Software Quality in Healthcare Systems

Dr. Bill Curtis
SVP & Chief Scientist, CAST Research Labs
Director, Consortium for IT Software Quality
Co-author, Capability Maturity Model (CMM)

Perennial Software Crisis

- Weak system modeling
- No safety analysis
- Inadequate system testing
- Amateurish code

- Weak system modeling
- Poor performance analysis
- Inadequate system testing
- Amateurish code

Personal Observations

- Modeling is not common industry practice:
  - Only 5% of IT organizations model
  - Few of these models are complete
  - Even fewer are up to date

- The most common modeling tool is...Visio

- Agile methods emphasize code over models

- Healthcare has lagged other industry segments in adopting software engineering best practices

- Medical device software is ahead of medical IT software

Resistance to Modeling

- Past excessive documentation soured the environment
- Few software developers were trained as engineers
- Developers have thin knowledge of application domains
- Many developers are not system thinkers
- Modeling takes time and demand is immediate
- Emergent design is the new mantra, code is truth
- Management ignores the cost of quality
MBSE — A Centuries Old Practice

“I am an artist, I write beautiful code”

“I am an artist, I model the system before writing code”

Model-Based Sculpture Engineering

Model for ‘Daniel in the Lion’s Den’
Bernini’s ‘Daniel in the Lion’s Den’
Model for ‘Fountain of the Moor’
Bernini’s ‘Fountain of the Moor’

Code Unit Level — Individual Developers

- Code style & layout
- Expression complexity
- Code documentation
- Class or program design
- Basic coding standards
- Developer level
Technology Level — Projects and Teams

1. Code Unit Level
   - Code style & layout
   - Expression complexity
   - Code documentation
   - Class or program design
   - Basic coding standards
   - Developer level

2. Technology Level
   - Single language/technology layer
   - Intra-technology architecture
   - Intra-layer dependencies
   - Design & structure
   - Inter-program invocation
   - Security vulnerabilities
   - Development team level

System Level — Multiple Organizations

1. Code Unit Level
   - Code style & layout
   - Expression complexity
   - Code documentation
   - Class or program design
   - Basic coding standards
   - Developer level

2. Technology Level
   - Single language/technology layer
   - Intra-technology architecture
   - Intra-layer dependencies
   - Design & structure
   - Inter-program invocation
   - Security vulnerabilities
   - Development team level

3. Application Stack Level
   - Function point
   - Effort estimation
   - Data access control
   - SDK versioning
   - Calibration across technologies
   - IT organization level
The Fourth Wave in Software Engineering

1. **Languages**
   - What: 3rd & 4th generation languages, structured programming
   - When: 1965-1980
   - Why: Give developers greater power for expressing programs

2. **Methods**
   - What: Design methods, CASE tools
   - When: 1980-1990
   - Why: Give developers better aids for constructing systems

3. **Process**
   - What: CMM, ITIL, PMBOK, Agile
   - When: 1990-2005
   - Why: Improve software management and discipline

4. **Product**
   - What: Architecture, MBSE, Structural quality, Reuse
   - When: 2005
   - Why: Improve the engineering of software products
**Capability Maturity Model (CMM/CMMI)**

- **Innovate**
  - Level 5: Optimizing
  - Level 4: Quantitatively Manage
  - Level 3: Defined
  - Level 2: Managed
  - Level 1: Initial

- **Optimize**
  - Innovation management
  - Capable management
  - Standardized practices

- **Standardize**
  - Repeatability practices

- **Stabilize**

---

**Raytheon's Cost of Quality**

- **Performance** — cost of building it right first time
- **Nonconformance** — cost of rework
- **Appraisal** — cost of testing
- **Prevention** — cost of preventing nonconformance

<table>
<thead>
<tr>
<th>Year</th>
<th>CMM Level</th>
<th>Perform</th>
<th>Non-conform</th>
<th>Appraisal</th>
<th>Prevent</th>
</tr>
</thead>
<tbody>
<tr>
<td>1994</td>
<td>4</td>
<td>76%</td>
<td>6%</td>
<td>17%</td>
<td></td>
</tr>
<tr>
<td>1992</td>
<td>3</td>
<td>66%</td>
<td>11%</td>
<td>23%</td>
<td></td>
</tr>
<tr>
<td>1990</td>
<td>2</td>
<td>55%</td>
<td>18%</td>
<td>15%</td>
<td>12%</td>
</tr>
<tr>
<td>1988</td>
<td>1</td>
<td>34%</td>
<td>41%</td>
<td>15%</td>
<td>7%</td>
</tr>
</tbody>
</table>

Enterprise Value Chain — Lean Modeling

Medical IT systems provider — $XX,000,000 loss per year in settlements

- Sales
  - Bill of sale
- Legal
  - Contract
- Provisioning
  - Specifications
- Finance
  - Invoice

Items left off bill of sale — Incomplete information — Unspecified Requirements — Incorrect invoices

Lean analysis:
- Cascading rework
- Inefficient practices
- Overwork, bottlenecks

Modeling Healthcare Process Maturity

<table>
<thead>
<tr>
<th>State of the organization</th>
<th>Implications</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Level 5 Innovating</strong></td>
<td>Make proactive improvements to close gaps between process capability and business targets</td>
</tr>
<tr>
<td></td>
<td>Simplified requirements and systems</td>
</tr>
<tr>
<td><strong>Level 4 Optimized</strong></td>
<td>Optimize process performance statistically to reduce variation, predict outcomes, eliminate waste</td>
</tr>
<tr>
<td></td>
<td>High performance, synchronized systems</td>
</tr>
<tr>
<td><strong>Level 3 Standardized</strong></td>
<td>Establish standard business processes, measures, and training to achieve an economy of scale</td>
</tr>
<tr>
<td></td>
<td>Common standardized requirements</td>
</tr>
<tr>
<td><strong>Level 2 Stabilized</strong></td>
<td>Stabilize local work, establish repeatable practices, manage the capability to meet commitments</td>
</tr>
<tr>
<td></td>
<td>Multiple variations of requirements, complex systems</td>
</tr>
<tr>
<td><strong>Level 1 Initial</strong></td>
<td>Inconsistent processes and management that rely on individual motivation and often heroic effort</td>
</tr>
<tr>
<td></td>
<td>Messy, incomplete, and conflicting requirements</td>
</tr>
</tbody>
</table>
ISO 25010 — Software Product Quality


Reliability: Confidentiality, Integrity, Non-repudiation, Accountability, Authority, Compliance.

Operability: Co-existence, Interoperability, Compliance.

Maintainability: Modularity, Reusability, Analyzability, Changeability, Modifiability, Modularity, Stability, Testability, Compliance.


Testing is Not Enough

“As higher levels of assurance are demanded … testing cannot deliver the level of confidence required at a reasonable cost.”

“The correctness of the code is rarely the weakest link.”

“…a failure to satisfy a non-functional requirement can be critical, even catastrophic…non-functional requirements are sometimes difficult to verify. We cannot write a test case to verify a system’s reliability…The ability to associate code to non-functional properties can be a powerful weapon in a software engineer’s arsenal.”

**Architecturally Complex Defects**

A structural flaw involving interactions among multiple components that reside in different application layers.

- **Component-level violations**
  - % of total app defects: 92%
  - % of total repair effort: 48%

- **Architecturally Complex Defects**
  - 8%

80% of architecturally complex defects touch an **Architectural Hotspot**—a badly designed component causing problems.

Architectural hotspots provide a roadmap for remediating the worst risk, rework, and cost drivers.

---

**Agile/Waterfall Mix Best**

<table>
<thead>
<tr>
<th></th>
<th>Robustness</th>
<th>Performance</th>
<th>Security</th>
</tr>
</thead>
<tbody>
<tr>
<td>Agile Mix None Other Water</td>
<td><img src="image" alt="Boxplot" /></td>
<td><img src="image" alt="Boxplot" /></td>
<td><img src="image" alt="Boxplot" /></td>
</tr>
</tbody>
</table>

**Changeability**

<table>
<thead>
<tr>
<th></th>
<th>Agile Mix None Other Water</th>
</tr>
</thead>
</table>

**Transferability**

<table>
<thead>
<tr>
<th></th>
<th>Agile Mix None Other Water</th>
</tr>
</thead>
</table>

**Agile Mix None Other Water**

- Agile: $n = 57$
- Mix (Agile, Waterfall): $n = 46$
- No method: $n = 21$
- Other: $n = 36$
- Waterfall: $n = 60$

All F-tests significant: $df = 4, 215$; $p < .02$