The Future of Software Processes, with Implications for Cost and Schedule Estimation

Barry Boehm, USC
COCOMO/SCM 14 Forum Tutorial
October 26, 1999

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

Outline

- Process Trends and Estimation Implications
  - Process Segmentation
  - Product line processes
  - Concurrent engineering
  - RAD and SAIV

- Conclusions
Future Software Trends

- Increased complexity
  - Everything connected
  - Opportunities for chaos (agents)
  - Systems of systems
- Decreased control of content
  - Infrastructure
  - COTS components
- Faster change
  - Time-to-market pressures
  - Marry in haste; no leisure to repent
  - Adapt or die (e-commerce)
- Fantastic opportunities
  - Personal, corporate, national, global

Usual Hardware-Software Trend Comparison

- Different counting rules
- Try counting software as Lines of Code in Service
  \[
  \text{LOCs/platform} = \sum (\#\text{platforms}) \times (\#\text{object LOCs/platform})
  \]

Trends in Software Expansion (Bernstein, 1997)
The Boom in User Programming

User programming
Repeat of "everyone a phone operator" trend
55M Users in US in year 2005

<table>
<thead>
<tr>
<th>Application generators (0.6M)</th>
<th>Applications composition (0.7M)</th>
<th>System Integration (0.7M)</th>
<th>Infrastructure (0.75M)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Process Model Decision Table

<table>
<thead>
<tr>
<th>Objectives, Constraints</th>
<th>Alternatives</th>
<th>Model</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>Growth Envelope</td>
<td></td>
<td>Buy COTS</td>
<td>Simple Inventory Control</td>
</tr>
<tr>
<td>Understanding of Reqts.</td>
<td>Limited</td>
<td>Transform or Evol. Devel.</td>
<td>Small Business - DP Application</td>
</tr>
<tr>
<td>Robustness</td>
<td>Limited</td>
<td>Evol. Prototype</td>
<td>Advanced Pattern Recognition</td>
</tr>
<tr>
<td>Available Technology</td>
<td>Limited</td>
<td>Waterfall</td>
<td>Rebuild of old system</td>
</tr>
<tr>
<td>Architecture Understanding</td>
<td>Limited to Large</td>
<td>Risk Reduction followed by Waterfall</td>
<td>Complex Situation Assessment</td>
</tr>
<tr>
<td></td>
<td>Low</td>
<td>High</td>
<td>High-performance Avionics</td>
</tr>
<tr>
<td></td>
<td>High</td>
<td>High</td>
<td>Evolutionary Development</td>
</tr>
<tr>
<td></td>
<td>Low-Medium</td>
<td>High</td>
<td>Data Exploitation</td>
</tr>
<tr>
<td></td>
<td>Limited to Medium</td>
<td>High</td>
<td>Medium to High</td>
</tr>
<tr>
<td></td>
<td>Low</td>
<td>Medium to High</td>
<td>Capabilities-to-Requirements</td>
</tr>
<tr>
<td></td>
<td>Medium to Large</td>
<td>High</td>
<td>Electronic Publishing</td>
</tr>
<tr>
<td></td>
<td>Very Large</td>
<td>High</td>
<td>Air Traffic Control</td>
</tr>
<tr>
<td></td>
<td>Medium to Large</td>
<td>Low</td>
<td>Software Support Environment</td>
</tr>
</tbody>
</table>

©USC-CSE
COCOMO II Coverage of Future SW Practices Sectors

- User Programming: No need for cost model
- Applications Composition: Use application points
  - Count (weight) screens, reports, 3GL routines
- System Integration; development of applications generators and infrastructure software
  - Prototyping: Applications composition model
  - Early design: Function Points and/or Source Statements and 7 cost drivers
  - Post-architecture: Source Statements and/or Function Points and 17 cost drivers
  - Stronger reuse/reengineering model

### Baseline Application Point Estimation Procedure

<table>
<thead>
<tr>
<th>Step</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Assess elements in the project: screens, filters, and 3GL components; then calculate the cost of the application. Assume the standard definitions of the elements in your ICASE environment.</td>
</tr>
<tr>
<td>2</td>
<td>Classify each element instance into simple, medium, or difficult complexity levels depending on values of characteristic dimensions. Use the following scheme:</td>
</tr>
<tr>
<td>3</td>
<td>Weigh each element instance using the following scheme. The weights reflect the relative effort required to implement an instance of that complexity level.</td>
</tr>
<tr>
<td>4</td>
<td>Determine Application Points: add all the weighted element instances to get one number, the Application Point.</td>
</tr>
<tr>
<td>5</td>
<td>Estimate the percentage of reuse you expect to be achieved in this project. Compute the New Application Points to be developed: NAP = Application Points \times (100 - \text{reuse}) / 100.</td>
</tr>
<tr>
<td>6</td>
<td>Determine a productivity index, PROD, using the following scheme:</td>
</tr>
<tr>
<td>7</td>
<td>Compute the estimated person-months: PM = \text{NAP} / \text{PROD}.</td>
</tr>
</tbody>
</table>

#### End of Baseline Application Point Estimation Procedure
**Relations to MBASE°/Rational Anchor Point Milestones**

- **App. Compos.:** Inception, Elaboration, Construction, Transition
  - LCO, LCA
  - IOC

  - SRR, PDR, Development
  - IOC

*MBASE: Model-Based (System) Architecting and Software Engineering*

**Early Design and Post-Arch Model**

- **Effort:**
  
  \[ PM_{estimated} = A \times (\text{Size}) \times \prod \text{SF} \times \prod \text{EM} \]

- **Size**
  - KSLOC (Thousands of Source Lines of Code)
  - UFP (Unadjusted Function Points) \* KSLOC/UFP
  - KSLOC/UFP factor varies by language

- **SF:** Scale Factors (5)
- **EM:** Effort Multipliers (7 for ED, 17 for PA)
New Scaling Exponent Approach

- Nominal person-months = A*(size)**B
- B = 0.91 + 0.01 ∑(exponent driver ratings)
  - B ranges from 0.91 to 1.23
  - 5 drivers; 6 rating levels each
- Exponent drivers:
  - Precedentness
  - Development flexibility
  - Architecture/ risk resolution
  - Team cohesion
  - Process maturity (derived from SEI CMM)

Project Scale Factors

\[ PM_{\text{estimated}} = 3.67 \times (\text{Size})^{SF} \times \prod_{i} EM_{i} \]

\[ SF = 0.91 + 0.01 \times \sum w_{i} \]

<table>
<thead>
<tr>
<th>Scale Factors</th>
<th>Very Low</th>
<th>Low</th>
<th>Nominal</th>
<th>High</th>
<th>Very High</th>
<th>Extra High</th>
</tr>
</thead>
<tbody>
<tr>
<td>PREC</td>
<td>thorough</td>
<td>markedly</td>
<td>somewhat</td>
<td>generally</td>
<td>typically</td>
<td>thoroughly</td>
</tr>
<tr>
<td>FLEX</td>
<td>vigorous</td>
<td>substantially</td>
<td>some</td>
<td>generally</td>
<td>some</td>
<td>generally</td>
</tr>
<tr>
<td>RESL</td>
<td>less (20%)</td>
<td>some (40%)</td>
<td>usually</td>
<td>generally</td>
<td>almost</td>
<td>fully (70%)</td>
</tr>
<tr>
<td>TEAM</td>
<td>very difficult</td>
<td>some difficult</td>
<td>occasionally</td>
<td>some difficult</td>
<td>generally</td>
<td>completely</td>
</tr>
<tr>
<td>PMAT</td>
<td>weighted sum of 18 KPA achievement levels</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

10/26/99

10/26/99

OUSC-CSE

OUSC-CSE
Calibration: COCOMO II.1997 Vs.2000

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></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, Bayesian</td>
<td>90% Experts</td>
</tr>
</tbody>
</table>

PRED (.30) Values
- Effort
  - By Org'n 81% 52% 75%
- Schedule 65% 61% 72%
  - By Org'n 62% 81%

COCOMO II Family

<table>
<thead>
<tr>
<th>Model</th>
<th># Drivers</th>
<th>Sizing</th>
</tr>
</thead>
<tbody>
<tr>
<td>Application Composition</td>
<td>2</td>
<td>Application Points</td>
</tr>
<tr>
<td>Early Design</td>
<td>7</td>
<td>Function Points or SLOC</td>
</tr>
<tr>
<td>Post-Architecture</td>
<td>17</td>
<td>Function Points or SLOC</td>
</tr>
<tr>
<td>COCOMO 81</td>
<td>15</td>
<td>SLOC (FP Extensions)</td>
</tr>
</tbody>
</table>

©USC-CSE 10/26/99 15
COCOMO Model Comparisons

<table>
<thead>
<tr>
<th>Metric</th>
<th>COCOMO</th>
<th>Asin COCOMO</th>
<th>COCOMO II (Semi-Custom)</th>
<th>COCOMO II (Rapid-Change)</th>
<th>COCOMO II (Fast-Architecture)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Size</td>
<td>Delivered Source Instructions (DSI) or Source Lines of Code (SLOC)</td>
<td>DSIs or SLOC</td>
<td>Application Points</td>
<td>Functions Points (FP) and Language or SLOC</td>
<td>Functions Points (FP) and Language or SLOC</td>
</tr>
<tr>
<td>Metrics</td>
<td>Equivalent SLOC + Lines (DSI+CM, DM)</td>
<td>Equivalent SLOC + Lines (DSI+CM, DM)</td>
<td>Equivalent SLOC + Lines (DSI+CM, DM)</td>
<td>Equivalent SLOC + Lines (DSI+CM, DM)</td>
<td>Equivalent SLOC + Lines (DSI+CM, DM)</td>
</tr>
<tr>
<td>Repe. Charge</td>
<td>Requirements Volatility rating (RVOL)</td>
<td>RVOL rating</td>
<td>Implicit in Model</td>
<td>Change %: RQEV</td>
<td>RQEV</td>
</tr>
<tr>
<td>Measurement</td>
<td>Annual Change Traffic (ACT) +</td>
<td>Adjusted + Adjusted</td>
<td>ACT</td>
<td>Object Prior ACT</td>
<td>[ACT:SU] (RVOL)</td>
</tr>
<tr>
<td>Scale (N/A)</td>
<td>Original: 105 Intra-reduced: 10</td>
<td>Embodied: 1:4:1:24 depending on degree of: early risk estimation, product lines, product requirements, and project maturity</td>
<td>11 = 9.1123 depending on the degree of: project size, time, effort, memory, data, and project complexity</td>
<td>Project maturity level</td>
<td>Project maturity level</td>
</tr>
<tr>
<td>Product Cost Drivers</td>
<td>RELY, DATA, CPILX</td>
<td>RELY, DATA, CPILX, EMA</td>
<td>None</td>
<td>RQEV + AIRE +</td>
<td>RELY, DATA, TOOL</td>
</tr>
<tr>
<td>Software Cost Drivers</td>
<td>TIME, STOR, TYPE, TURN</td>
<td>TIME, STOR, VM, VMH, VMH, TYPE, TURN</td>
<td>None</td>
<td>Platform difficulties (PDB)</td>
<td>TIME, STOR, VMH, VMH, TYPE, TURN</td>
</tr>
<tr>
<td>Premium Cost Drivers</td>
<td>ADEP, ADEP, FCAP, VEYF, LEAP</td>
<td>ADEP, ADEP, FCAP, VEYF, LEAP</td>
<td>None</td>
<td>Project capability and experience (PCE), PRED, PRED</td>
<td>ADEP, ADEP, FCAP, VEYF, LEAP, TYPE, PRED</td>
</tr>
<tr>
<td>Project Cost Drivers</td>
<td>MODE, TOOL, SCIO</td>
<td>MODE, TOOL, SCIO</td>
<td>None</td>
<td>SCIO, PCE</td>
<td>TOOL, SCIO, SCIO</td>
</tr>
</tbody>
</table>

* Different Multiples of Different Rating Scale

10/26/99 ©USC-CSE

Outline

- Process Trends and Estimation Implications
  - Process Segmentation
  - Product line processes
  - Concurrent engineering
  - RAD and SAIV

- Conclusions

10/26/99 ©USC-CSE
Reuse and Product Line Management

- **Challenges**
  - Estimate costs of both reusing software and developing software for future reuse
  - Estimate extra effects on schedule (if any)

- **Responses**
  - New nonlinear reuse model for effective size
  - Cost of developing reusable software estimated by RUSE effort multiplier
  - Gathering schedule data

Nonlinear Reuse Effects

Data on 2954 NASA modules [Selby, 1988]
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)
  - $\text{ESLOC} = \text{ASLOC}[\text{AA+AAF}(1+0.02(\text{SU})(\text{UNFM}))]$
  - $\text{AAF} < 0.5$

$$\text{AAF} = 0.4(\text{DM}) + 0.3(\text{CM}) + 0.3(\text{IM})$$

Results in revised Equivalent Source Lines of Code ($\text{ESLOC}$)

Software Understanding Rating / Increment

<table>
<thead>
<tr>
<th>Structure</th>
<th>Very Low</th>
<th>Low</th>
<th>Medium-Low</th>
<th>Medium-High</th>
<th>High</th>
<th>Very High</th>
</tr>
</thead>
<tbody>
<tr>
<td>Applicability</td>
<td>No match between program and application context.</td>
<td>Some correlation between program and application.</td>
<td>Good correlation between program and application.</td>
<td>Chose match between program and application context.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Clarity</td>
<td>Very poor; spaghetti code.</td>
<td>Moderate level of clarity; useful documentation.</td>
<td>Good level of clarity; useful documentation; some weak areas.</td>
<td>Very good level of clarity; useful documentation; well-organized, well-commented code.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Documentation</td>
<td>Program documentation missing; obscure or obsolete.</td>
<td>Some code with comments and headers, some useful documentation.</td>
<td>Moderate level of code commentary and headers; documentation useful, descriptive, well-organized.</td>
<td>Good code commentary and headers; documentation well-organized, well-commented, with design plans.</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>SU Increment in ESLOC</th>
<th>0</th>
<th>10</th>
<th>20</th>
<th>30</th>
<th>40</th>
</tr>
</thead>
</table>

©USC-CSE 21
Cost to Develop for Reuse 100 KSLOC Manufacturing Control Systems

<table>
<thead>
<tr>
<th>Effort Multipliers</th>
<th>VL</th>
<th>L</th>
<th>N</th>
<th>H</th>
<th>VH</th>
<th>XH</th>
</tr>
</thead>
<tbody>
<tr>
<td>Developed for Reuse</td>
<td>.95</td>
<td>1.0</td>
<td>1.07</td>
<td>1.15</td>
<td>1.24</td>
<td></td>
</tr>
<tr>
<td>Required Reliability</td>
<td>.92</td>
<td>.92</td>
<td>1.0</td>
<td>1.10</td>
<td>1.26</td>
<td></td>
</tr>
<tr>
<td>Required Documentation</td>
<td>.91</td>
<td>.91</td>
<td>1.0</td>
<td>1.11</td>
<td>1.23</td>
<td></td>
</tr>
</tbody>
</table>

- Full Cost: $2.94(100)^{1.10}(1.18) = $4400K
- International Factory Reuse: halfway between VH and XH
- Recommended Reliability rating: 1 level lower
- Recommended Documentation rating: High
- Develop for Reuse: $4400(1.195)(1.18)(1.11) = $6824K

Subsequent Development-with-Reuse

Black-box plug-and-play: 30 KSLOC
Reuse with modifications: 30 KSLOC
New factory-specific SW: 40 KSLOC
Assessment and assimilation (AA): 2%
Software understanding factor (SU): 10%
Unfamiliarity factor (UNFM): 0.3%
% design modified (DM): 10%
% code modified (CM): 20%
% integration modified (IM): 20%
AAF = .4(.10) + .3 (.20) + .1 (.16) = .16

ESLOC = 40 + (30)(.02) + (30)(.02 + .3)(.16)
= 40 + 0.6 + 6.3 = 46.9

COST = 2.94(46.9)^{1.18}(1.18)(1.1)(58K) = $9266K
Summary: Reuse vs. Redevelopment Costs

<table>
<thead>
<tr>
<th>Number of Factories</th>
<th>Redevelopment Cost</th>
<th>Product Line Cost</th>
<th>Investment Return</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>$4,000</td>
<td>$6,824</td>
<td>-$2,824</td>
</tr>
<tr>
<td>2</td>
<td>5,400</td>
<td>9,790</td>
<td>-990</td>
</tr>
<tr>
<td>3</td>
<td>13,200</td>
<td>12,756</td>
<td>444</td>
</tr>
<tr>
<td>4</td>
<td>17,600</td>
<td>15,722</td>
<td>1,878</td>
</tr>
</tbody>
</table>

Outline

- Process Trends and Estimation
  - Process Segmentation
  - Product line processes
  - Concurrent engineering
    - RAD and SAIV
- Conclusions
Concurrent Engineering

- Problems with sequential engineering
  - Model clashes
- Spiral, WinWin Spiral, MBASE
  - Anchor points and Rational Unified Process
- Implications for cost and schedule estimation

Problems with Sequential Engineering

- Takes too long
- Buries risk
- HCI systems are “emergent”
  - IKIWISI: I’ll know it when I see it
- Breeds model clashes
  - 2167A, COTS, Ada, 4GL’s, CAIV
Sequential Engineering Buries Risk

![Graph showing the comparison between two architectures: Arch A with custom many cache processors and Arch B with modified client-server. The graph illustrates the cost and response time improvements after prototyping.](image)

Understanding the Tar Pit: Model Clashes

- **Model Clash**: An incompatibility among the underlying assumptions of a set of models
  - Often unrecognized
  - Produces conflicts, confusion, mistrust, frustration, rework, throwaway systems

Copyright © USC-CSE 2000
The army warfighting experiment process has created a development method that could result in shortening the time for fielding equipment from seven to 10 years down to two or three.

"When you say spiral development, you're talking more than just hardware and software," said Col. Joa Leigh, commander of the digital force coordination cell at Fort Hood, Texas.

"You're talking about the development of hardware, software, training, leader development, TTP (tactics, techniques, and procedures) development and, to some degree, doctrinal development. That spiral development is very powerful."

Alternative to spiral development is to continue equipment development in its current linear, step-by-step method, according to Gen. William W. Hartzog, command of Training and Doctrine Command.

"When a concept or design is generated by one part of the partnership, it is studied and used by the other members.

Through feedback from the entire partnership, the concept or equipment is improved."

Spiral development is vital to creating future army

by Jim Caldwell
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 stakeholders' objectives
2. Identify stakeholders' constraints
3. Define project success
4. Identify project risk
5. Define project scope
6. Define project schedule

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
Model-Based Process Tailoring (MBASE)

Success Models Drive Other Model Choices

<table>
<thead>
<tr>
<th>Success Model</th>
<th>Demo agent-based E-commerce system at COMDEX in 9 months</th>
<th>Safe air traffic control system</th>
</tr>
</thead>
<tbody>
<tr>
<td>Key Stakeholders</td>
<td>Entrepreneurs, venture capitalists, customers</td>
<td>Controllers, Govt. agencies, developers</td>
</tr>
<tr>
<td>Key Property Models</td>
<td>Schedule estimation</td>
<td>Safety models</td>
</tr>
<tr>
<td>Process Model</td>
<td>Design-to-schedule</td>
<td>Initial spiral to risk-manage COTS, etc.; Final waterfall to verify safety provisions</td>
</tr>
<tr>
<td>Product Model</td>
<td>Domain constrained by schedule; architected for ease in dropping features to meet schedule</td>
<td>Architected for fault tolerance, ease of safety verification</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

MBASE Project Experience at USC/Columbia

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Fall Semester: LCA Package</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Teams</td>
<td>15</td>
<td>16</td>
<td>20</td>
<td>20</td>
<td>13</td>
</tr>
<tr>
<td>Students</td>
<td>86</td>
<td>80</td>
<td>102</td>
<td>107</td>
<td>59</td>
</tr>
<tr>
<td>Applications</td>
<td>12</td>
<td>15</td>
<td>17</td>
<td>10</td>
<td>10</td>
</tr>
<tr>
<td>Teams failing LCA review</td>
<td>4</td>
<td>4</td>
<td>1</td>
<td>10</td>
<td>6</td>
</tr>
<tr>
<td>Teams failing LCA review</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>Pages, LCA package</td>
<td>160</td>
<td>103</td>
<td>114</td>
<td>124</td>
<td>116</td>
</tr>
<tr>
<td>Pages, LCA package</td>
<td>230</td>
<td>154</td>
<td>167</td>
<td>142</td>
<td>142</td>
</tr>
<tr>
<td>Client Evaluation (1-6, 5 best)</td>
<td>4.49</td>
<td>4.67</td>
<td>4.74</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Spring Semester: IOC Package</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Teams</td>
<td>6</td>
<td>5</td>
<td>6</td>
<td>Remained the same since projects were only one semester long</td>
<td></td>
</tr>
<tr>
<td>Students</td>
<td>28</td>
<td>23</td>
<td>28</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Applications</td>
<td>5</td>
<td>5</td>
<td>6</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Teams failing IOC acceptance review</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Applications satisfying clients</td>
<td>5</td>
<td>5</td>
<td>6</td>
<td>20*</td>
<td>13*</td>
</tr>
<tr>
<td>Applications not overtaken by events</td>
<td>6</td>
<td>4</td>
<td>4</td>
<td>10</td>
<td>9</td>
</tr>
<tr>
<td>Applications continued</td>
<td>3</td>
<td>3</td>
<td>4</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Applications used</td>
<td>1</td>
<td>3</td>
<td>TBD</td>
<td>10</td>
<td>5</td>
</tr>
<tr>
<td>Client evaluation</td>
<td>-</td>
<td>4.15</td>
<td>4.3</td>
<td>4.44</td>
<td>4.51</td>
</tr>
</tbody>
</table>

10/26/99 ©USC-CSE
MBASE Electronic Process Guide (1)

- Developed in collaboration with CMU-SEI
- Using SEI EPG tool

MBASE Electronic Process Guide (2)
Concurrent Engineering

- Problems with sequential engineering
  - Model clashes
- Spiral, WinWin Spiral, MBASE
  - Anchor points and Rational Unified Process
- Implications for cost and schedule estimation

Relations to MBASE*/Rational Anchor Point Milestones

<table>
<thead>
<tr>
<th>Sys Devel</th>
<th>App. Compos.</th>
<th>Inception</th>
<th>Elaboration, Construction</th>
<th>Transition</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td>LCO, LCA</td>
<td>IOC</td>
</tr>
<tr>
<td>Waterfall</td>
<td>SRR</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Inception</td>
<td>Elaboration</td>
<td></td>
<td>Development</td>
<td></td>
</tr>
<tr>
<td>Phase</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>LCO</td>
<td>LCA</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>IOC</td>
<td>Trans.</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

*MBASE: Model-Based (System) Architecting and Software Engineering
### COCOMO II: MBASE Phase Distributions

**Representative 100 KSLOC Project**

**Effort = 2.94 \(100\) = 466 person-months**

**Schedule = 3.67 \(466\) = 26 months**

<table>
<thead>
<tr>
<th>Phase Distribution</th>
<th>Effort</th>
<th>Schedule</th>
<th>% PM</th>
<th>% Mo</th>
<th>% Staff</th>
</tr>
</thead>
<tbody>
<tr>
<td>Elaboration</td>
<td>0.24</td>
<td>0.24</td>
<td>112</td>
<td>112</td>
<td>112</td>
</tr>
<tr>
<td>Construction</td>
<td>0.76</td>
<td>0.76</td>
<td>354</td>
<td>354</td>
<td>354</td>
</tr>
</tbody>
</table>
Outline

• Process Trends and Estimation Implications
  – Process Segmentation
  – Product line processes
  – Concurrent engineering
  ➔ RAD and SAIV
• Conclusions

RAD and SAIV

• Rapid Application Development (RAD)
  - Opportunity Tree
• Schedule as Independent Variable (SAIV)
  - Agent-Based Processes
• Estimation Implications: CORADMO
**Future Software Trends**

- Increased complexity
  - Everything connected
  - Opportunities for chaos (agents)
  - Systems of systems
- Decreased control of content
  - Infrastructure
  - COTS components
- Faster change
  - Time-to-market pressures
  - Marry in haste; no leisure to repent
  - Adapt or die (e-commerce)
- Fantastic opportunities
  - Personal, corporate, national, global

**RAD Motivation**

- Earlier ROI
- Market Window
- Technology Half-Life
Complexity of RAD Improvements

- Cost: task savings basically map 1:1 into project savings
- Schedule: task savings map 1:1 into project savings only while task is on critical path
  - Complicating factors: scale, dynamism, interdependent tasks
  - System dynamics an attractive analysis approach
Schedule As Independent Variable (SAIV)

- Build a core capability early
- Drop additional features as necessary to meet schedule
  - Need prioritized features ("requirements").
  - Need an architecture enabling easily-dropped features.
  - Need good estimates, planning and control.
Pareto Analysis of Rework Costs

Ada Process Model for Large-System RAD

"ALL TOO FREQUENT PROCESS MODEL"

Ada PROCESS MODEL

10/29/99
Getting Tasks Off the Critical Path

- Decomposition and parallelization
  - Replace Critical Design Review by unit inspections
  - Pre-integrate subsystems
  - Parallelize off-nominal testing
  - Massive beta testing
  - Pre-work unit level acceptance tests
- Pre-positioning facilities, components, tools, experts, data
  - "Overinvesting" in reusable components
"Overinvesting" in Reusable Components

Cumulative Cost vs. Number of Similar Systems

Optimum for 2 similar systems:
- Cost
- Schedule for #2

Reuse at HP's Queensferry Telecommunication Division

Time to Market (months) vs. Year

Non-reuse Project
Reuse project
People and RAD

- Better and fewer people
  - Bright, quick, versatile, adaptable, creative, experienced, focused
- Clear RAD priorities and incentives
  - Early completion bonus and recognition
- Teambuilding and shared vision
  - All of the stakeholders
- Co-location
- Capitalization
- Learning, metrics, continuous improvement

Agent-Aided Processes- SEDA

- Develop normative software product, process, property, and success models
  - MBASE; initially in digital library domain
- Develop agents to detect, suggest potential model clashes and violations
- Experimentally apply, refine, and generalize agents
SEDA Model Clash Advisor: I

**Success Model - Property Model Clash**

You have a model clash between your
- Success Model: Complete development in 12 weeks
- and your
- Property Model: COCOMO II schedule estimate of 20 weeks.

You have two primary options to remove the model clash. They are complementary and can be pursued in parallel.
1. Adjust your project's cost/schedule driver decisions to produce a shorter estimated schedule (e.g., via reuse, better tools, more stable infrastructure). Click on the S-COST tool to get help in assessing your options.
2. Use a schedule-as-independent variable (SAIV) process model. Click on the SAIV section of the MBASE Guidelines for how-to information.

SEDA Model Clash Advisor: II

**Process Model - Product Model Clash**

You have a model clash between your
- Process Model: Schedule as Independent Variable (SAIV)
- and your
- Product Model: Unprioritized requirements.

If you proceed as-is and need to drop requirements to meet schedule, you will lose valuable time working out their relative priorities. It's best to do it now.
RAD and SAIV

- Rapid Application Development (RAD)
  - Opportunity Tree
- Schedule as Independent Variable (SAIV)
  - Agent-Based Processes
- Estimation Implications: CORADMO

Proposed COCOMO II RAD Extension (CORADMO)
Need to Improve Classic Schedule Model

Months ~ 2.5 \(\sqrt[3]{\text{Person-Months}}\)

- No reflection of RAD Opportunity Tree tradeoffs
  - Reuse, VHLL’s (RVHL)
  - Business Process Reengineering (BPRS)
  - Collaboration Technology (CLAB)
  - Architecture, Risk Resolution (RESL)
  - Preposition Assets (PPOS)

RAD Opportunity Tree

- Business process reengineering - BPRS
- Reusing assets - RVHL
- Applications generation - RVHL
- Design-to-schedule - O
- Tools and automation - O
- Work streamlining (80-20) - O
- Increasing parallelism - RESL
- Reducing failures - RESL
- Reducing their effects - RESL
- Early error elimination - RESL
- Process anchor points - RESL
- Improving process maturity - O
- Collaboration technology - CLAB
- Minimizing task dependencies - BPRS
- Avoiding high fan-in, fan-out - BPRS
- Reducing task variance - BPRS
- Removing tasks from critical path - BPRS
- 24x7 development - PPOS
- Nightly builds, testing - PPOS
- Weekend warriors - PPOS

Better People and Incentives - O
Transition to Learning Organization - constraint

©USC-CSE
Unsenny ol Southern California

COSSEMO Stage Distribution Model: Early Design/ Post Architecture

<table>
<thead>
<tr>
<th>Effort %</th>
<th>Inception</th>
<th>Elaboration</th>
<th>Construction</th>
<th>Total S&amp;C</th>
</tr>
</thead>
<tbody>
<tr>
<td>14</td>
<td>28</td>
<td>72</td>
<td>100</td>
<td></td>
</tr>
<tr>
<td>Schedule %</td>
<td>40</td>
<td>40</td>
<td>60%</td>
<td>100</td>
</tr>
<tr>
<td>$\delta / \delta P$</td>
<td>0.35</td>
<td>0.7</td>
<td>1.2</td>
<td>1.0</td>
</tr>
</tbody>
</table>

PM/M-P

| 32K | 66113.7 ± 35 | 723/13.7 ± 33 | 1858/20.6 ± 91 | 2580/34.3 ± 75 |
| 512K | 16.64/4.3 ± 3.5 | 33.61/4.8 ± 7 | 86.67/3.2 ± 12 | 149/12 ± 10 |

COSSEMO Duration Calculation

Months as F(PM)
CORADMO: Reuse and VHLL's (RVHL)

- Standard 3GL module reuse: no adjustment
- Schedule compression in Inception and Elaboration stages due to faster prototyping, option exploration
  - effect depends on experience in doing this
  - compression/expansion doesn't alter staff level

<table>
<thead>
<tr>
<th>Schedule Multipliers</th>
<th>Rapid Prototyping Experience</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>VL</td>
</tr>
<tr>
<td>Inception</td>
<td>1.05</td>
</tr>
<tr>
<td>Elaboration</td>
<td>1.03</td>
</tr>
<tr>
<td>Construction</td>
<td>1.0</td>
</tr>
</tbody>
</table>

Development Process Reengineering and Streamlining (DPRS)

Detailed rating scale provided
Gains depend on current level of bureaucracy

- Same effect on effort; staff level held constant

<table>
<thead>
<tr>
<th>Schedule and Effort Multipliers</th>
<th>Inception</th>
<th>Elaboration</th>
<th>Construction</th>
</tr>
</thead>
<tbody>
<tr>
<td>VL - Heavily Bureaucratic</td>
<td>1.20</td>
<td>1.15</td>
<td>1.15</td>
</tr>
<tr>
<td>L - Bureaucratic</td>
<td>1.08</td>
<td>1.06</td>
<td>1.06</td>
</tr>
<tr>
<td>N - Basic good business practices</td>
<td>1.0</td>
<td>1.0</td>
<td>1.0</td>
</tr>
<tr>
<td>H - Partly streamlined</td>
<td>.96</td>
<td>.98</td>
<td>.98</td>
</tr>
<tr>
<td>VH - Fully streamlined</td>
<td>.90</td>
<td>.95</td>
<td>.95</td>
</tr>
</tbody>
</table>
### DPRS Rating Scale

<table>
<thead>
<tr>
<th>Number of Requests required per task</th>
<th>VL</th>
<th>L</th>
<th>N</th>
<th>H</th>
<th>VH</th>
</tr>
</thead>
<tbody>
<tr>
<td>Time taken per approval</td>
<td>Extreme</td>
<td>Occasionally Reduced</td>
<td>Mature</td>
<td>Active</td>
<td>Diminished</td>
</tr>
<tr>
<td>Reduced task dependencies, critical path tasks</td>
<td>None</td>
<td>Little</td>
<td>Mature</td>
<td>Task Accepted</td>
<td>Diminished</td>
</tr>
<tr>
<td>Followup to expedite task completion</td>
<td>None</td>
<td>Little</td>
<td>Emphasized</td>
<td>Diminished</td>
<td>Streamlined</td>
</tr>
<tr>
<td>Process measurement &amp; streamlining</td>
<td>None</td>
<td>Little</td>
<td>Mature</td>
<td>Task Accepted</td>
<td>Diminished</td>
</tr>
</tbody>
</table>

### Collaboration Technology (CLAB)

- **Basic rating scale and multipliers provided**
- **Detailed rating scale complex, TBD**
  - Effects dependent on TEAM, SITE rating
  - Needs integration with TOOL rating
  - Collaboration tool maturity, experience, scope (domain, negotiation, option-analysis tool support)

**Same effect on effort; same staff level**

<table>
<thead>
<tr>
<th>Schedule Multipliers (Effort Unchanged)</th>
<th>VL</th>
<th>L</th>
<th>N</th>
<th>H</th>
<th>VH</th>
<th>EH</th>
</tr>
</thead>
<tbody>
<tr>
<td>Inception</td>
<td>1.31</td>
<td>1.10</td>
<td>1.0</td>
<td>.93</td>
<td>.86</td>
<td>.80</td>
</tr>
<tr>
<td>Elaboration</td>
<td>1.15</td>
<td>1.07</td>
<td>1.0</td>
<td>.95</td>
<td>.80</td>
<td>.86</td>
</tr>
<tr>
<td>Construction</td>
<td>1.10</td>
<td>1.05</td>
<td>1.0</td>
<td>.98</td>
<td>.95</td>
<td>.93</td>
</tr>
</tbody>
</table>
Architecture / Risk Resolution (RESL)

Same as COCOMO II RESL rating scale

Enables parallel construction

- Assumes higher level of staffing available and used
- Otherwise no schedule compression

<table>
<thead>
<tr>
<th>Schedule Multipliers (Effort Unchanged)</th>
<th>VL</th>
<th>L</th>
<th>N</th>
<th>H</th>
<th>VH</th>
<th>EH</th>
</tr>
</thead>
<tbody>
<tr>
<td>Inception</td>
<td>1.0</td>
<td>1.0</td>
<td>1.0</td>
<td>1.0</td>
<td>1.0</td>
<td>1.0</td>
</tr>
<tr>
<td>Elaboration</td>
<td>1.0</td>
<td>1.0</td>
<td>1.0</td>
<td>1.0</td>
<td>1.0</td>
<td>1.0</td>
</tr>
<tr>
<td>Construction</td>
<td>1.0</td>
<td>1.0</td>
<td>1.0</td>
<td>.91</td>
<td>.83</td>
<td>.75</td>
</tr>
</tbody>
</table>

Prepositioning Assets (PPOS)

Degree to which assets are pre-tailored to project and furnished to project for use on demand

- People skills and teambuilding
- Processes and tools
- Architecture and componentry

<table>
<thead>
<tr>
<th>PM/M=P Multipliers</th>
<th>N</th>
<th>H</th>
<th>VH</th>
<th>EH</th>
</tr>
</thead>
<tbody>
<tr>
<td>Rating</td>
<td>Basic project legacy, no tailoring</td>
<td>Some prepositioning &amp; tailoring</td>
<td>Key items prepositioned &amp; tailored</td>
<td>All items prepositioned &amp; tailored</td>
</tr>
<tr>
<td>Inception</td>
<td>1.15/1.15=1.0</td>
<td>1.0/1.0=1.0</td>
<td>1.06/1.95=1.25</td>
<td>1.2/1.85=1.41</td>
</tr>
<tr>
<td>Elaboration</td>
<td>1.0/1.10=1.0</td>
<td>1.0/1.0=1.0</td>
<td>1.06/1.95=1.25</td>
<td>1.2/1.85=1.41</td>
</tr>
<tr>
<td>Construction</td>
<td>1.05/1.05=1.0</td>
<td>1.0/1.0=1.0</td>
<td>1.06/1.95=1.25</td>
<td>1.2/1.85=1.41</td>
</tr>
</tbody>
</table>
People Capabilities as Constraint

- Used Early Design Personnel Capability Rating (ACAP, PCAP, PCON, Merge) as constraint
- RAD driver ratings can't be higher than personnel capability rating

<table>
<thead>
<tr>
<th>Multipliers</th>
<th>Inception</th>
<th>Elaboration</th>
<th>Construction</th>
</tr>
</thead>
<tbody>
<tr>
<td>RVHL</td>
<td>0.92/0.92=1.00</td>
<td>0.95/0.95=1.00</td>
<td>1.00/1.00=1.00</td>
</tr>
<tr>
<td>BPRS</td>
<td>0.92/0.92=1.00</td>
<td>0.95/0.95=1.00</td>
<td>0.95/0.95=1.00</td>
</tr>
<tr>
<td>CLAB</td>
<td>0.80/0.80=1.00</td>
<td>0.86/0.86=1.00</td>
<td>0.93/0.93=1.00</td>
</tr>
<tr>
<td>RESL</td>
<td>1.00/1.00=1.00</td>
<td>1.00/1.00=1.00</td>
<td>1.00/0.75=1.00</td>
</tr>
<tr>
<td>PPOS</td>
<td>1.20/0.85=1.00</td>
<td>1.20/0.85=1.00</td>
<td>1.20/0.85=1.41</td>
</tr>
<tr>
<td>k</td>
<td>0.81/0.56=1.41</td>
<td>0.93/0.66=1.41</td>
<td>1.06/0.56=1.88</td>
</tr>
</tbody>
</table>
**Application Development Example -1**

- **32 KSLOC Project (PM/ M = P)**

<table>
<thead>
<tr>
<th></th>
<th>Inception</th>
<th>Elaboration</th>
<th>Construction</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>Baseline</td>
<td>16.8/4.8 = 3.5</td>
<td>3.6/4.8 = 7.0</td>
<td>86.4/7.2 = 12</td>
<td>120/12 = 10</td>
</tr>
<tr>
<td>TT</td>
<td>.81/.58 = 1.41</td>
<td>.93/.66 = 1.41</td>
<td>1.06/.56 = 1.88</td>
<td>1.02/.6 = 1.70</td>
</tr>
<tr>
<td>RAD</td>
<td>13.6/2.8 = 4.8</td>
<td>31.2/3.2 = 9.8</td>
<td>91.6/4 = 23</td>
<td>123/7.2 = 17</td>
</tr>
<tr>
<td></td>
<td>136/10 = 13.6</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- **512KSLOC Project (PM/ M = P)**

<table>
<thead>
<tr>
<th></th>
<th>Inception</th>
<th>Elaboration</th>
<th>Construction</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>Baseline</td>
<td>361/13.7 = 26</td>
<td>722/13.7 = 53</td>
<td>1868/20.6 = 90</td>
<td>2580/34.3 = 75</td>
</tr>
<tr>
<td>TT</td>
<td>.81/.58 = 1.41</td>
<td>.93/.66 = 1.41</td>
<td>1.06/5.6 = 1.88</td>
<td>1.02/.6 = 1.70</td>
</tr>
<tr>
<td>RAD</td>
<td>292/7.95 = 37</td>
<td>671/9 = 74</td>
<td>1969/11.5 = 171</td>
<td>2640/28.5 = 129</td>
</tr>
<tr>
<td></td>
<td>2933/28.2 = 103</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Conclusions**

- Software field experiencing tremendous (dizzying) rate of growth and change
- Estimation models need to continuously recalibrate and re-engineer
- Software organizations need to continuously recalibrate and re-engineer
- Need good metrics, closed-loop feedback process
Using COCOMO II to Cope With Change: I

System objectives: functionality, performance, quality

Project Parameters: Personnel, team sites, platform

Corporate parameters: tools, processes, reuse

Rescope

COCOMO II

Cost, Schedule, risks

OK?

Yes

No

Using COCOMO II to Cope With Change: II

System objectives: functionality, performance, quality

Input Parameters: Function, scale, class, pattern

Corporate parameters: code, procedures, specs

Rescope

COCOMO II

Cost, Schedule, risks

Exercise project B-level variables

Rescope

Exercise project C-level variables

Yes

No

End Project

Exercise project D-level variables

Start execution

Yes

No

Doyle?
Using COCOMO II to Cope With Change: III

2. Project Parameters: project, team, site, platform.
3. Corporate parameters: tools, processes, reuse.
4. Recalibrate or extend COCOMO II.
5. Rescope.
7. Minimize rework.
8. Does COCOMO II fit this project?
9. Execute project milestones.
10. Senior management, tools, resources.
11. End Project.
12. Retain expenditures.
13. Does COCOMO II calibration data fit this project?
14. Execute project milestones.
15. Senior management, tools, resources.
16. End Project.
17. If yes, then:
18. Recompute COCOMO II.
20. Minimize rework.
21. Does COCOMO II fit this project?
22. Execute project milestones.
23. Senior management, tools, resources.
24. End Project.
25. If no, then:
26. Evaluate Corporate SW Improvement Strategies.
27. Recompute COCOMO II.
28. Cost, schedule, quality.
29. Minimize rework.
30. Does COCOMO II fit this project?
31. Execute project milestones.
32. Senior management, tools, resources.
33. End Project.
34. If no, then:
35. Retain expenditures.
36. End Project.

Using COCOMO II to Cope With Change: IV

2. Project Parameters: project, team, site, platform.
3. Corporate parameters: tools, processes, reuse.
4. Evaluate Corporate SW Improvement Strategies.
5. Improved Corporate Parameters.
6. Recompute COCOMO II.
7. Cost, schedule, quality.
8. Minimize rework.
9. Does COCOMO II fit this project?
10. Execute project milestones.
11. Senior management, tools, resources.
12. End Project.
13. If yes, then:
14. Recompute COCOMO II.
15. Cost, schedule, quality.
17. Does COCOMO II fit this project?
18. Execute project milestones.
19. Senior management, tools, resources.
20. End Project.
21. If no, then:
22. Evaluate Corporate SW Improvement Strategies.
23. Recompute COCOMO II.
25. Minimize rework.
26. Does COCOMO II fit this project?
27. Execute project milestones.
28. Senior management, tools, resources.
29. End Project.
30. If no, then:
31. Retain expenditures.
32. End Project.