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)

MBSE in Healthcare Summit
Boston, MA
June 18, 2014

Perennial Software Crisis

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


An Investigation of the Therac-25 Accidents

Late 1980s

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

Leveson, N.G. & Turner, C.S. (1993). IEEE Computer, 26 (7), 18-41.

CAST

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

CULTURE OF CODE

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 for 'Daniel in the Lion's Den'

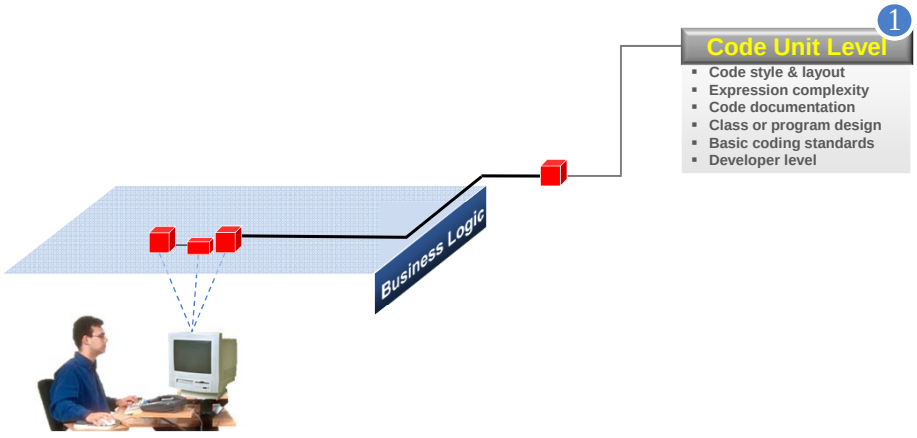
Bernini's 'Daniel in the Lion's Den'

Model for 'Fountain of the Moor'

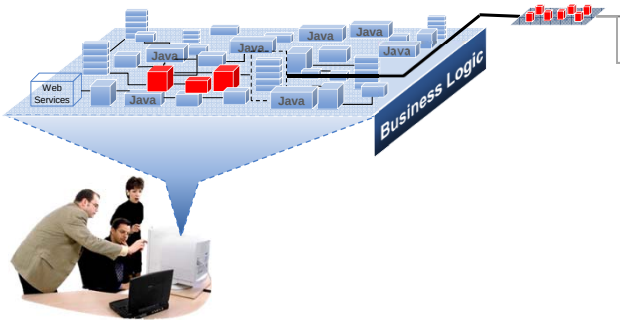
Bernini's Fountain of the Moor'

Model-Based Sculpture Engineering

Code Unit Level — Individual Developers



Technology Level — Projects and Teams



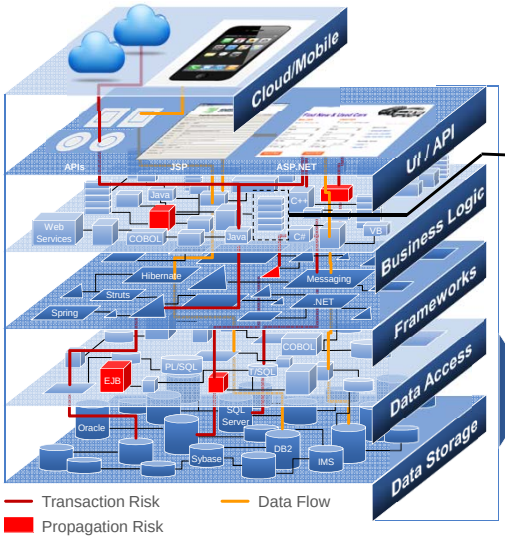
1 Code Unit Level

- 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

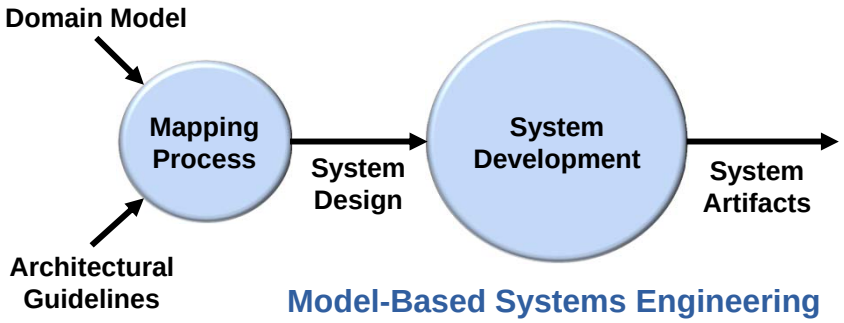
2 Technology Level

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

3 Application Stack Level

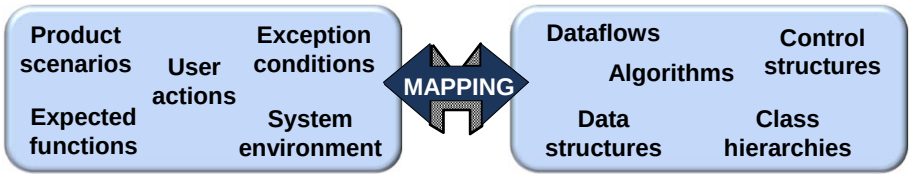
- Integration quality
- Architectural compliance
- Risk propagation
- Application security
- Resiliency checks
- Transaction integrity
- Function point,
- Effort estimation
- Data access control
- SDK versioning
- Calibration across technologies
- IT organization level

MBSE and System Gurus



Model-Based Systems Engineering

Guru system architects

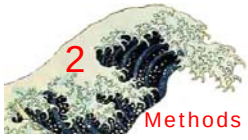


Domain knowledge **Computational knowledge**

The Fourth Wave in Software Engineering

What: Architecture, MBSE, Structural quality, Reuse
When: 2005➔
Why: Improve the engineering of software products

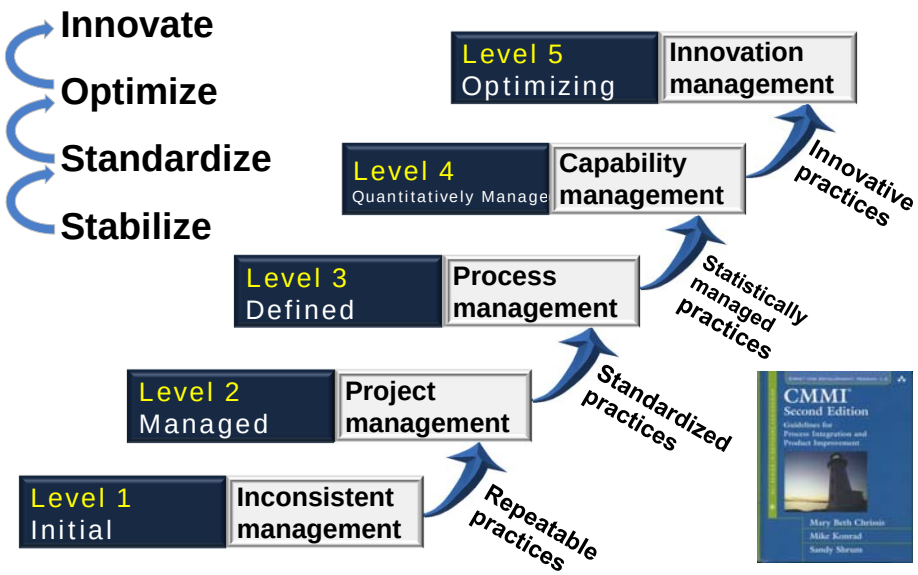
What: CMM, ITIL, PMBOK, Agile
When: 1990-2005
Why: Improve software management and discipline



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

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

Capability Maturity Model (CMM/CMMI)



Paulk, M.C, Weber, C.V., Curtis, B., & Chrissis, M.B. (1995). Capability Maturity Model. Addison-Wesley.

Raytheon's Cost of Quality

Performance — ***cost of building it right first time***

Nonconformance — ***cost of rework***

Appraisal — ***cost of testing***

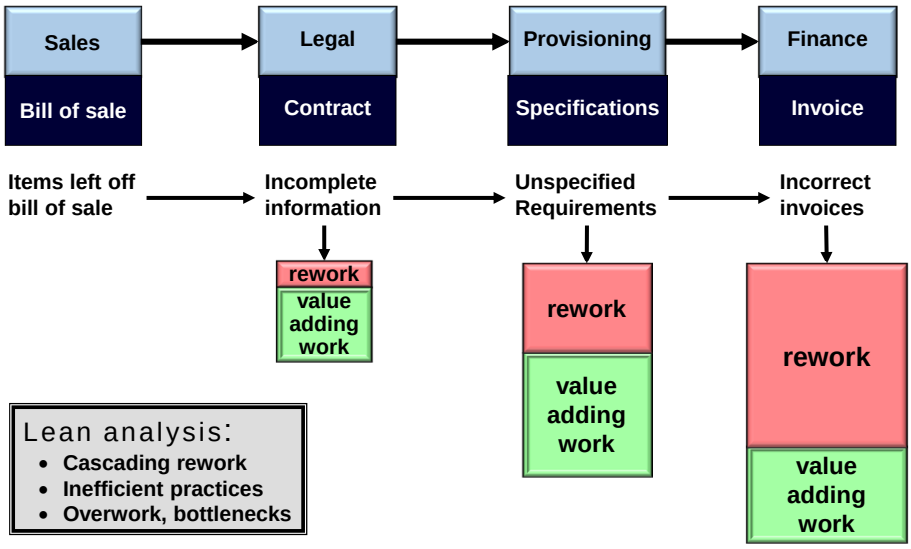
Prevention — ***cost of preventing nonconformance***

Year	CMM Level	Perform	Non-conform	Appraisal	Prevent
1994	4	76%	6%	17%	
1992	3	66%	11%	23%	
1990	2	55%	18%	15%	12%
1988	1	34%	41%	15%	7%

Dion, IEEE Software, 1993; Haley, SEI TR, 1995.

Enterprise Value Chain — Lean Modeling

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

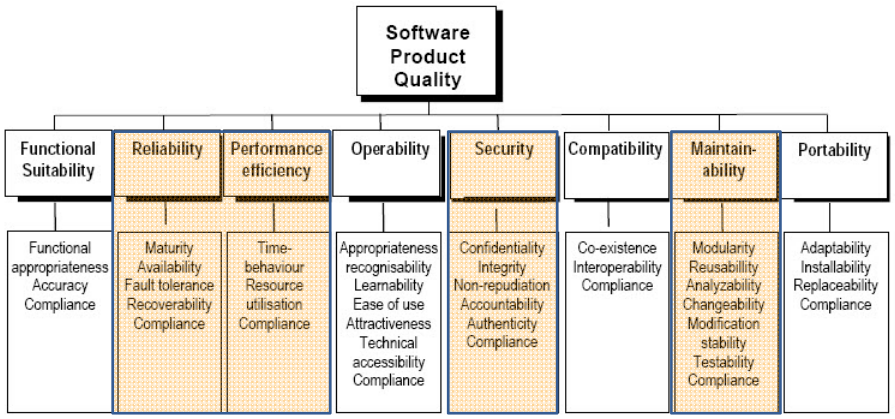


Lean analysis:
- Cascading rework
- Inefficient practices
- Overwork, bottlenecks

Modeling Healthcare Process Maturity

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

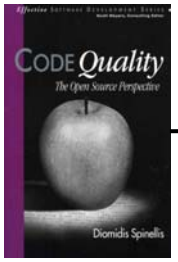
ISO 25010 — Software Product Quality



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."

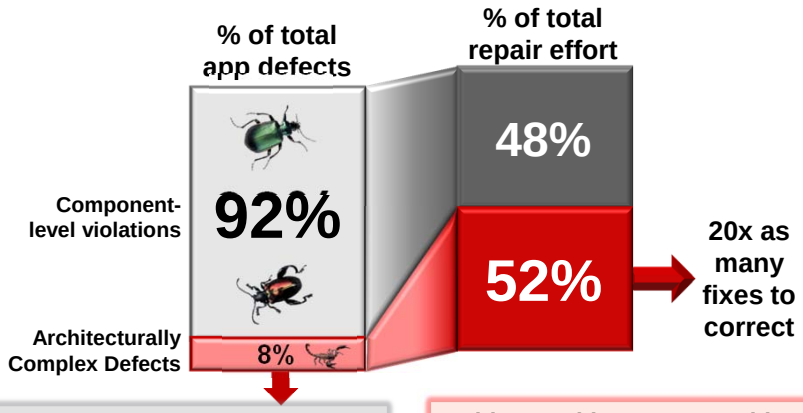


"…a failure to satisfy a non-functional requirement can be critical, even catastrophic…non-functional requirements are sometimes difficult to verify. We cannot write a test case to verify a system's reliability…The ability to associate code to non-functional properties can be a powerful weapon in a software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

Jackson, D. (2009). Communications of the ACM, 52 (4); Spinellis, D. (2006). Code Quality. Addison-Wesley.

Architecturally Complex Defects

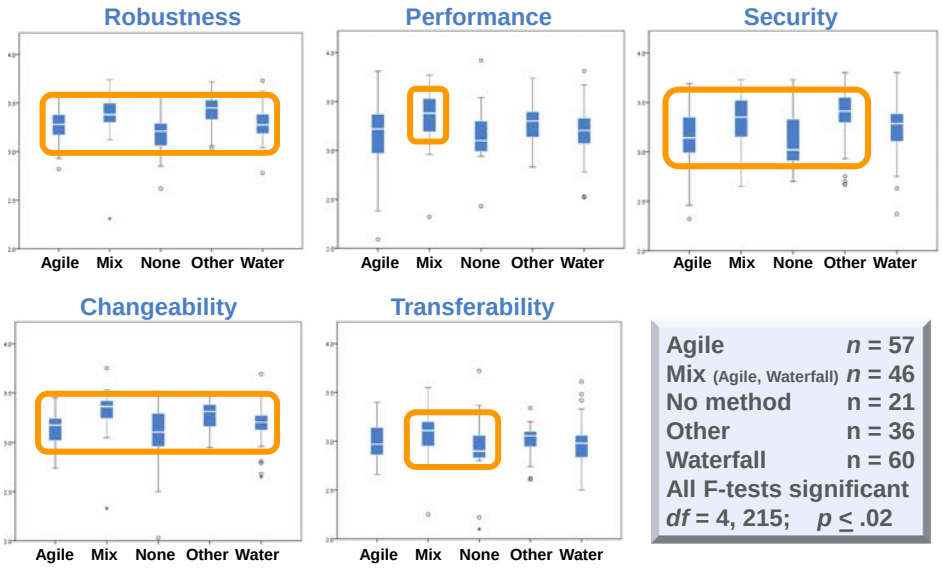
Architecturally Complex Defect — A structural flaw involving interactions among multiple components that reside in different application layers



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



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; $p \leq .02$

Consortium for IT Software Quality