Productivity Decline in Directed System of Systems Software Development

Ramin Moazeni
A. Winsor Brown
Barry Boehm
Table of Contents

• Incremental Commitment Model (ICM)
  – Overview
  – Multi-Build Software and Overlap across builds
  – Directed Systems of Systems
  – Systems
  – COINCOMO

• Incremental Development Productivity Decline (IDPD)
  – Overview
  – Cost Drivers
  – Effect on number of increments

• Conclusion
ICM LC Processes for Systems

<table>
<thead>
<tr>
<th>Phases (EVADO)</th>
<th>Activities</th>
<th>Exploration</th>
<th>Valuation</th>
<th>Architecting</th>
<th>Development 1</th>
<th>Architecting 2</th>
<th>Development 2</th>
<th>Architecting 3</th>
<th>Development 3</th>
<th>Architecting 4</th>
<th>Operations 1</th>
<th>Operations 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>DoD, General/DoD Milestones</td>
<td>Initial Scoping</td>
<td>Concept Definition, Investment Analysis</td>
<td>Software System Architecting</td>
<td>Increment 1 Development</td>
<td>Increment 2 Architecting Rebaseline</td>
<td>Increment 2 Development</td>
<td>Increment 3 Architecting Rebaseline</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td></td>
</tr>
</tbody>
</table>

Concurrent Risk-and-Opportunity-Driven Growth of System Understanding and Definition

Evaluation of Evidence of Feasibility to Proceed

Stakeholder Review and Commitment

Risk? Acceptable High, but Addressable Too High, Unaddressable

Risk? Acceptable High, but Addressable Too High, Unaddressable

Risk? Acceptable High, but Addressable Too High, Unaddressable

Risk? Acceptable High, but Addressable Too High, Unaddressable

Risk? Acceptable High, but Addressable Too High, Unaddressable

Risk? Acceptable High, but Addressable Too High, Unaddressable

Risk? Acceptable High, but Addressable Too High, Unaddressable

Adjust Scope, Priorities, or Discontinue
ICM-Sw/RUP Activity/Process Model

<table>
<thead>
<tr>
<th>Phases</th>
<th>Inception</th>
<th>Elaboration</th>
<th>Construction</th>
<th>Transition</th>
</tr>
</thead>
<tbody>
<tr>
<td>Disciplines</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Business Modeling</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Requirements</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Analysis &amp; Design</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Implementation</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Test</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Deployment</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Configuration &amp; Change Mgmt</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Project Management</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Environment</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Iterations</th>
<th>Initial</th>
<th>Elab #1</th>
<th>Elab #2</th>
<th>Const #1</th>
<th>Const #2</th>
<th>Const #N</th>
<th>Tran #1</th>
<th>Tran #2</th>
</tr>
</thead>
<tbody>
<tr>
<td>VCR: Valuation Commitment Review</td>
<td>DCR: Development Commitment Review</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>FCR: Foundations Commitment Review</td>
<td>OCR: Operations Commitment Review</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Why Multiple Build Software Systems

Simplest:  Early Functionality in the hands of ALL users

• Architecture/Core plus some functionality
• Implies Full Qualification/Acceptance Sw Testing each software build so systems can go into Integration & Test earlier

Increasingly Complex Systems

• Multiple, diverse "platforms"
• Different "foci" of functionality (in each build)
• Network Centric Systems Operation
• Evolution/federation of legacy systems
• System of Systems by design
Overlaps Across Software Builds

Evolve During Transition [After Sw IOC]

Evolve After Architecture Complete

I. Elab. Construction Transition
ICM Showing Multi-Build Software in a System

- DoD General/DoD Milestones
- Exploration Commitment Review
- Valuation Commitment Review
- Architecture Commitment Review
- Development Commitment Review
- Operations Commitment Review
- Exploration (ECR)
- Valuation (VCR/CD)
- Architecture (ACR/A)
- Development (DCR/B)
- Operations Commitment (OCR/C1, OCR/C2, OCR/C3)

Activities (EVADO):
- Exploration
- Valuation
- Architecting
- Development
- Operations

Concurrent Risk-and-Opportunity-Driven Growth of System Understanding and Definition:
- Initial Scoping
- Concept Definition, Investment Analysis
- Software System Architecting
- Increment Development
- Increment Operations

Evaluation of Evidence of Feasibility to Proceed:
- Feasibility Rationales

Stakeholder Review and Commitment:
- Risk? Acceptable
- Risk? Acceptable
- Risk? Acceptable
- Risk? Acceptable

Adjust Scope, Priorities, or Discontinue
What is a “System of Systems”

- Very large systems developed by creating a framework or architecture to integrate constituent systems.
- SoS constituent systems independently developed and managed
  - New or existing systems in various stages of development/evolution
  - May include a significant number of COTS products
  - Have their own purpose
  - Can dynamically come and go from SoS
- SoS exhibits emergent behavior not otherwise achievable by component systems
- Typical domains
  - Business: Enterprise-wide and cross-enterprise integration to support core business enterprise operations across functional and geographical areas
  - Military: Dynamic communications infrastructure to support operations in a constantly changing, sometimes adversarial, environment

*Based on Mark Maier’s SoS definition [Maier, 1998]*
Types of “System of Systems”

- **Virtual** [Maier, 1998]
  - Lacks a central management authority and a clear SoS purpose
  - Often ad hoc and may use a service-oriented architecture where the constituent systems are not necessarily known

- **Collaborative** [Maier, 1998]
  - Constituent system engineering teams work together more or less voluntarily to fulfill agreed upon central purposes
  - No SoSE team to guide or manage activities of constituent systems

- **Acknowledged** [Dahmann, 2008]
  - Have recognized objectives, a designated manager, and resources at the SoS level (SoSE team)
  - Constituent systems maintain their independent ownership, objectives, funding, and development approaches

- **Directed** [Maier, 2008]
  - SoS centrally managed by a government, corporate, or Lead System Integrator (LSI) and built to fulfill specific purposes
  - Constituent systems maintain ability to operate independently, but evolution subordinated to centrally managed purpose
ICM Showing Multi-Build Software in DSOS
Multi-Build Software Cost Estimation Using COINCOMO

Using the COINCOMO 2.0 tool

- COCOMO model as a base: estimated the software Effort (PM) and Schedule (M) for each module
- COPSEMO model to separate the man power loading across Elaboration and Construction phases
- COPSEMO model to add additional effort and schedule for Inception and Transition phases

Used a spreadsheet to combine efforts AFTER aligning the beginning of Elaboration with the end of Construction

COCOMO calculates effort and schedule for the Elaboration and Construction phases of a build with new code and code carried forward from the previous build treated as re-used code with very favorable re-use parameters.
Incremental Development Productivity Decline (IDPD)

• Overview
  – The “Incremental Productivity Decline” (IDPD) factor represents the percentage of decline in software producibility from one increment to the next.
  – The decline is due to factors such as previous-increment breakage and usage feedback, increased integration and testing effort.
  – Another source of productivity decline is that maintenance of reused previous build software is not based on equivalent lines of software credited during the previous build, but on the full amount of reused software.
    • Build 1: 200 KSLOC new, 200K Reused@20% yields a 240 K ESLOC “count” for estimation models.
    • Build 2: there are 400 KSLOC of Build 1 to maintain and integrate
  – Such phenomena may cause the IDPD factor to be higher for some builds and lower for others.
Incremental Development Productivity Decline (IDPD)

- Example: Site Defense BMD Software
  - 5 builds, 7 years, $100M
  - Build 1 productivity over 300 SLOC/person month
  - Build 5 productivity under 150 SLOC/PM
    - Including Build 1-4 breakage, integration, rework
    - 318% change in requirements across all builds
    - IDPD factor=20% productivity decrease per build
  - Similar trends in later unprecedented systems
  - Not unique to DoD: key source of Windows Vista delays
IDPD Ranges

• Some savings: more experienced personnel (5-20%)
  – Depending on personnel turnover rates
• Some increases: code base growth, diseconomies of scale, requirements volatility, user requests
  – Breakage, maintenance of full code base (20-40%)
  – Diseconomies of scale in development, integration (10-25%)
  – Requirements volatility; user requests (10-25%)
• Best case: 20% more effort (IDPD=6%)
• Worst case: 85% (IDPD=23%)
Effects of IDPD on Number of Increments

- Model relating productivity decline to number of builds needed to reach 8M SLOC Full Operational Capability
- Assumes Build 1 production of 2M SLOC @ 100 SLOC/PM
  - 20000 PM / 24 mo. = 833 developers
  - Constant staff size for all builds
- Analysis varies the productivity decline per build
  - Extremely important to determine the incremental development productivity decline (IDPD) factor per build
Conclusion

- Staffing stability helps to improve team cohesion and developer experience, thus provide positive contribution to productivity outcome

- Design deficiency and code breakage causes productivity declines
  - If the original design is insufficient to accommodate additional modules, and a re-architecting effort was necessary to put this project back on track
  - Inserting new code into the previous build adds effort to read, analyze, and test both the new and old code in order to ensure nothing is broken, this extra effort may be mitigated by experienced staff
Q & A

• Questions?
• Comments?
• Thank you very much