages, increments</td>
<td></td>
</tr>
<tr>
<td></td>
<td>- Top-level WWWWWHH* by stage</td>
<td></td>
</tr>
<tr>
<td><strong>Feasibility Rationale</strong></td>
<td>• Assurance of consistency among elements above</td>
<td>• Assurance of consistency among elements above</td>
</tr>
<tr>
<td></td>
<td>- via analysis, measurement, prototyping, simulation, etc.</td>
<td>• All major risks resolved or covered by risk management plan</td>
</tr>
<tr>
<td></td>
<td>- Business case analysis for requirements, feasible architectures</td>
<td></td>
</tr>
</tbody>
</table>

Initial Operational Capability (IOC)

• Software preparation
  – Operational and support software
  – Data preparation, COTS licenses
  – Operational readiness testing

• Site preparation
  – Facilities, equipment, supplies, vendor support

• User, operator, and maintainer preparation
  – Selection, teambuilding, training
Anchor Points and Rational USDPh Phases

<table>
<thead>
<tr>
<th>Engineering Stage</th>
<th>Manufacturing Stage</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Inception</strong></td>
<td></td>
</tr>
<tr>
<td>Feasibility</td>
<td></td>
</tr>
<tr>
<td>Iterations</td>
<td></td>
</tr>
<tr>
<td><strong>Elaboration</strong></td>
<td></td>
</tr>
<tr>
<td>Architecture</td>
<td></td>
</tr>
<tr>
<td>Iterations</td>
<td></td>
</tr>
<tr>
<td><strong>Construction</strong></td>
<td></td>
</tr>
<tr>
<td>Usable</td>
<td></td>
</tr>
<tr>
<td>Iterations</td>
<td></td>
</tr>
<tr>
<td><strong>Transition</strong></td>
<td></td>
</tr>
<tr>
<td>Product</td>
<td></td>
</tr>
<tr>
<td>Releases</td>
<td></td>
</tr>
</tbody>
</table>

RATIONAL Software Corporation
Win-Win Negotiation Model

Win Condition
- Rationale
- Attachments

Involves

Issue
- Rationale
- Attachments

Addresses

Agreement
- Rationale
- Attachments

Covers

Option
- Rationale
- Attachments

Adopts

Win-Win equilibrium
- All Win Conditions covered by Agreements
- No outstanding Issues
WinWin Look and Feel

- WIN CONDITIONS
  - swong-MINC-1: user friendly
  - swong-MINC-2: online system
  - swong-MINC-3: current web site
  - swong-MINC-4: maintained by USC
  - swong-MINC-5: upgrading system
  - swong-MINC-6: retrieving data
  - swong-MINC-7: online help

- ISSUES
  - swong-ISSU-1: Access to library
  - swong-ISSU-2: necessary equipment
  - swong-ISSU-3: SQL in user interface
  - swong-ISSU-4: Customer will not install
  - swong-ISSU-5: library problems
  - swong-ISSU-6: database cannot retrieve

- OPTIONS
  - swong-OPTN-1: condition for access

- AGREEMENTS
  - swong-AGRE-12: Developer can use
  - swong-AGRE-13: Developer can use

---

1. Operational Modes
   - 1.1 Classes of Service
   - 1.2 Online Training, Documentation
   - 1.3 Backup System

2. Capabilities
   - 2.1 Display and Media Parameters
     - 2.1.1 Text, images, and graphics

---

**Role:**
- User

**Status:**
- Active

**Priority:**
- Very High

**State:**
- [ ]

**Rationale:**
System upgrades should be allowed, because library may have images in other format in the future. System should be reusable. Library doesn't want to spend money later on changing the system all.

---

7/28/99 ©USC-CSE
Product Model Research at USC-CSE

• Guidelines for LCO, LCA product deliverables
  – Operational Concept Description
    - Domain model, stakeholder roles, use case scenarios
  – Requirements Description
    - Priorities, evolution requirements, WinWin traceability
  – Architecture description
    - Extended UML, COTS integration
  – Feasibility Rationale
    - Assurance of consistency, feasibility, risk resolution

• Architecture Research
Architecture Research

• SAAGE (Software Architecture Analysis, Generation, and Evolution)

• Architecture style and view integration
  – UML view integration
Software Architecture, Analysis, Generation, and Evolution (SAAGE)

Research focus:
• style-based application design and implementation
• component- and connector-based architectural composition
• architecture modeling and analysis
• architecture-based software evolution
• consistent refinement of architecture into design
• system generation

Supported by an integrated toolset composed of in-house and third-party tools
SAAGE Overview

• Integrated environment for transforming C2-style architectures into UML
View Integration Model (UML/Analyzer Tool)

A System for Defining and Analyzing the Conceptual Integrity of UML Models

- implements generic view integration model
- describes and identifies causes of architectural and design mismatches across UML views
- is integrated with Rational Rose®
- complements pure view comparison with transformation (abstraction, translation, and unification) and mapping techniques
- currently supports class, object, sequence, collaboration, state, and C2 diagrams
- uses mismatch rules and transformation rules
Consistency between Architecture and Design

C2 Architecture
(represented in UML)

Class Design
(UML class diagram)

Automatically derived Views

©USC-CSE

7/28/99
Property Model Research at USC-CSE

• Constructive Cost Model (COCOMO) II

• COCOMO II Extensions
  – COCOTS: COTS Integration
  – COQUALMO: Delivered Defect Density
  – CORADMO: Rapid Development Cost and Schedule
  – COPSEMO: Phase/Activity Distribution
  – COPROMO: Productivity Improvement Analysis

• Property Tradeoff Assistance: QARCC, S-COST
COCOMO II Summary

• Extended Modeling Approach

  Effort ~ (Environment) * (Size)\(^\text{Process}\)

  Schedule ~ (Effort)\(^\text{Process}\)

• Tailorable Family of Models

  - Keyed to information available and development approach
## COCOMO II Family

<table>
<thead>
<tr>
<th>Model</th>
<th># Drivers</th>
<th>Sizing</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Envir.</td>
<td>Process</td>
</tr>
<tr>
<td>Applications Composition</td>
<td>2</td>
<td>0</td>
</tr>
<tr>
<td>Early Design</td>
<td>7</td>
<td>5</td>
</tr>
<tr>
<td>Post-Architecture</td>
<td>17</td>
<td>5</td>
</tr>
<tr>
<td>COCOMO 81</td>
<td>15</td>
<td>1</td>
</tr>
</tbody>
</table>
Calibration: COCOMO II.1997 Vs. .1999

<table>
<thead>
<tr>
<th></th>
<th>1981</th>
<th>1997</th>
<th>1999</th>
</tr>
</thead>
<tbody>
<tr>
<td>Project Data Points</td>
<td>63</td>
<td>83</td>
<td>161</td>
</tr>
<tr>
<td>Calibration</td>
<td></td>
<td>10% Data, 90% Experts</td>
<td>Bayesian</td>
</tr>
<tr>
<td>PRED (.30) Values</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>• Effort</td>
<td>81%</td>
<td>52%</td>
<td>75%</td>
</tr>
<tr>
<td>– by Org’n</td>
<td>64%</td>
<td>80%</td>
<td></td>
</tr>
<tr>
<td>• Schedule</td>
<td>65%</td>
<td>61%</td>
<td>72%</td>
</tr>
<tr>
<td>– by Org’n</td>
<td>62%</td>
<td>81%</td>
<td></td>
</tr>
</tbody>
</table>

7/28/99  ©USC-CSE  20
Process Maturity (PMAT) Effects

- Effort reduction per maturity level, 100 KDSI project
- Normalized for effects of other variables

- Clark Ph.D. dissertation (112 projects)
  - Research model: 12-23% per level
  - COCOMO II subsets: 9-29% per level

- COCOMO II.1999 (161 projects)
  - 4-11% per level

- PMAT positive contribution is statistically significant
COCOMO vs. COCOTS Cost Sources

1) COTS Assessment
2) COTS Tailoring

3) COTS/Application Glue Code Development and (System) Test

Application Code Development Integration and Test Separate from COTS Effects

4) Increased Application Effort due to COTS Volatility

LCO - Life Cycle Objectives
LCA - Life Cycle Architecture
IOC - Initial Operational Capability

COCOMO Effort Estimate
COCOTS Effort Estimate Components

STAFFING

TIME
Integrated COQUALMO

- Software Size estimate
- Software product, process, computer and personnel attributes
- Defect removal capability levels
- Defect density per unit of size
- Software development effort, cost and schedule estimate
- Number of residual defects

Defect Introduction Model

COQUALMO

cost
COCOMO II RAD Extension (CORADMO)

COCOMO II cost drivers (except SCED)

Language Level, experience,...

Stage Distributions

Baseline effort, schedule

Effort, schedule by stage

CORADMO

RVHL
BPRS
CLAB
RESL
PPOS

RAD effort, schedule by stage
COCOMO II Book

- Publication date now early-2000
  - Full draft by August 1
- Will use current calibration values as COCOMO II.2000
- Providing USC COCOMO II.2000 on CD included with book
  - Also manuals, tutorials, commercial model demos
- Plan new edition, recalibration every 2 years
  - Affiliates - only recalibration in interim
Success Model Research at USC-CSE

- **Stakeholder Win-Win (Theory W)**
  - Win win negotiation model and tools
  - Two Cultures and expectations management

- **Success Model profiles and model clashes**
Unmet Expectations Problems

• LCO success condition
  – Describes at least one feasible architecture
  – Satisfying requirements within cost/schedule/resource constraints
  – Viable cost-effective business case
  – Stakeholder concurrence on key system parameters

• Projects That Failed LCO Criteria
  - 1996: 4 out of 16 (25%)
  - 1997: 4 out of 15 (27%)

why?
Requirements and Expectations: Domain Model Clashes

• Easy/hard things for software people

“If you can do queries with all those ands, ors, synonyms, data ranges, etc., it should be easy to do natural language queries.”

“If you can scan the document and digitize the text, it should be easy to digitize the figures too.”

• Easy/hard things for librarians

“It was nice that you could add this access feature, but it overly (centralizes, decentralizes) control of our intellectual property rights.”

“It was nice that you could extend the system to serve the medical people, but they haven’t agreed to live with our usage guidelines.”
1998 Simplifier/Complicator Experiment

- Identify application simplifiers and complicators
  - For each digital library sub-domain
  - For both developers and clients
- Provide with explanations to developers and clients
  - Highlight relation to risk management
- Homework exercise to analyze simplifiers and complicators
  - For two of upcoming digital library projects
- Evaluate effect on LCO review failure rate
### Example S&C’s

<table>
<thead>
<tr>
<th>Type of Application</th>
<th>Simple Block Diagram</th>
<th>Examples</th>
<th>Simplifiers</th>
<th>Complicators</th>
</tr>
</thead>
</table>
| Multimedia Archive  | ![Diagram](query MM asset info) | 1, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 20, 31, 32, 35, 36, 37, 39 | • Use standard query languages  
• Use standard or COTS search engine  
• Uniform media formats | • Natural language processing  
• Automated cataloging or indexing  
• Digitizing large archives  
• Digitizing complex or fragile artifacts  
• Rapid access to large Archives  
• Access to heterogeneous media collections  
• Automated annotation/description/ or meanings to digital assets  
• Integration of legacy systems |
The Results

• Projects That Failed LCO Criteria
  - 1996: 4 out of 16 (25%)
  - 1997: 4 out of 15 (27%)
  - 1998: 1 out of 20 (5%)

• 40% of Student critiques cited S&C’s as helpful
  - In focusing on achievable requirements set within tight schedule
  - In understanding project risks and tradeoffs
Success Model-Clash Profiles: General ...

Users
- Many features
- Changeable requirements
- Applications compatibility & control
- High levels of Service
- Voice in acquisition
- Flexible contract

Acquirers
- Mission cost/effectiveness
- Limited budget, Schedule
- Government standards compliance
- Political correctness
- Development visibility & control
- Rigorous contract

Maintainers
- Freedom of choice: process
- Freedom of choice: team
- Freedom of choice: COTS/reuse

Developers
- Ease of transition
- Ease of maintenance
- Applications compatibility & control
- Voice in acquisition

PC: Process
PD: Product
PP: Property
S: Success

©USC-CSE
Outline

• Developing Software/System Models at USC-CSE
  – Process, Product, Property, and Success Models

• Integrating Software/System Models
  – Motivation: Model Clashes
  – Model-Based (System) Architecting and Software Engineering (MBASE)

• Examples of Successful MBASE Use
  – Over 50 Digital Library projects
  – TRW CCPDS-R project

• Early Adopters
• Conclusions
• Further information
• Model Clash: An incompatibility among the underlying assumptions of a set of models
  – Often unrecognized
  – Produces conflicts, confusion, mistrust, frustration, rework, throwaway systems
Examples of Model Clashes

- Design-to-schedule process, unprioritized requirements, and tightly-coupled architecture
- COTS-driven product and Waterfall process
- Risk-based process and spec-based progress payments
- Evolutionary development without life-cycle architecture
- Golden Rule and stakeholder win-win
- Spec-based process and I’l know it when I see it success model
MBASE Model Integration Framework

Success Models
Win-Win; IKIWISI; Business-Case; Mission Models;...

Process Models
- Life-Cycle
  - Waterfall;
  - Evolutionary;
  - Incremental;
  - WW Spiral
- Anchor Points
- Risk Mgmt.
- Activities
  - CMM KPA’s
...

Product Models
- Domain
- Artifacts
  - Rqts.
  - Arch.
  - Code
  - Doc’n
- Packaging
  - Embedded
  - Shrink Wrap
  - Turn Key
- Product Line
...

Property Models
Cost&Schedule; Performance; Assurance; Usability;...

Entry/Exit Criteria

V&V Criteria

Product Development & Evolution Process

Planning & Control

Evaluation & Analysis
Product Line Domain Scope a Function of ROI, Scope of Empowered PL Manager

Return on Investment (ROI)

Scope of empowered PLM

too few instances to generate payoff

too general to be competitive

Breadth of Domain
MBASE Process Framework

- Stakeholders
- Success Models
- Property Models
- Conceptual Product Models
- Domain/Environment Models
- WinWin Spiral Process
- Life Cycle
- Architecture Package
- Plan in LCA Package
- Process Models
- IPM\(_1\)
- IPM\(_n\)
- Reified Product Models

**Stakeholders**
- enable satisficing among
- impose constraints on
- provide parameters for
- provide evaluations for
- impose constraints on
- determine the relevance of
- identify, prioritize

**Success Models**
- provide parameters for

**Property Models**
- provide parameters for

**Conceptual Product Models**
- are refinements of
- set context for
- determinate the relevance of
- identify, prioritize

**Domain/Environment Models**
-... intermediate...
- reify...

**IPM\(_1\)**
-... intermediate...

**IPM\(_n\)**
-... reify...

**Process Models**
- guide progress in selecting and reifying
- serve and satisfy
## Success Models Drive Other Model Choices

<table>
<thead>
<tr>
<th>Success Model</th>
<th>Process Model</th>
<th>Product Model</th>
</tr>
</thead>
<tbody>
<tr>
<td>Demo agent-based E-commerce system at COMDEX in 9 months</td>
<td>Design-to-schedule</td>
<td>Domain constrained by schedule; architected for ease in dropping features to meet schedule</td>
</tr>
<tr>
<td>Safe air traffic control system</td>
<td>Initial spiral to risk-manage COTS, etc.; Final waterfall to verify safety provisions</td>
<td>Architected for fault tolerance, ease of safety verification</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Key Stakeholders</th>
<th>Key Property Models</th>
<th>Key Property Models</th>
</tr>
</thead>
<tbody>
<tr>
<td>Entrepreneurs, venture capitalists, customers</td>
<td>Schedule estimation</td>
<td>Safety models</td>
</tr>
<tr>
<td>Controllers, Govt. agencies, developers</td>
<td>Safety models</td>
<td>Schedule estimation</td>
</tr>
</tbody>
</table>

### Success Models
- **Demo agent-based E-commerce system at COMDEX in 9 months**
- **Safe air traffic control system**

### Key Stakeholders
- **Entrepreneurs, venture capitalists, customers**
- **Controllers, Govt. agencies, developers**

### Key Property Models
- **Schedule estimation**
- **Safety models**

### Process Models
- **Design-to-schedule**
- **Initial spiral to risk-manage COTS, etc.; Final waterfall to verify safety provisions**

### Product Models
- **Domain constrained by schedule; architected for ease in dropping features to meet schedule**
- **Architected for fault tolerance, ease of safety verification**
Outline

• Developing Software/System Models at USC-CSE
  – Process, Product, Property, and Success Models

• Integrating Software/System Models
  – Motivation: Model Clashes
  – Model-Based (System) Architecting and Software Engineering (MBASE)

➡• Examples of Successful MBASE Use
  – Over 50 Digital Library projects
  – TRW CCPDS-R project

• Early Adopters
• Conclusions
• Further information
The Challenge

• 15 Digital Library Applications
  – 2 sentence problem statements
  – Librarian clients

• 86 Graduate Students
  – 30% with industry experience
  – Largely unfamiliar with each other, Library ops.

* Develop LCA packages in 11 weeks

• Re-form teams from 30 continuing students

* Develop IOC packages in 12 more weeks
  – Including 2-week beta test and transition
Problem Statement #4: Medieval Manuscripts

Ruth Wallach, Reference Center, Doheny Memorial Library

I am interested in the problem of scanning medieval manuscripts in such a way that a researcher would be able to both read the content, but also study the scribe’s hand, special markings, etc. A related issue is that of transmitting such images over the network.
Antiphonarium

Title: Antiphonarium
Author: Catholic Church
Date: 15th Century
Type: Liturgical & Ritual
Style: Not Available

Physical Characteristics:
On vellum; red staves with black Gregorian capitals and rubrication.
Dimension of leaves is 37cm x 41cm.
MBASE Laboratory

• 15 software engineering projects/year
  - 5-person USC Digital Library applications

• Rapidly developing successful applications
  - Multimedia, virtual assistants, data acquisition

• Integrating models and tools
  - DARPA-EDCS architecture and WinWin tools
  - Rational Rose, Unified Modeling Language

• Rapidly improving artifact integration
  - 1996 integrated specs, plans: 160 pages
  - 1997 integrated specs, plans: 110 pages

• Results transitioning to early adopters

• Ultimate goal: Model-integrated SW Engr. agents
## Case Study: CCPDS-R Project Overview

<table>
<thead>
<tr>
<th>Characteristic</th>
<th>CCPDS-R</th>
</tr>
</thead>
<tbody>
<tr>
<td>Domain</td>
<td>Ground based C3 development</td>
</tr>
<tr>
<td>Size/language</td>
<td>1.15M SLOC Ada</td>
</tr>
<tr>
<td>Average number of people</td>
<td>75</td>
</tr>
<tr>
<td>Schedule</td>
<td>75 months</td>
</tr>
<tr>
<td>Process/standards</td>
<td>DOD-STD-2167A Iterative development</td>
</tr>
<tr>
<td>Environment</td>
<td>Rational host</td>
</tr>
<tr>
<td></td>
<td>DEC host</td>
</tr>
<tr>
<td></td>
<td>DEC VMS targets</td>
</tr>
<tr>
<td>Contractor</td>
<td>TRW</td>
</tr>
<tr>
<td>Customer</td>
<td>USAF</td>
</tr>
<tr>
<td>Current status</td>
<td>Delivered On-budget, On-schedule</td>
</tr>
</tbody>
</table>
CCPDS-R MBASE Models

• Success Models
  – Reinterpreted DOD-STD-2167a; users involved
  – Award fee flowdown to performers

• Product Models
  – Domain model and architecture
  – Message-passing middleware (UNAS)

• Process Models
  – Ada process model and toolset
  – Incremental builds; early delivery

• Property Models
  – COCOMO cost & schedule
  – UNAS - based performance modeling
  – Extensive progress and quality metrics tools
Common Subsystem Macroprocess

Development Life Cycle

<table>
<thead>
<tr>
<th>Inception</th>
<th>Elaboration</th>
<th>Construction</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Architecture Iterations</td>
<td>Release Iterations</td>
</tr>
<tr>
<td></td>
<td>SSR, IPDR, PDR, CDR</td>
<td></td>
</tr>
</tbody>
</table>

Competitive design phase:
- Architectural prototypes
- Planning
- Requirements analysis

Contract award (LCO)

Architecture baseline under change control (LCA)

Early delivery of “alpha” capability to user
Early Adopters

- Rational, Xerox, FAA
- Air Force C2ISR Center
  - Field initial new C2ISR capabilities in 18 months
  - Determine, support common spiral model
  - General Officers’ Offsite Feb. 17-18, 1999
    - LG’s Kadish, Donahue, Martin
    - MG’s Cliver, Hawley, Carlson, Hess
  - Adopt WinWin Spiral Model as baseline
  - Revise draft AFI 63-123,
    “Evolutionary Acquisition for C2 Systems”
Conclusions

• Research on several classes of models is synergetic
  – COCOMO II, WinWin, spiral extensions, architecture research each helped by each other

• MBASE proving successful in avoiding model clashes

• USC-CSE Affiliate support has been essential
Further Information


USC-CSE Web-Site.  http://sunset.usc.edu