

Trends and New Directions in Software Architecture

Software Engineering Institute
Carnegie Mellon University
Pittsburgh, PA 15213

Linda Northrop
Chief Scientist, Software Solutions Division
SEI Fellow



Software Engineering Institute

Carnegie Mellon University

© 2015 Carnegie Mellon University

Copyright 2015 Carnegie Mellon University

This material is based upon work funded and supported by the Department of Defense under Contract No. FA8721-05-C-0003 with Carnegie Mellon University for the operation of the Software Engineering Institute, a federally funded research and development center.

Any opinions, findings and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the United States Department of Defense.

NO WARRANTY. THIS CARNEGIE MELLON UNIVERSITY AND SOFTWARE ENGINEERING INSTITUTE MATERIAL IS FURNISHED ON AN "AS-IS" BASIS. CARNEGIE MELLON UNIVERSITY MAKES NO WARRANTIES OF ANY KIND, EITHER EXPRESSED OR IMPLIED, AS TO ANY MATTER INCLUDING, BUT NOT LIMITED TO, WARRANTY OF FITNESS FOR PURPOSE OR MERCHANTABILITY, EXCLUSIVITY, OR RESULTS OBTAINED FROM USE OF THE MATERIAL. CARNEGIE MELLON UNIVERSITY DOES NOT MAKE ANY WARRANTY OF ANY KIND WITH RESPECT TO FREEDOM FROM PATENT, TRADEMARK, OR COPYRIGHT INFRINGEMENT.

This material has been approved for public release and unlimited distribution.

This material may be reproduced in its entirety, without modification, and freely distributed in written or electronic form without requesting formal permission. Permission is required for any other use. Requests for permission should be directed to the Software Engineering Institute at permission@sei.cmu.edu.

ATAM® is registered in the U.S. Patent and Trademark Office by Carnegie Mellon University.

Team Software ProcessSM and TSPSM are service marks of Carnegie Mellon University.

DM-0001699



Software Engineering Institute

Carnegie Mellon University

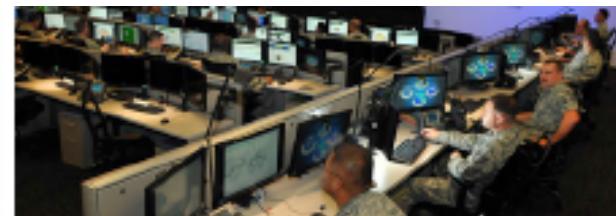
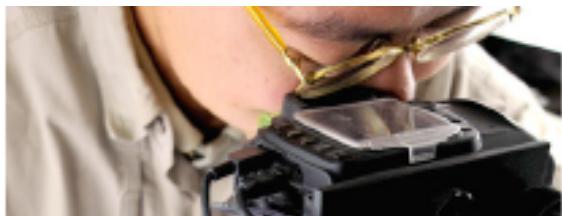
Linda Northrop
January 21, 2015

© 2015 Carnegie Mellon University

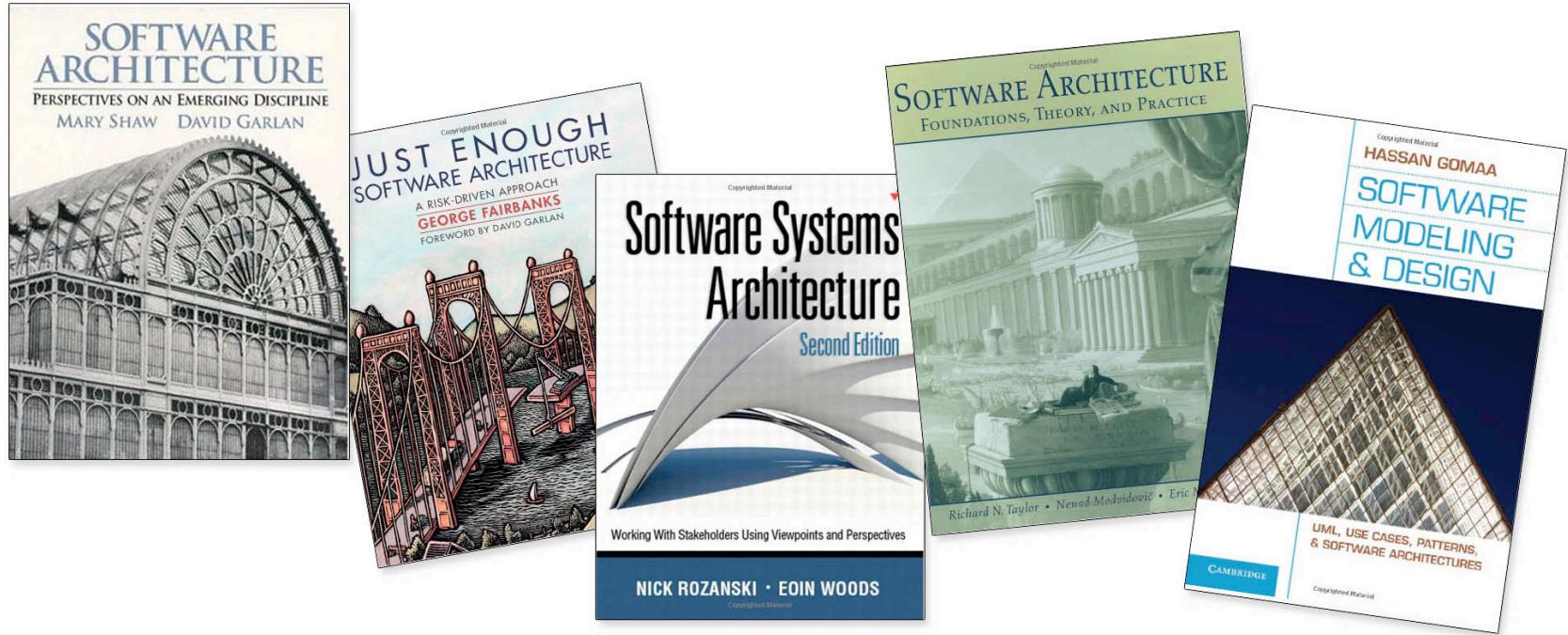
Software Architecture

- The quality and longevity of a software-reliant system is largely determined by its architecture.
- Recent US studies identify architectural issues as a systemic cause of software problems in government systems (OSD, NASA, NDIA, National Research Council).

Architecture is of enduring importance because it is the right abstraction for performing ongoing analyses throughout a system's lifetime.



Software Architecture Thinking



- High-level system design providing system-level structural abstractions and quality attributes, which help in managing complexity
- Makes engineering tradeoffs explicit



Quality Attributes

Quality attributes

- properties of work products or goods by which stakeholders judge their quality
- stem from business and mission goals.
- need to be characterized in a system-specific way

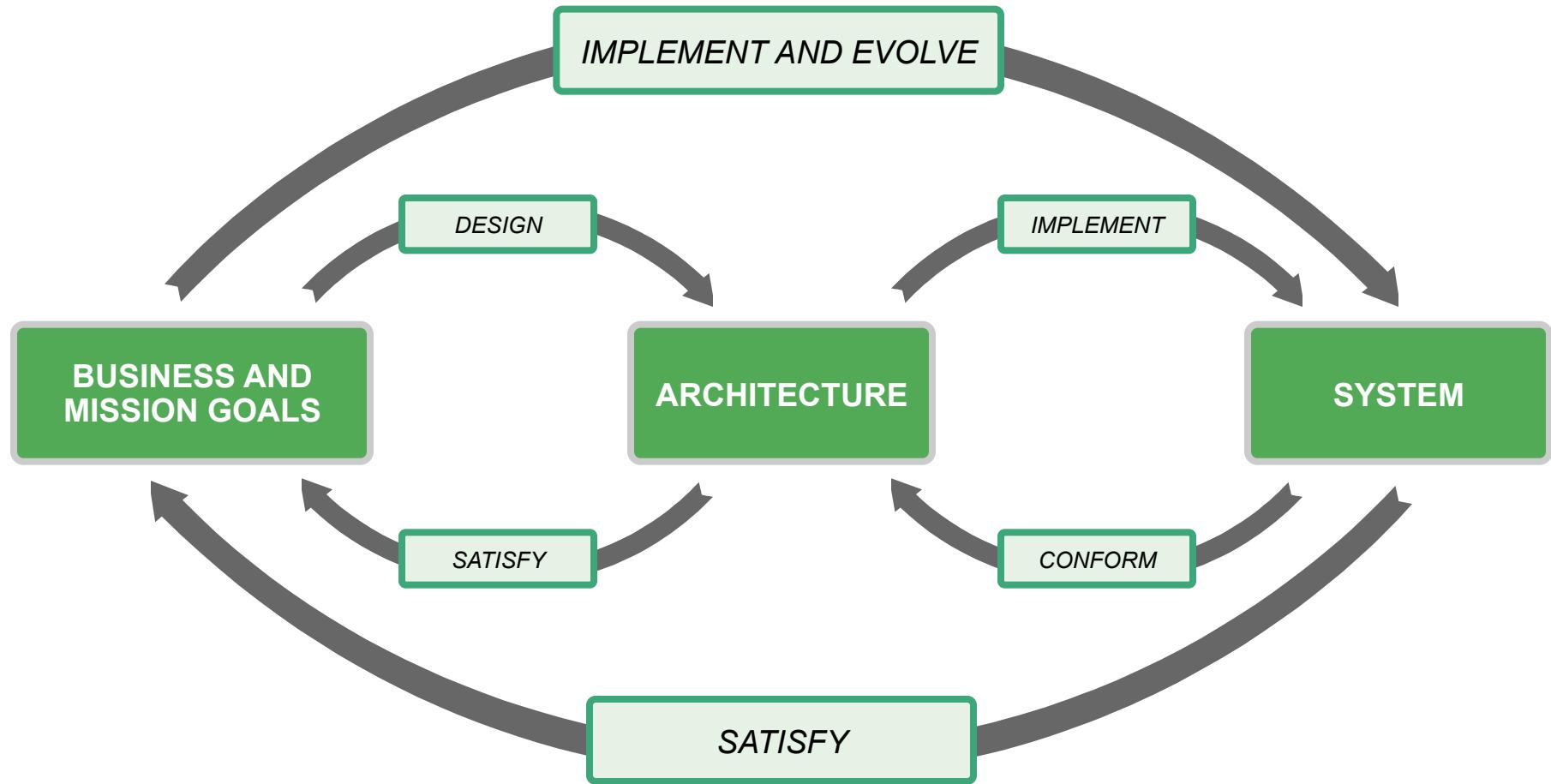


Quality attributes include

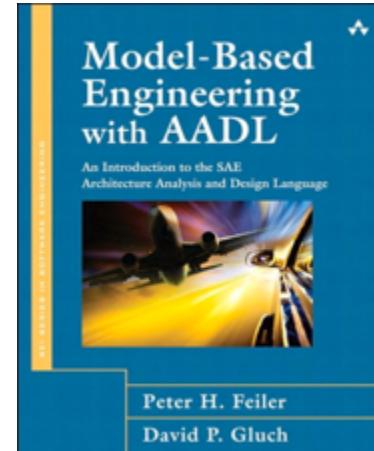
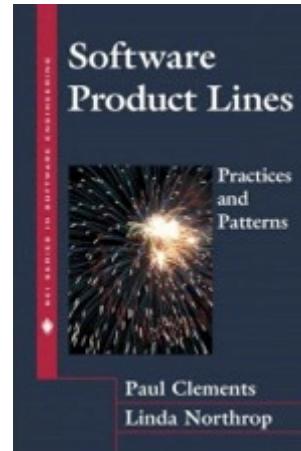
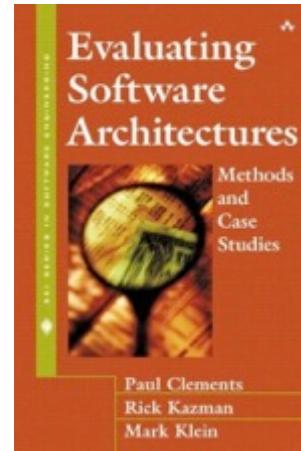
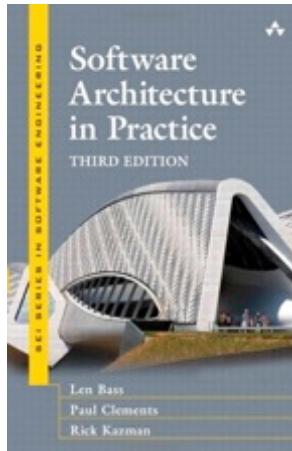
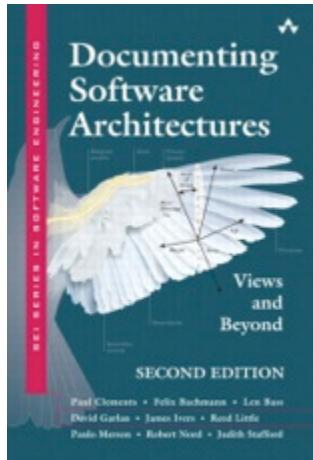
- Performance
- Availability
- Interoperability
- Modifiability
- Usability
- Security
- Etc.



Central Role of Architecture



Our View: Architecture-Centric Engineering



- *explicitly focus on quality attributes*
- *directly link to business and mission goals*
- *explicitly involve system stakeholders*
- *be grounded in state-of-the-art quality attribute models and reasoning frameworks*



Advancements Over the Years

- Architectural patterns
- Component-based approaches
- Company specific product lines
- Model-based approaches
- Frameworks and platforms
- Standard interfaces

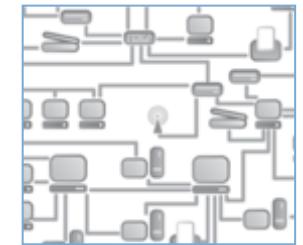
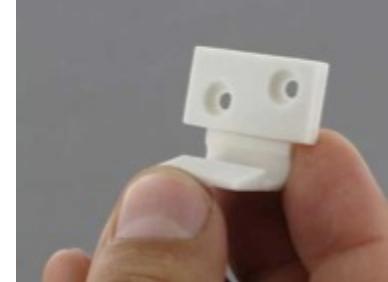
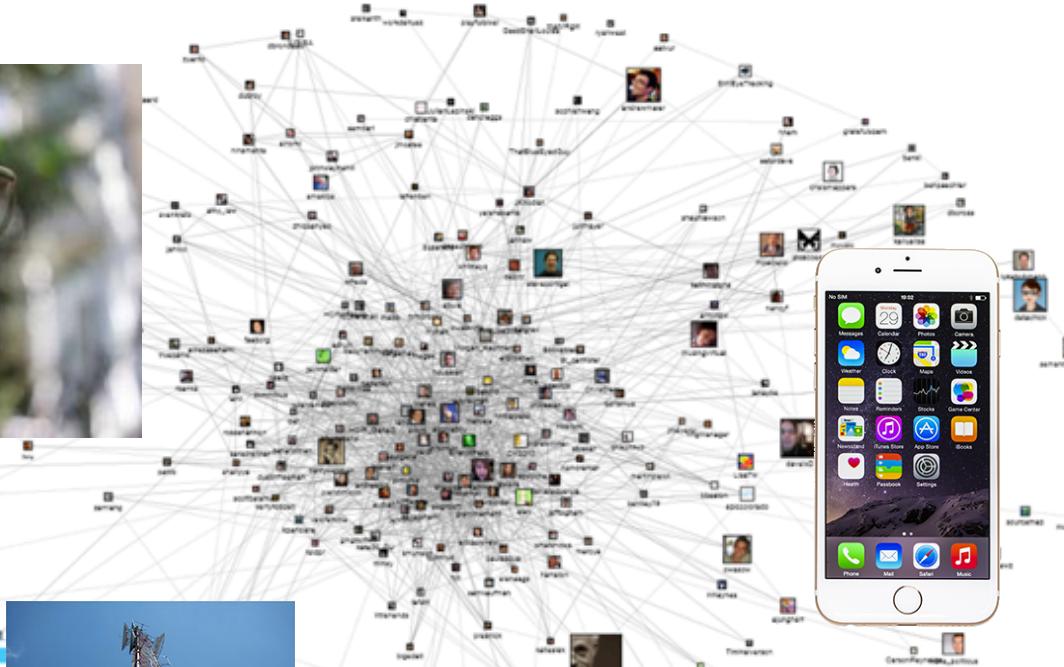


What HAS Changed?

- Increased connectivity
- Scale and complexity
 - decentralization and distribution
 - “big data”
 - increased operational tempo
 - inter-reliant ecosystems
 - vulnerability
 - collective action
- Disruptive and emerging technologies

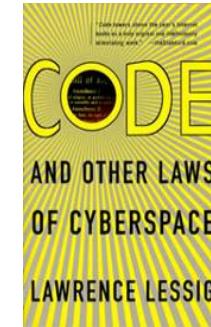


Technology Trends

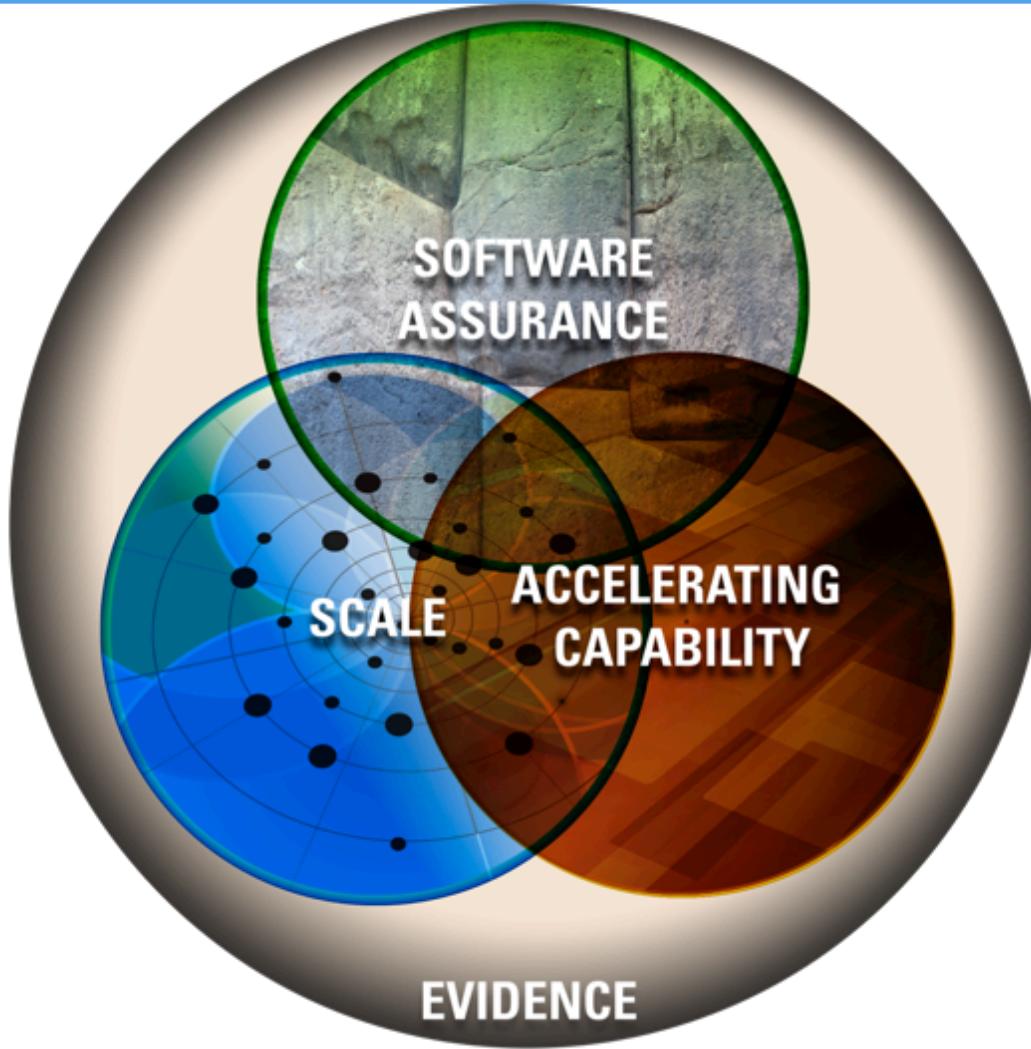


Software Development Trends

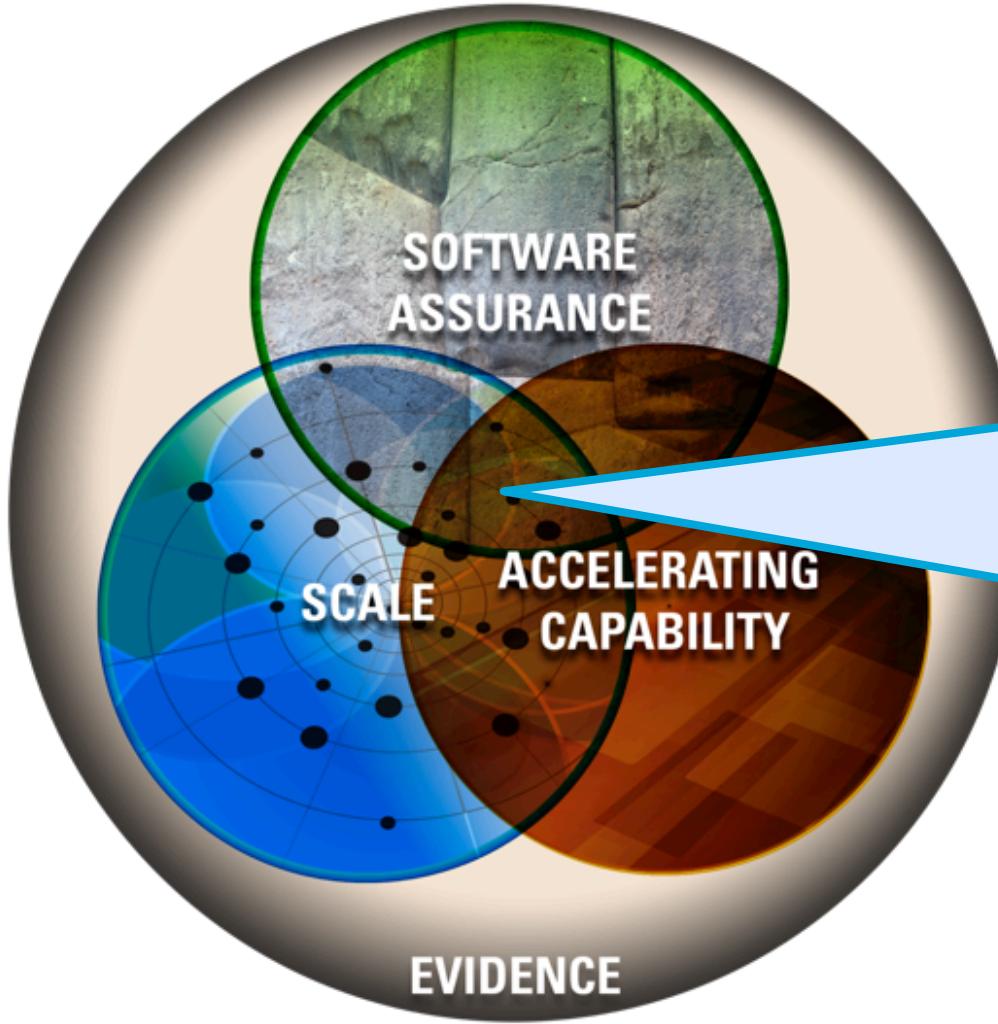
- Application frameworks
- Open source
- Cloud strategies
- NoSQL
- Machine Learning
- MDD
- Incremental approaches
- Dashboards
- Distributed development environments
- DevOps



Technical Challenges



The Intersection and Architecture



At the intersections there are difficult tradeoffs to be made in structure, process, time, and cost.

Architecture is the enabler for tradeoff analyses.



Architecture and Accelerated Capability

How much architecture design is enough?

Can architecture design be done incrementally?

There is a difference between being agile and doing agile.

Agility is enabled by architecture – not stifled by it.

Managing technical debt is key.



Managing Technical Debt*

A design or construction approach that's expedient in the short term but that creates a technical context that increases complexity and cost in the long term.

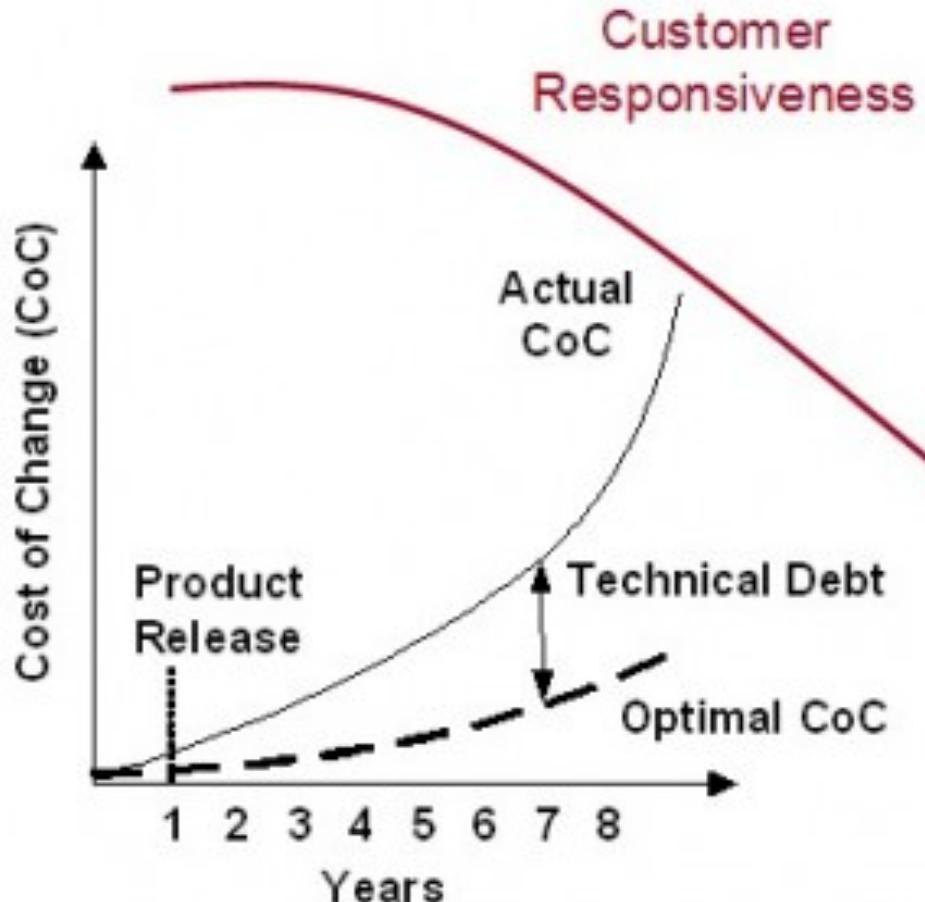
Some examples include:

- continuing to build on a foundation of poor quality legacy code
- prototype that turns into production code
- increasing use of "bad patches," which increases number of related systems that must be changed in parallel

* Term first used by Cunningham, W. 1992. *The WyCash Portfolio Management System*. OOPSLA '92 Experience Report. <http://c2.com/doc/oopsla92.html>.



Technical Debt Impact



From:
Jim Highsmith
2010



Technical Debt Landscape

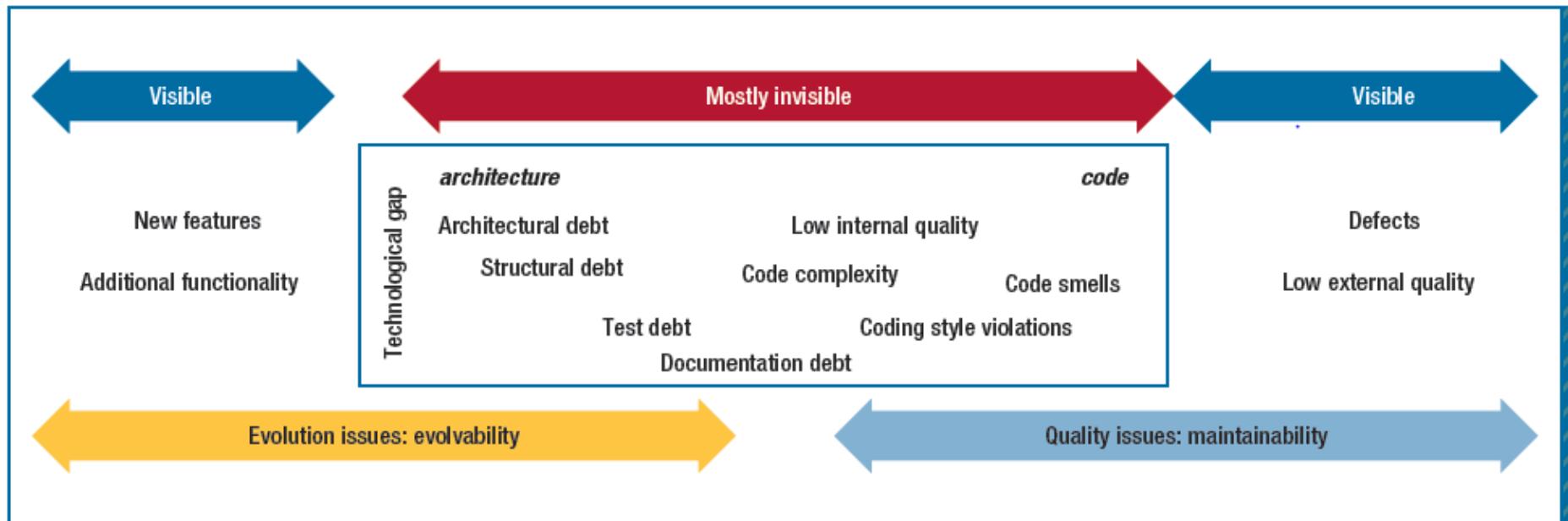


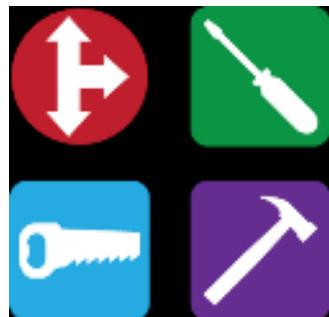
FIGURE 1. The technical debt landscape. On the left, evolution or its challenges; on the right, quality issues, both internal and external.

“invisible results of past decisions about software that negatively affect its future...deferred investment opportunities or poorly managed risks”

Kruchten, P. Nord, R.L., Ozkaya, I. 2012. Technical Debt: From Metaphor to Theory and Practice, IEEE Software, 29(6), Nov/Dec 2012.



Making Hard Choices About Technical Debt



In the quest to become market leader, players race to release a quality product to the marketplace.

The Hard Choices game is a simulation of the software development cycle meant to communicate the concepts of uncertainty, risk, options, and technical debt.



Hard Choices Strategy Game to Communicate Value of Architecture Thinking
game downloadable from <http://www.sei.cmu.edu/architecture/tools/hardchoices/>.

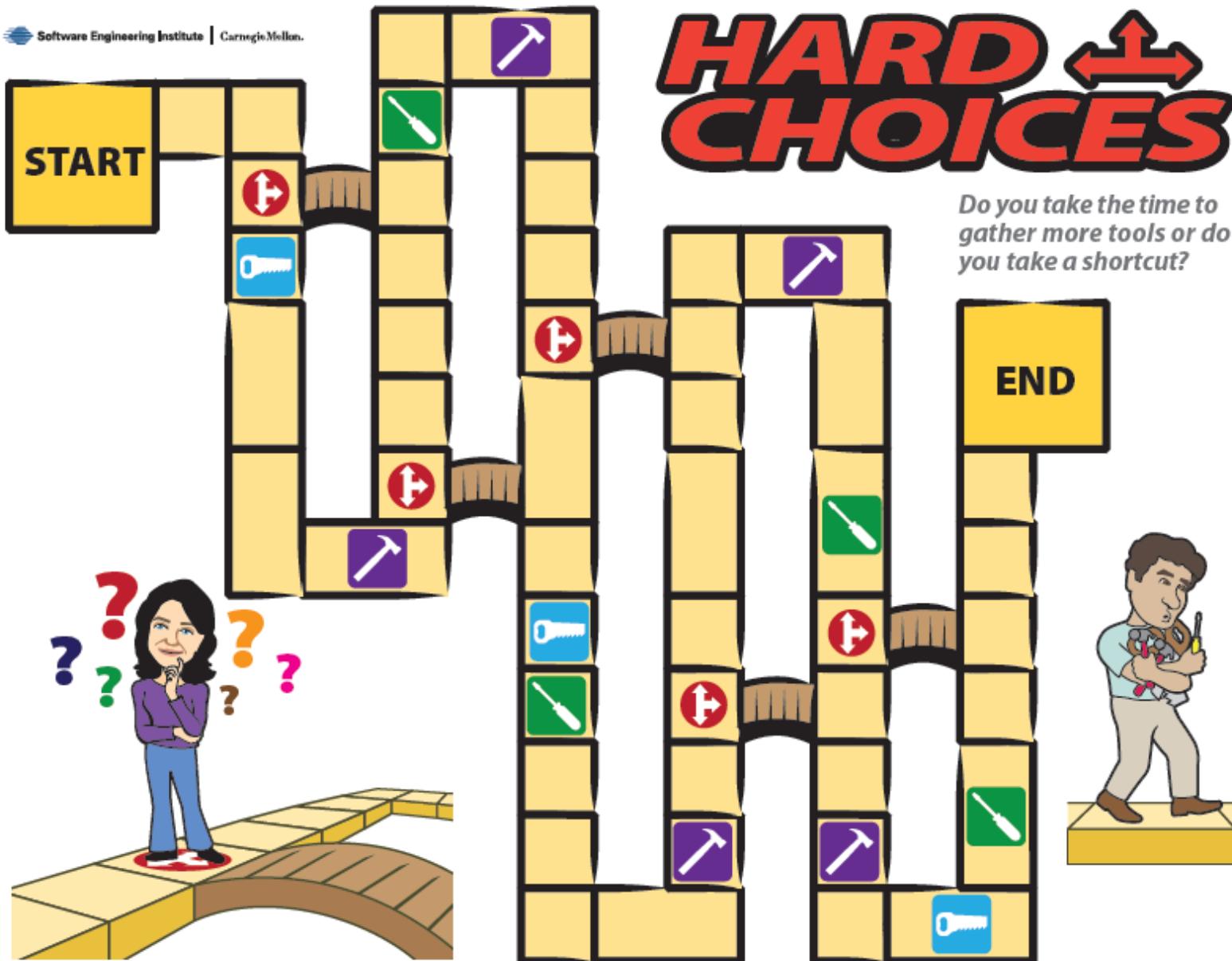


Software Engineering Institute

Carnegie Mellon University

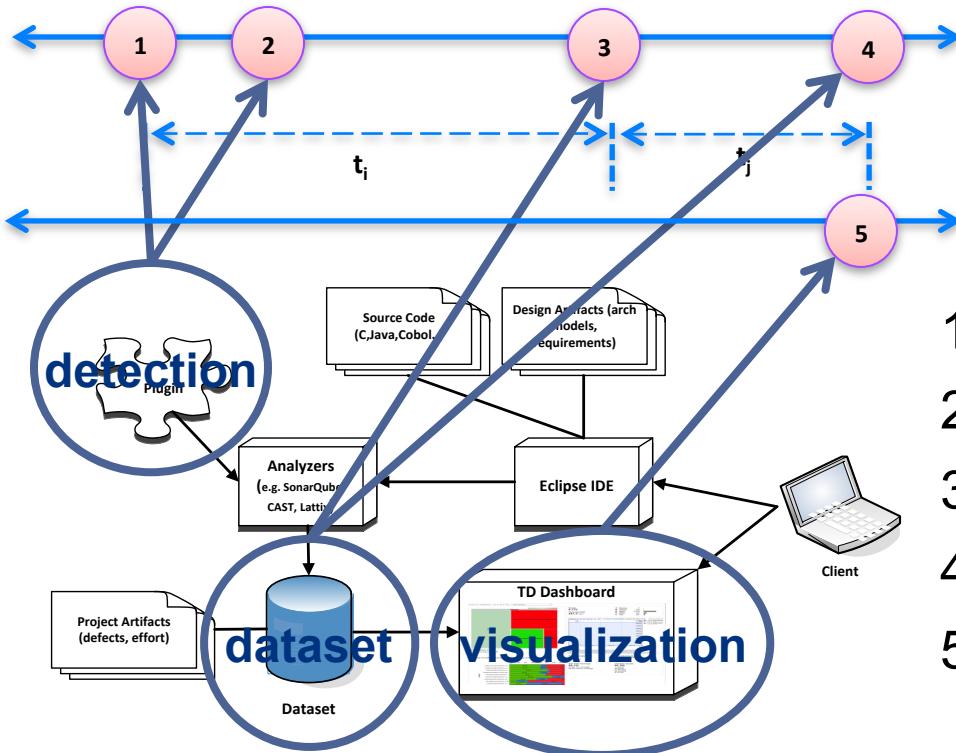
Linda Northrop
January 21, 2015

© 2015 Carnegie Mellon University



Our Current Research

What code and design indicators that correlate well with project measures allow us to manage technical debt?



1. time technical debt is incurred
2. time technical debt is recognized
3. time to plan and re-architect
4. time until debt is actually paid-off
5. continuous monitoring

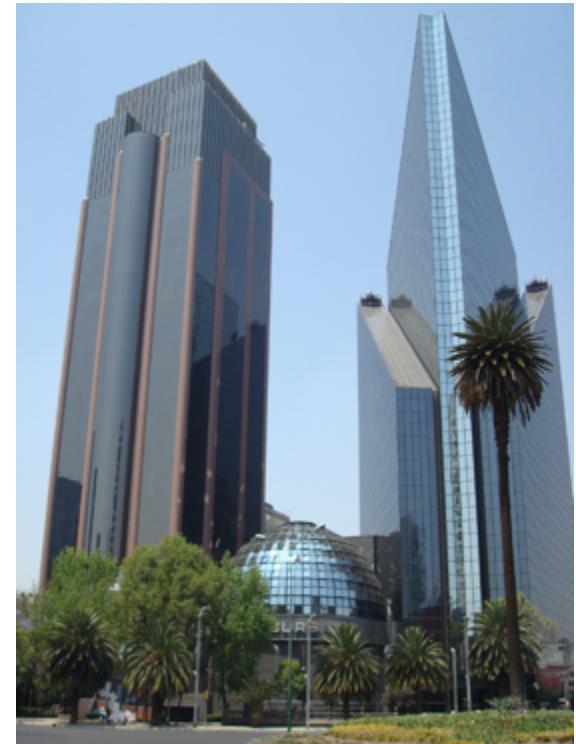


Architecture Done Incrementally

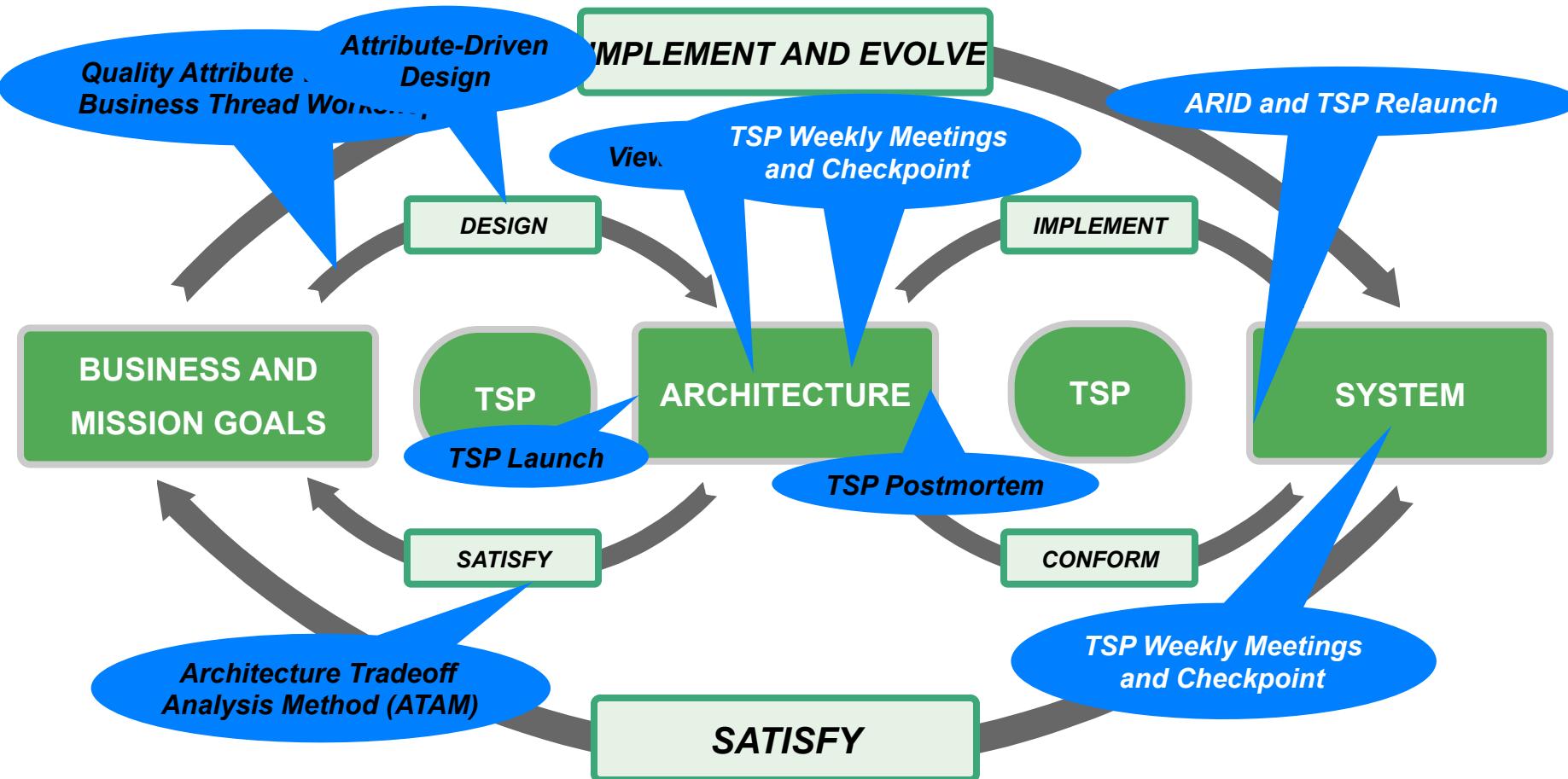
- Bolsa Mexicana de Valores (BMV) operates the Mexican Financial Markets on behalf of the Mexican government.
- Bursatec is the technology arm of the BMV.
- BMV desired a new stock trading engine to drive the market.
- BMV performed a build vs. buy analysis and determined that Bursatec would replace their three existing trading engines with one in-house developed system.

Bursatec committed to deliver a trading engine in 8-10 quarters.

- High performing
- Reliable and of high quality
- Scalable



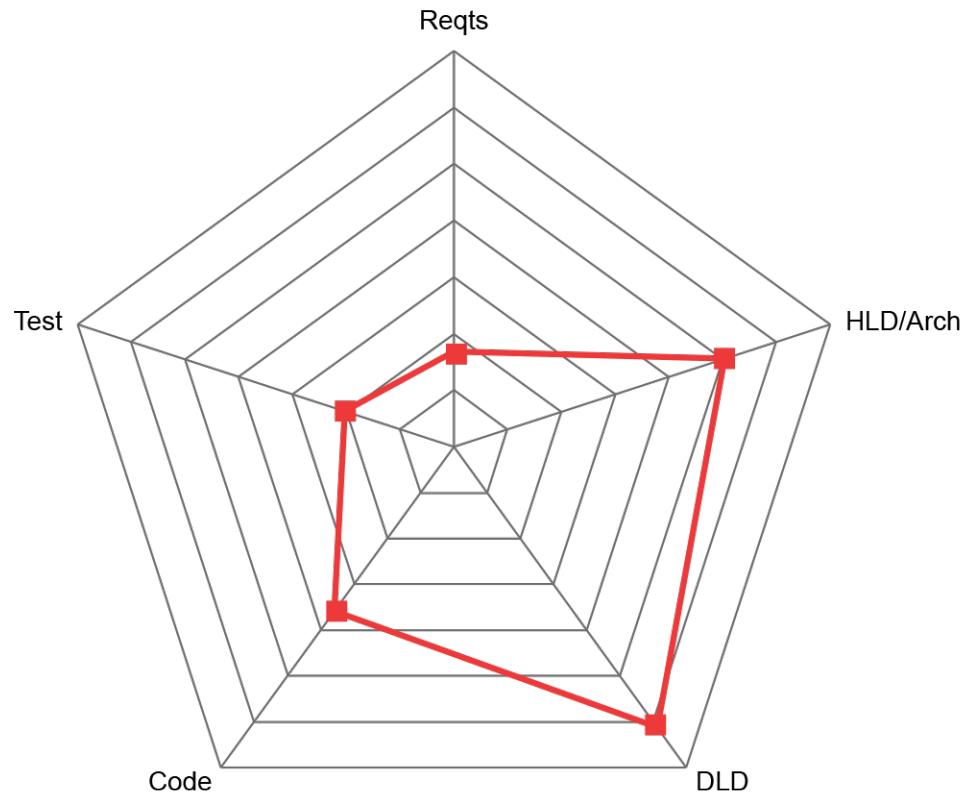
Approach



Team Software Process (TSP) and Architecture-Centric Engineering

Effort in Percent over Cycles – 1

Cycle 1 – 14 Weeks

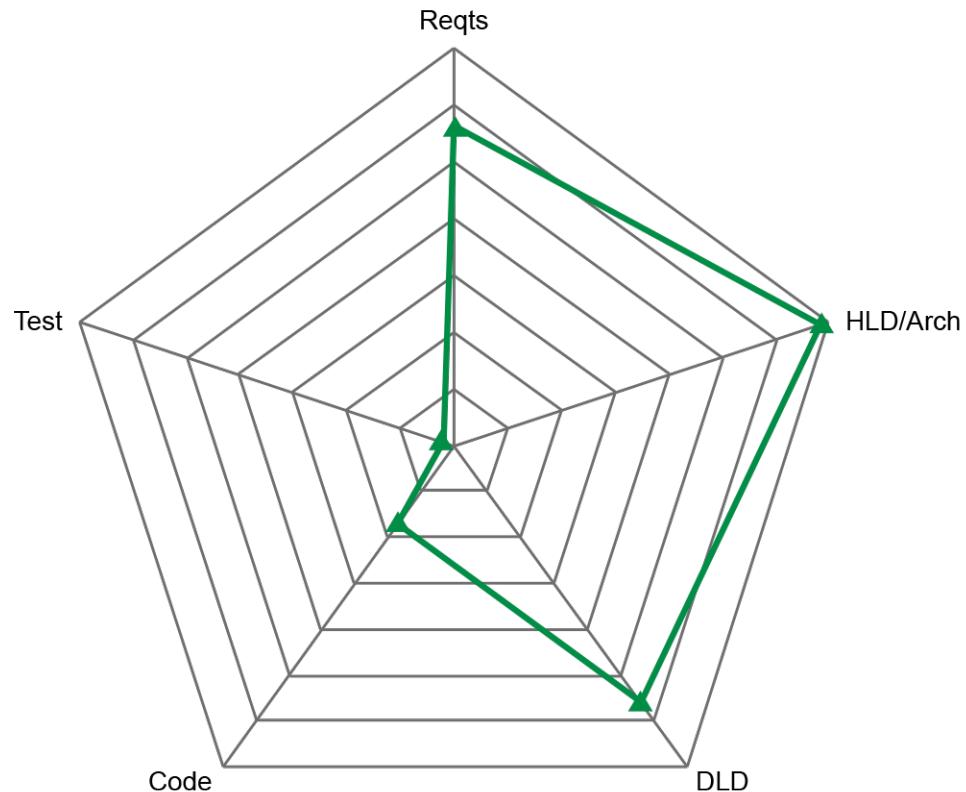


Reqs: Requirements
HLD/Arch: High level Design / Architecture
DLD: Detailed Design (UML)
Code: Coding (no detailed design)
Test: Testing



Effort in Percent over Cycles – 2

Cycle 2 – 10 Weeks

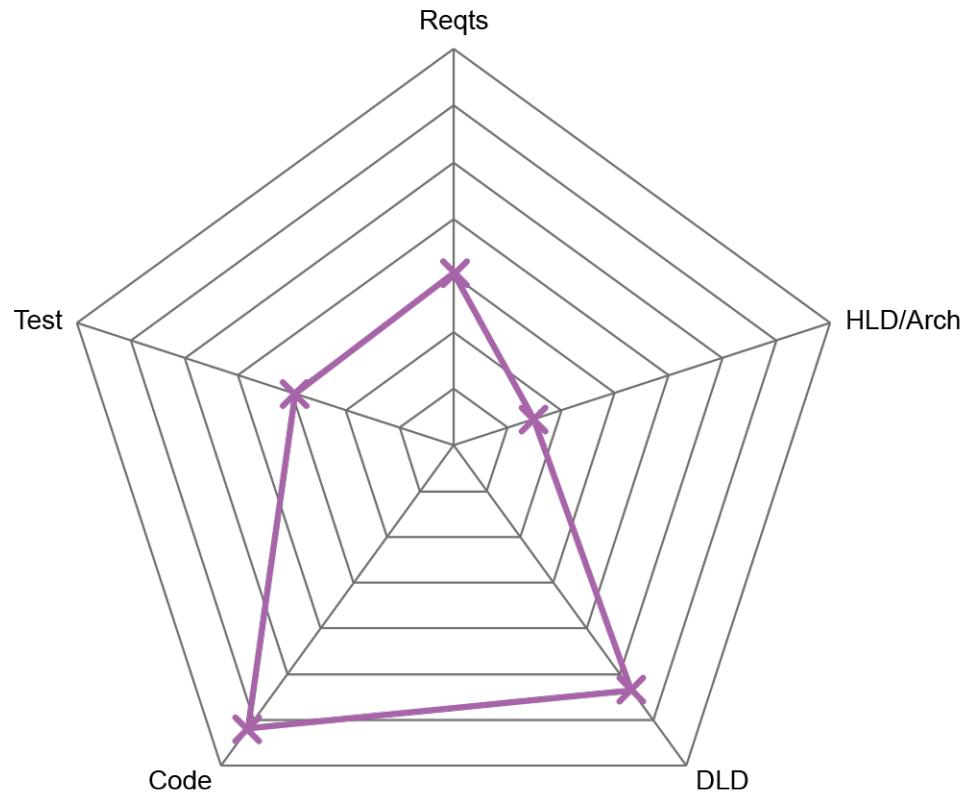


Reqts: Requirements
HLD/Arch: High level Design / Architecture
DLD: Detailed Design (UML)
Code: Coding (no detailed design)
Test: Testing



Effort in Percent over Cycles – 3

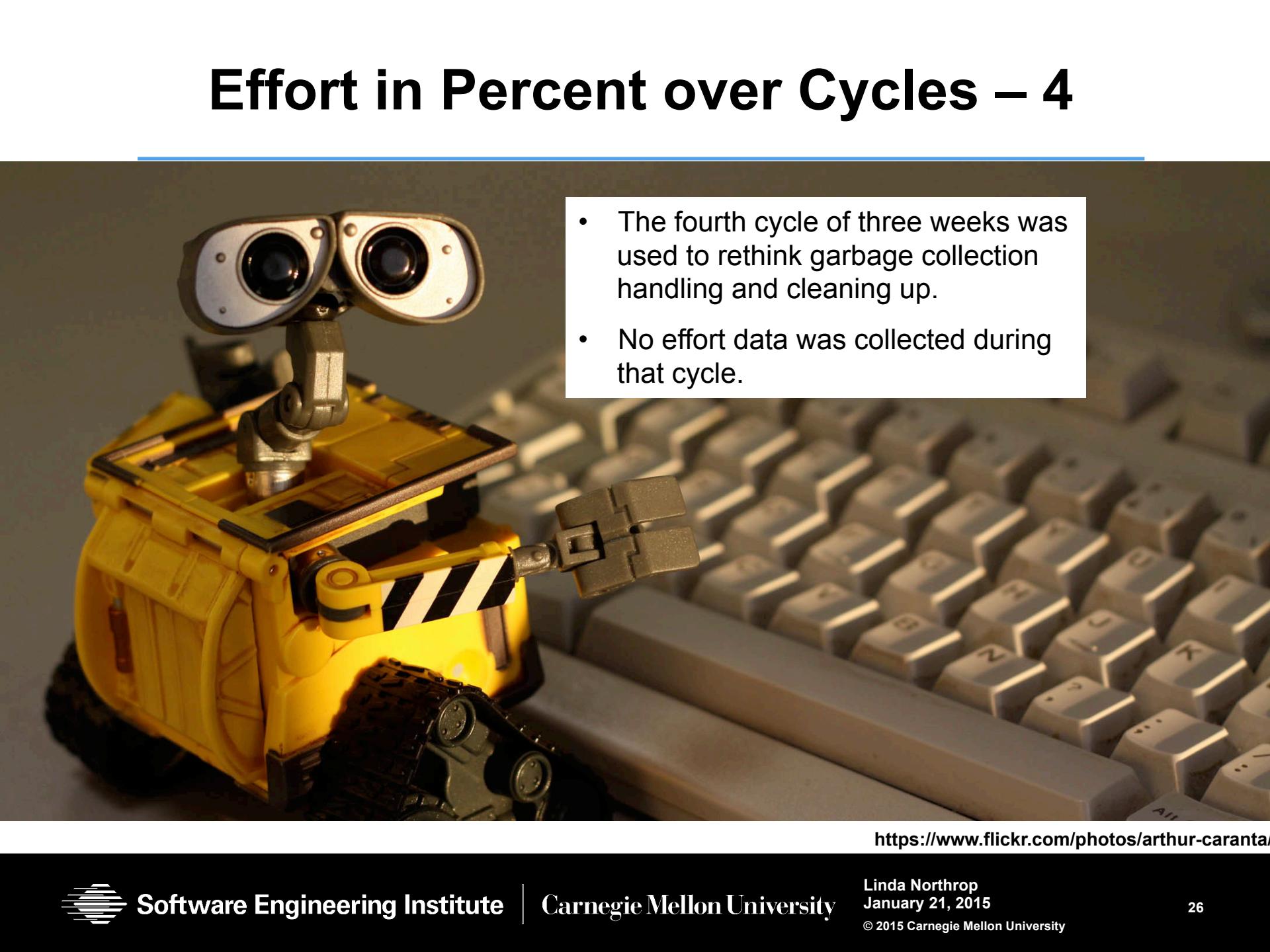
Cycle 3 – 18 Weeks



Reqs: Requirements
HLD/Arch: High level Design / Architecture
DLD: Detailed Design (UML)
Code: Coding (no detailed design)
Test: Testing



Effort in Percent over Cycles – 4

A yellow and white toy robot, resembling the character Wall-E, is positioned on top of a light-colored computer keyboard. The robot has large, expressive eyes on a articulated neck, a rectangular body with a yellow front section and black rear section, and two articulated arms. It is facing towards the right side of the frame. The background is a plain, light-colored wall.

- The fourth cycle of three weeks was used to rethink garbage collection handling and cleaning up.
- No effort data was collected during that cycle.

<https://www.flickr.com/photos/arthur-caranta>



Software Engineering Institute

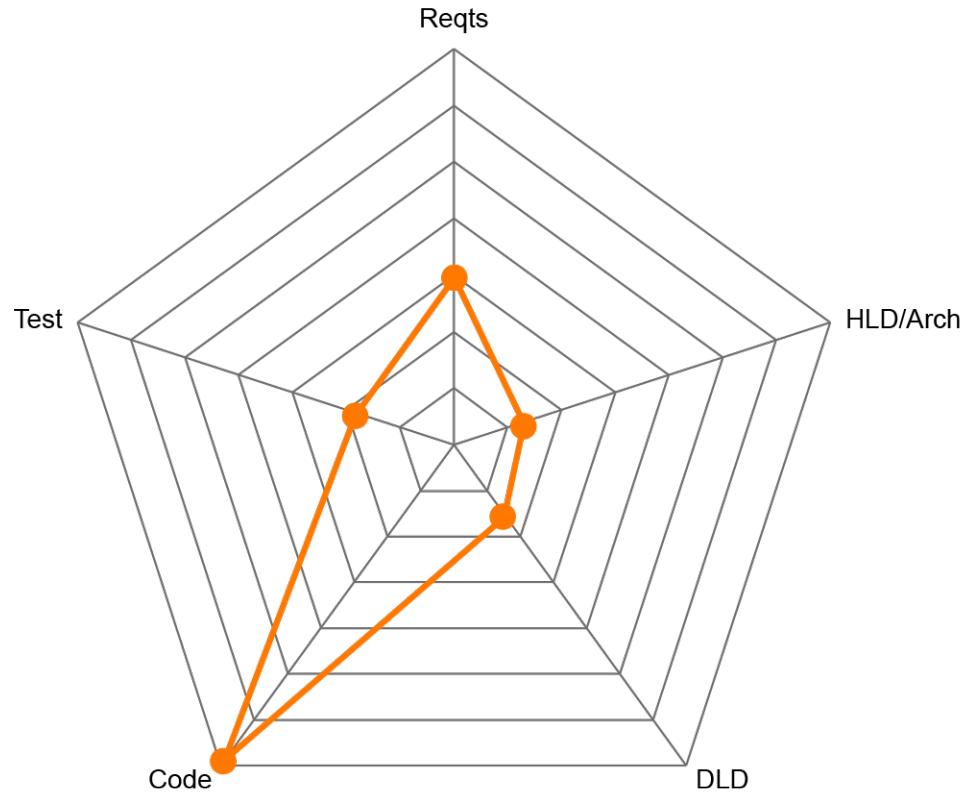
Carnegie Mellon University

Linda Northrop
January 21, 2015

© 2015 Carnegie Mellon University

Effort in Percent over Cycles – 5

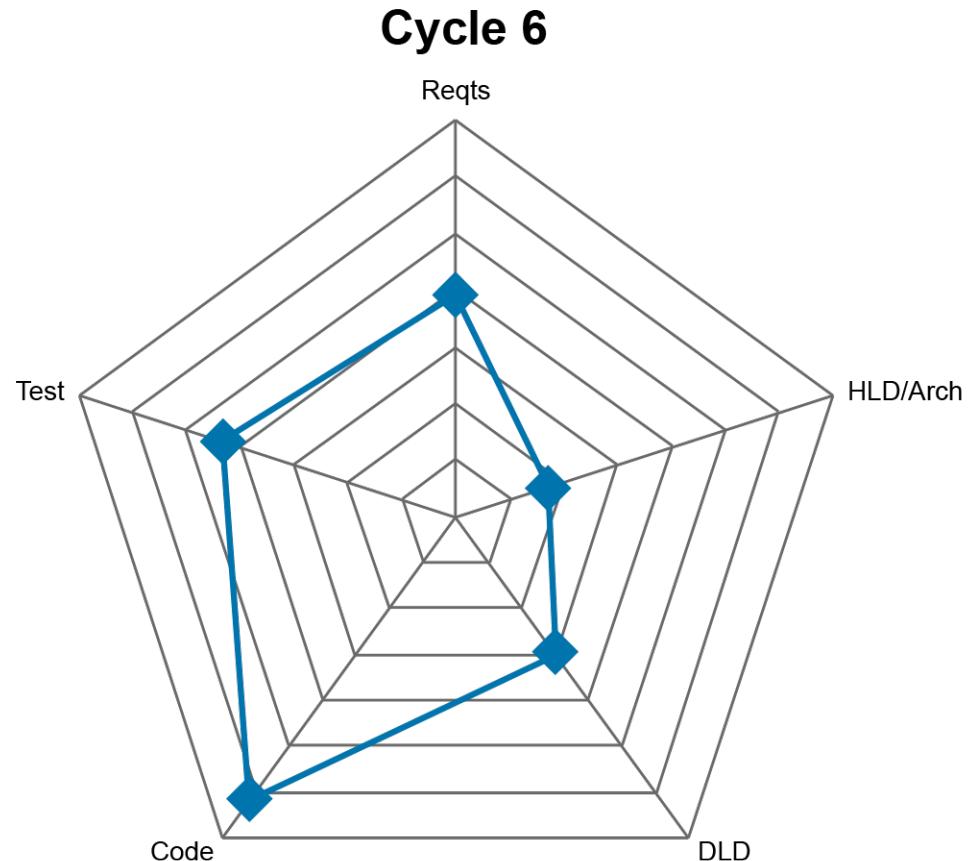
Cycle 5 - 25 Weeks



Reqs: Requirements
HLD/Arch: High level Design / Architecture
DLD: Detailed Design (UML)
Code: Coding (no detailed design)
Test: Testing



Effort in Percent over Cycles – 6



Reqs: Requirements
HLD/Arch: High level Design / Architecture
DLD: Detailed Design (UML)
Code: Coding (no detailed design)
Test: Testing



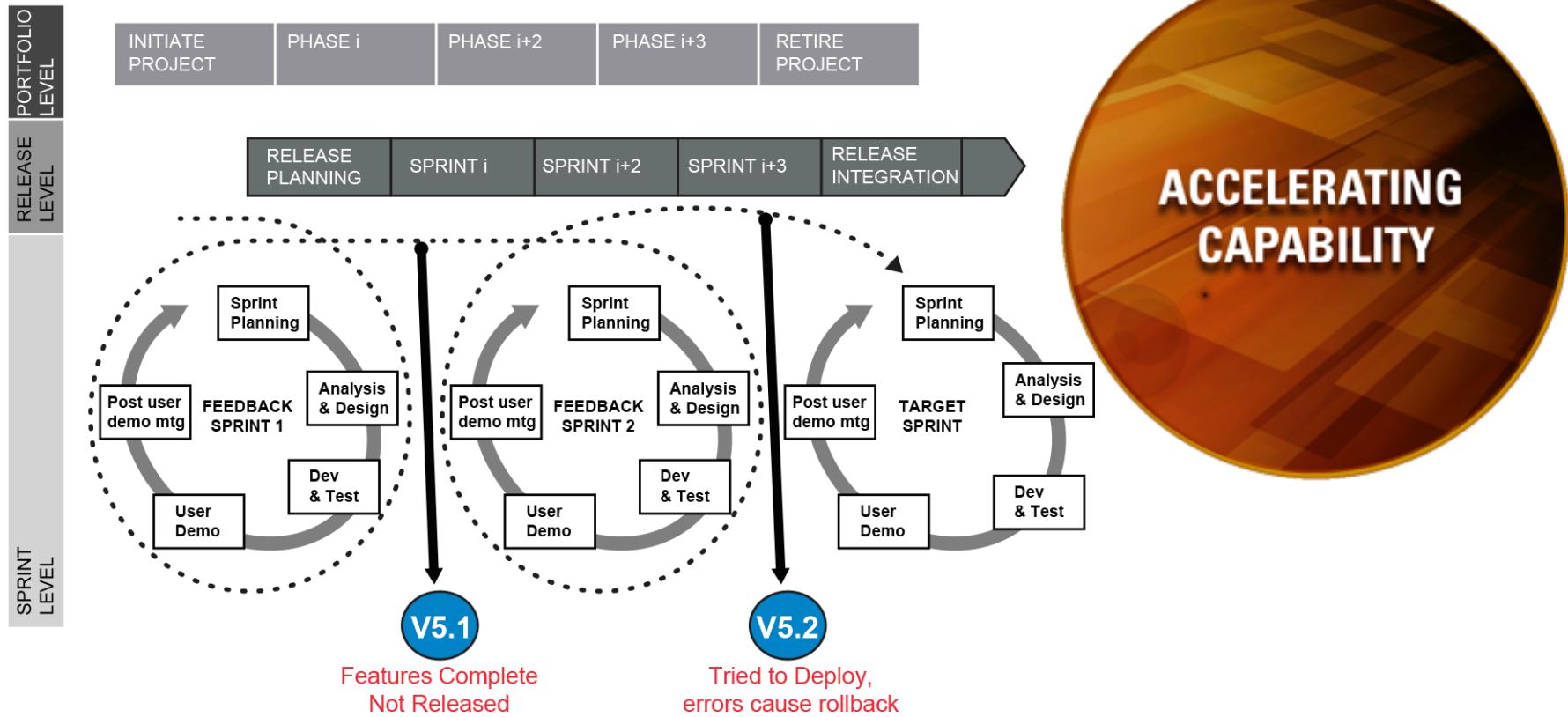
Results

| Results | Target | Actual |
|---|--------|---------|
| Latency | 1ms | 0.1ms |
| Throughput (transactions per second) | 1,000 | 200,000 |
| Schedule (months) | 18 | 17 |
| Quality (defects/KLOC found during validation testing) | 0.25 | 0.1 |



Deployment Challenges

The **DevOps** movement continues what Agile started.



DevOps: State of the Practice

Focus is on

- culture and teaming
- process and practices
 - value stream mapping
 - continuous delivery practices
 - *Lean* thinking
- tooling, automation, and measurement
 - tooling to automate repetitive tasks
 - static analysis
 - automation for monitoring architectural health
 - performance dashboards



Architecture and DevOps

Design decisions
that involve deployment-
related limitations can
blindside teams.



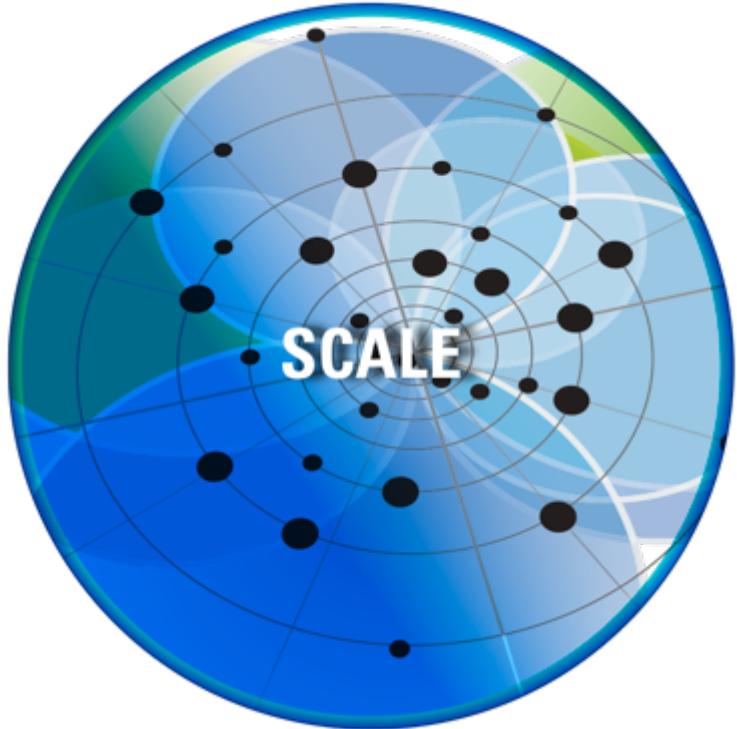
DevOps Tips

- Don't let designing for deployability be an afterthought.
- Use measurable deployability quality attributes.
- Consider architectural tactics that promote modifiability, testability, and operational resilience.
- Use architectural abstractions to reason about deployability implications of design options and tradeoffs.
- Establish monitoring mechanisms.



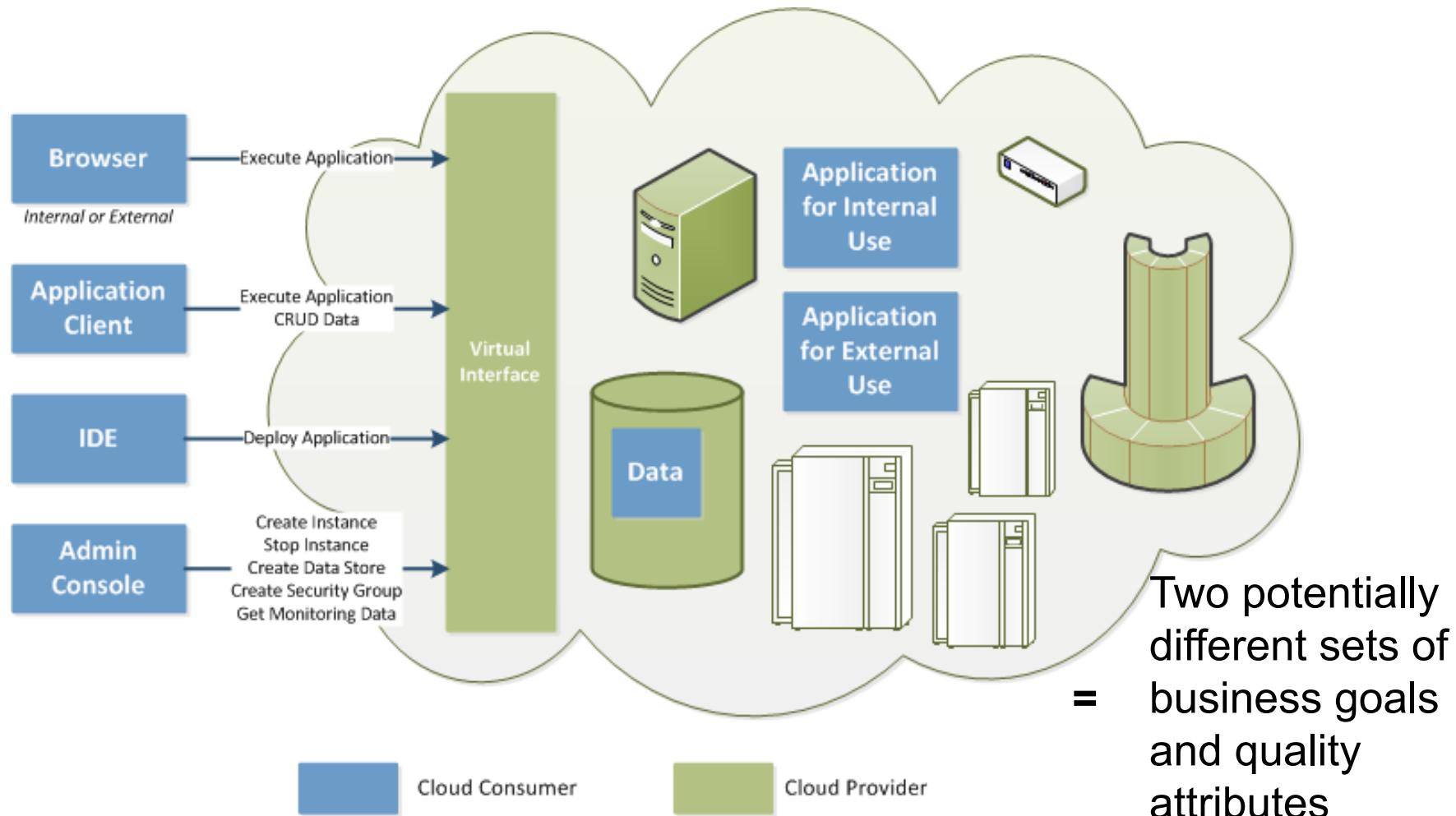
Architecture and Scale

- Cloud strategies
- Cloud strategies for mobility
- Big data



“Scale Changes Everything”

Two Perspectives of Software Architecture in Cloud Computing



Cloud Computing and Architecting

- SLAs cannot prevent failures.
- In cloud environments,
 - cloud consumers have to design and architect systems to account for lack of full control over important quality attributes.
 - cloud providers have to design and architect infrastructures and systems that provide the most efficient way to manage resources and keep promises made in SLAs.



Mobile Device Trends



Architecture Trends: Cyber-Foraging

- Edge Computing
- Using external resource-rich surrogates to augment the capabilities of resource-limited devices
 - code/computation offload
 - data staging
- Industry is starting to build on this concept to improve mobile user experience and decrease network traffic.
- Our research: cloudlet-based cyber-foraging
 - brings the cloud closer to the user

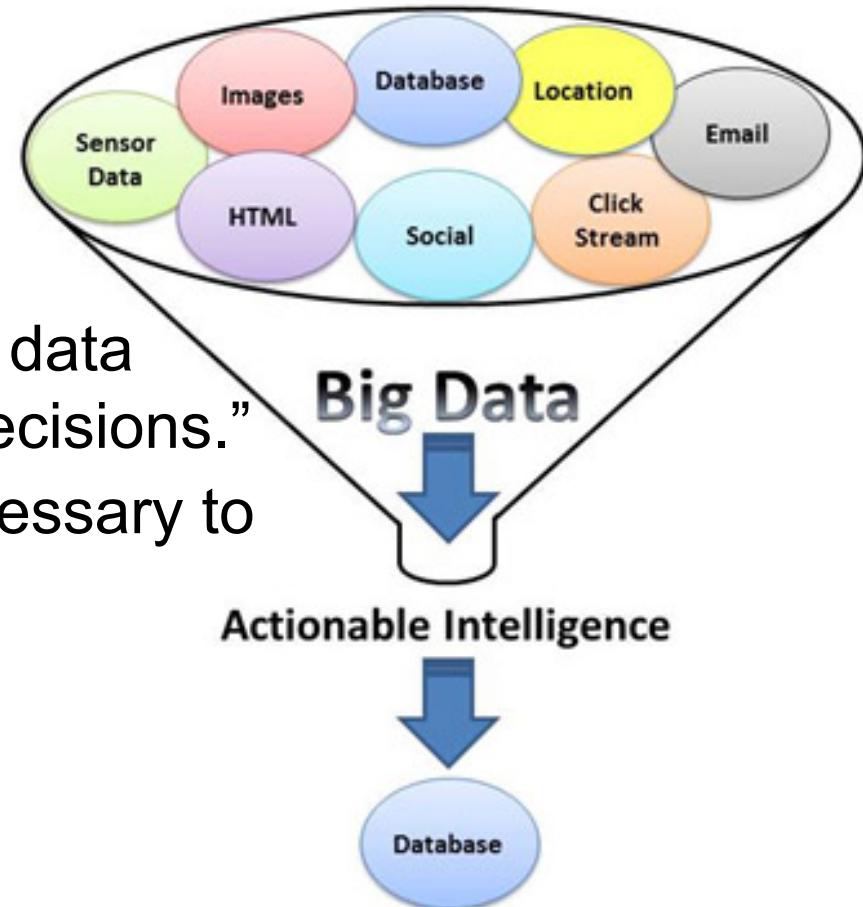


Cisco Systems
Fog Computing



Big Data Systems

- Two very distinct but related technological thrusts
 - Data analytics
 - Infrastructure
- Analytics is typically a massive data reduction exercise – “data to decisions.”
- Computation infrastructure necessary to ensure the analytics are
 - fast
 - scalable
 - secure
 - easy to use



Big Data – State of the Practice

“The problem is not solved”

Building scalable, assured big data systems is hard.



Building scalable, assured big data systems is expensive.



Big Data Survey

55%

OF BIG DATA PROJECTS
ARE NOT COMPLETED

WHEN IT COMES TO BIG DATA PRO
THE MOST SIGNIFICANT CHALLENGE



58%

INACCURATE SCOPE

80%

FINDING TALENT

76%

FINDING THE
RIGHT TOOLS

73%

UNDERSTANDING

TOP REQUIREMENTS OF BIG DATA SOLUTIONS

#1

EASE OF
MANAGEMENT

#2

ABILITY
TO SCALE



<http://visual.ly/cios-big-data>



Software Engineering Institute

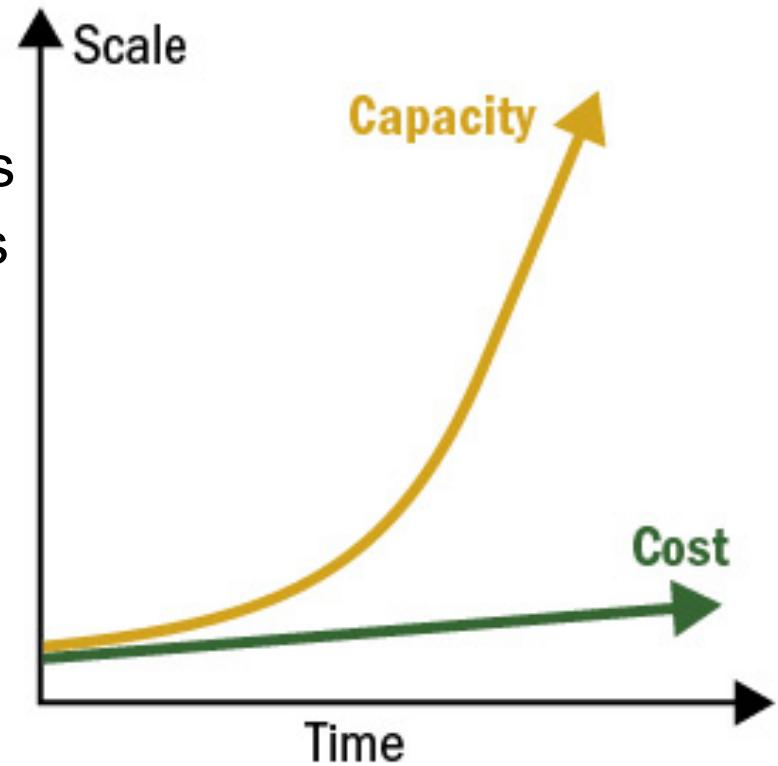
Carnegie Mellon University

Linda Northrop
January 21, 2015

© 2015 Carnegie Mellon University

Architecture and Big Data

- System costs must grow more slowly than system capacity.
- Approaches
 - scalable software architectures
 - scalable software technologies
 - scalable execution platforms
- Scalability reduces as implementation complexity grows.
- NoSQL models are not created equal.
- You can't manage what you don't monitor.



Our Current Research

- Lightweight Evaluation and Architecture Prototyping for Big Data (LEAP4BD)
- QuABase: A Knowledge Base for Big Data System Design
 - semantics-based knowledge model
 - general model of software architecture knowledge
 - populated with specific big data architecture knowledge
 - dynamic, generated, and queryable content
 - knowledge visualization



Architecture and Software Assurance

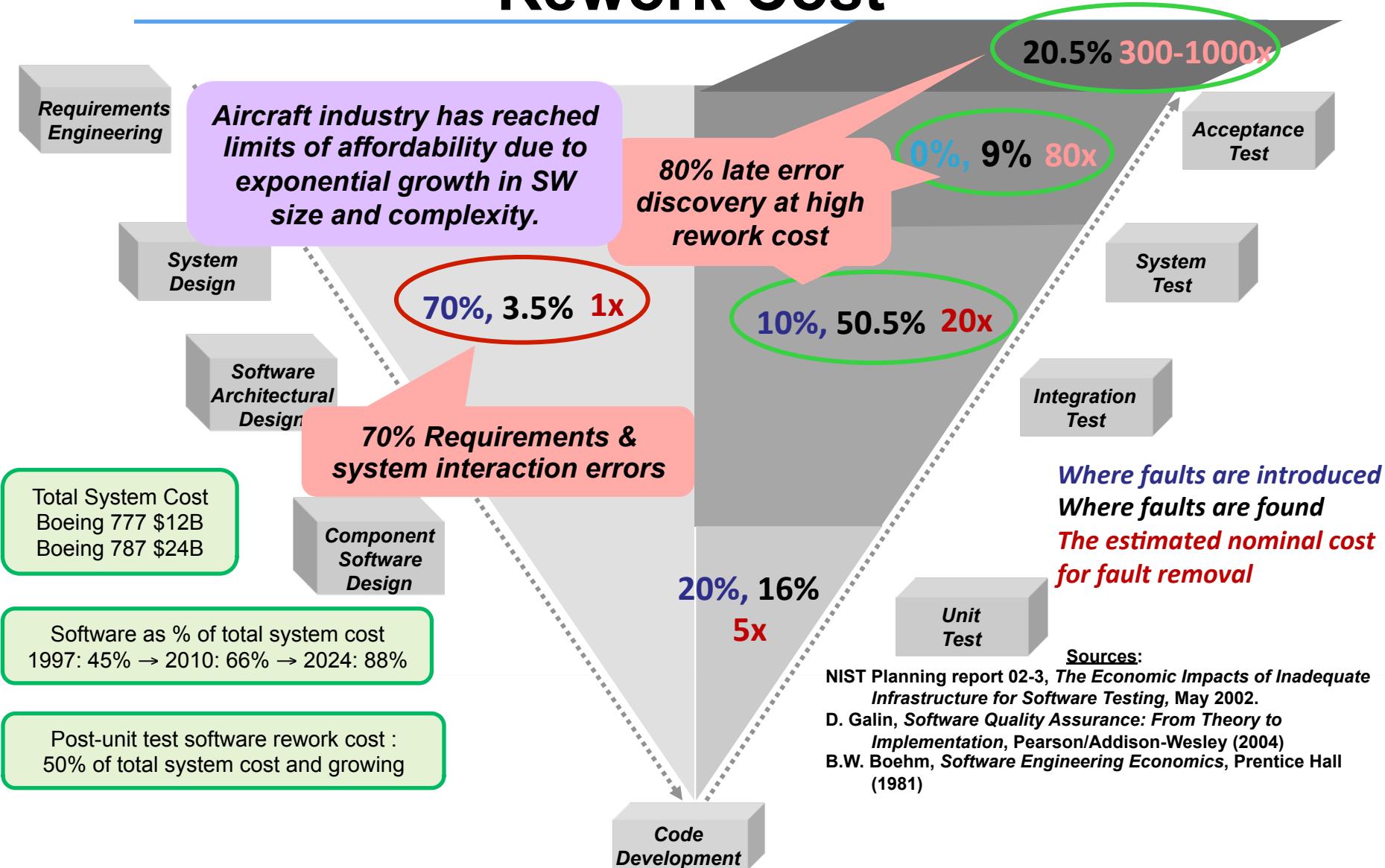


Architectural Models

- capture architecture in a form amenable to analysis, which contributes to assurance
- range from informal (e.g., visio diagrams) to formal (e.g., with precisely defined execution semantics)
- In safety critical systems formality is warranted.



High Fault Leakage Drives Major Increase in Rework Cost



SAE Architecture Analysis & Design Language (AADL) Standard Suite (AS-5506 Series)

- Core AADL language standard (V2.1-Sep 2012, V1-Nov 2004)
 - Strongly typed language with well-defined semantics
 - Textual and graphical notation
 - Standardized XMI interchange format

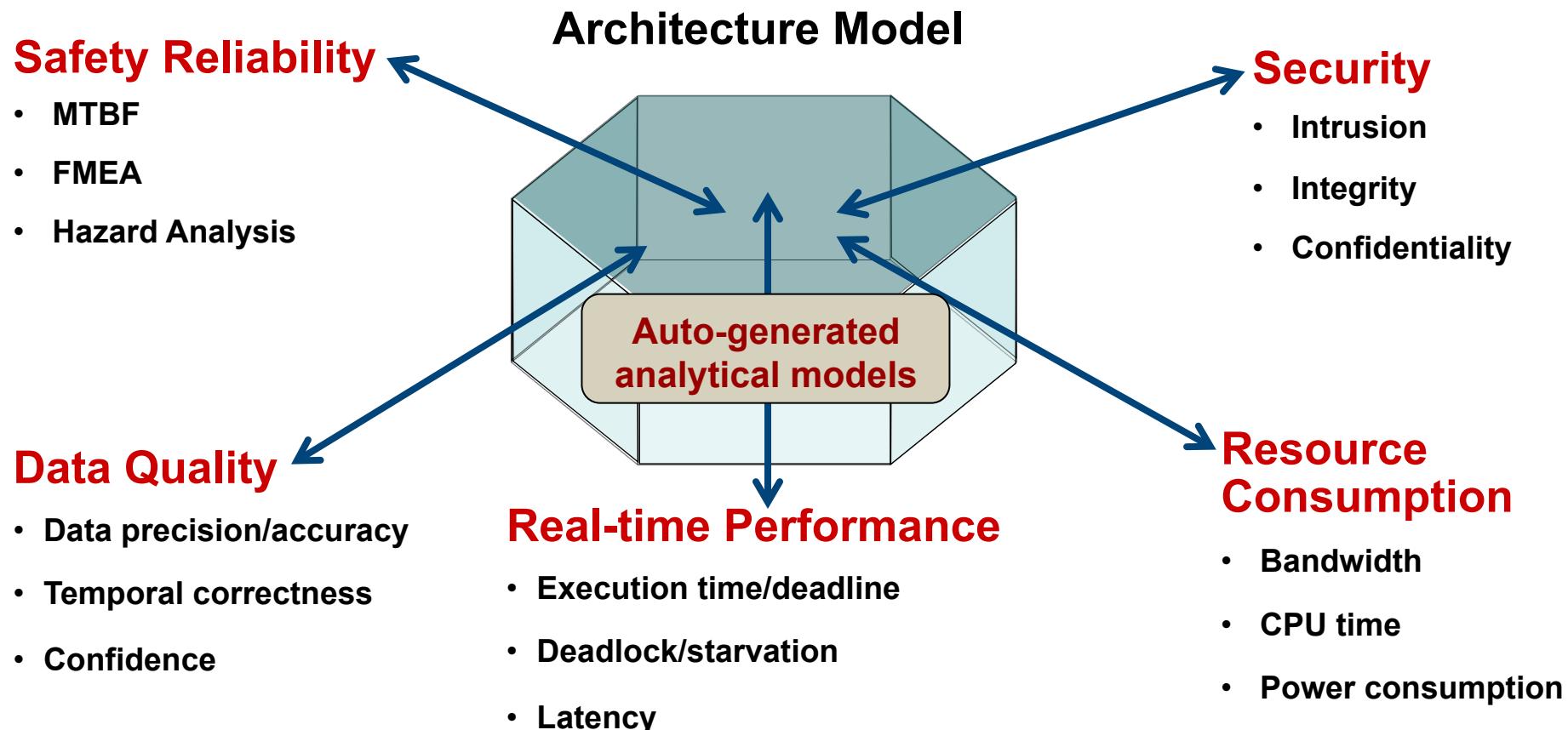
Standardized AADL Extensions

- Error Model language for safety, reliability, security analysis
- ARINC653 extension for partitioned architectures
- Behavior Specification Language for modes and interaction behavior
- Data Modeling extension for interfacing with data models (UML, ASN.1, ...)

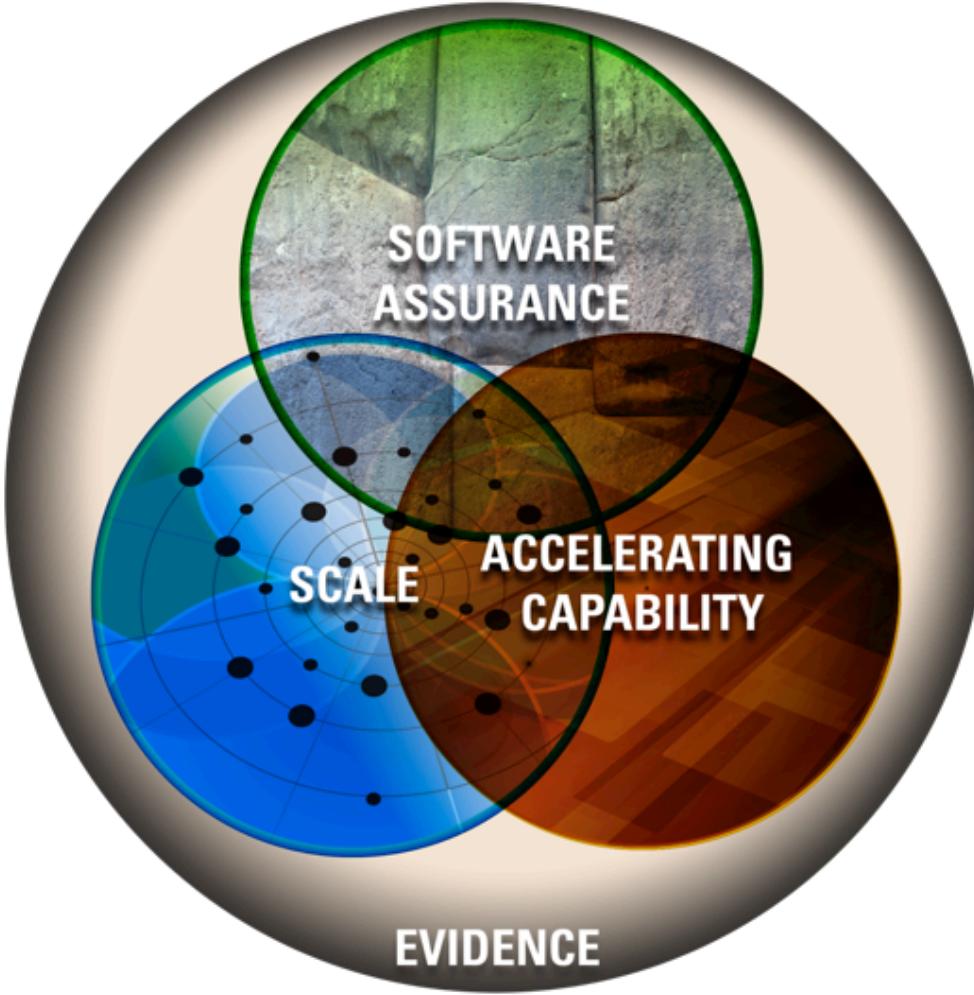


Architecture-Centric Quality Attribute Analyses

Single Annotated Architecture Model Addresses Impact Across Operational Quality Attributes



Conclusion



- Software architecture principles and their importance persist.
- Change brings new challenges.
- Software architecture practices and research are key to meeting these challenges.
- Much remains to be done.



This Is the Work of Many

At the SEI

- Felix Bachmann
- Stephany Bellomo
- Peter Feiler
- Ian Gorton
- James Ivers
- Rick Kazman
- John Klein
- Mark Klein
- Grace Lewis
- Ipek Ozkaya
- Rod Nord
- and many more...



More Information

Related Blogs:

<http://blog.sei.cmu.edu/archives.cfm/category/architecture>

Technical Debt:

http://www.sei.cmu.edu/architecture/research/arch_tech_debt/arch_tech_debt_library.cfm

Agile Architecting:

<https://www.sei.cmu.edu/architecture/research/agile-architecting/index.cfm>

Cloudlets:

<http://www.sei.cmu.edu/mobilecomputing/research/tactical-cloudlets/index.cfm>

AADL and Model-Based Engineering:

<http://www.sei.cmu.edu/architecture/research/model-based-engineering/index.cfm>



Software Engineering Institute

Carnegie Mellon University

Linda Northrop
January 21, 2015

© 2015 Carnegie Mellon University

Contact Information

Linda Northrop

SEI Fellow

Chief Scientist

Software Solutions Division

Telephone: 412-268-7638

Email: lmn@sei.cmu.edu

Website: <http://www.sei.cmu.edu/architecture>



U.S. Mail:

Software Engineering Institute
Carnegie Mellon University
Pittsburgh, PA 15213-3890

SEI Fax: 412-268-5758



Software Engineering Institute

Carnegie Mellon University

Linda Northrop
January 21, 2015

© 2015 Carnegie Mellon University