Reflections on Software Architecture

Linda Northrop

SEI Fellow

Software Engineering Institute
Carnegie Mellon University
Pittsburgh, PA 15213

Notices

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.

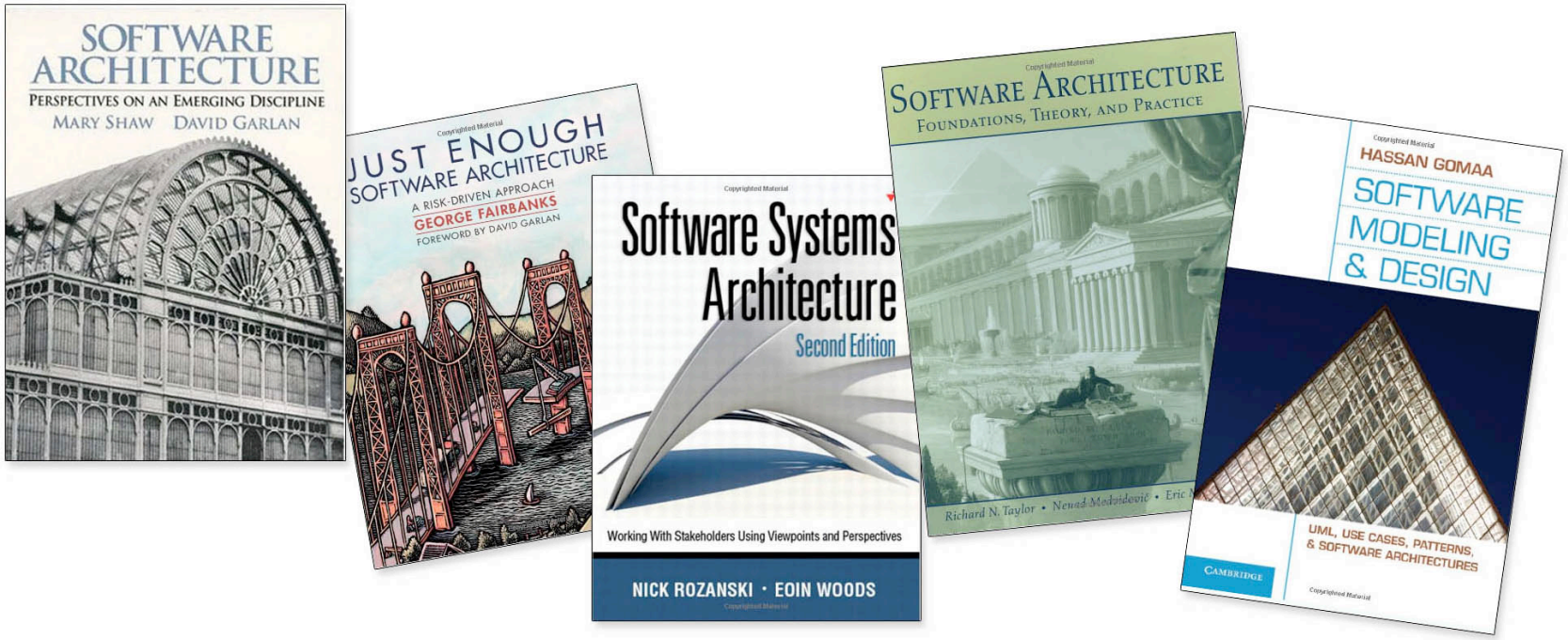
Software Architecture

The quality and longevity of a software-reliant system is largely determined by its architecture.

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

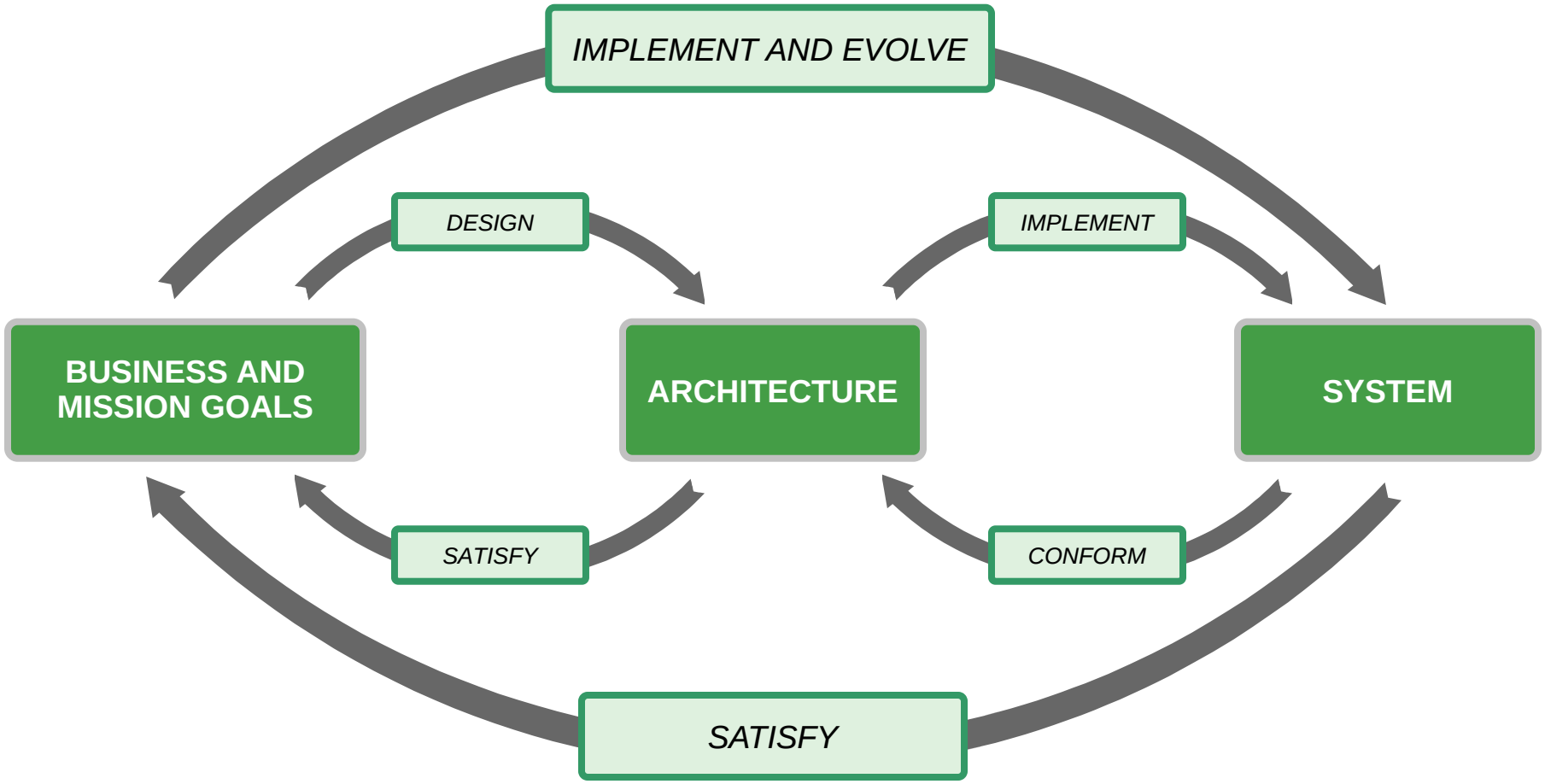
Quality attributes

- 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

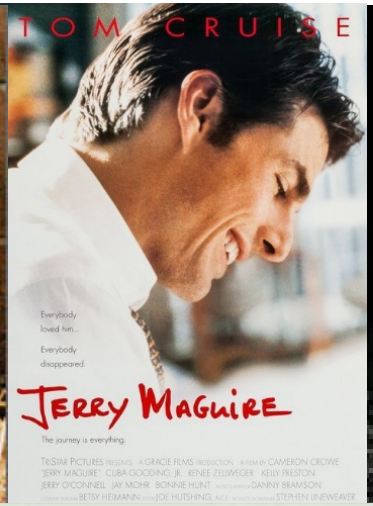
- Performance
- Availability
- Interoperability
- Modifiability
- Usability
- Security
- Etc.

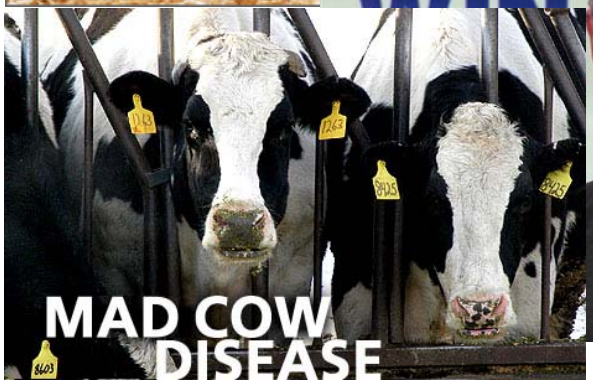
Central Role of Architecture



‘90’s – “The Golden Age of Software Architecture”

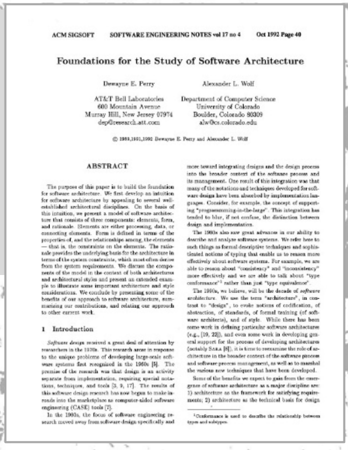
Clements and Shaw: IEEE Software. March/April 2006.





‘90’s: Early Foundations

PRINCIPLES


ACM SIGSOFT SOFTWARE ENGINEERING NOTES vol 17 no 4 Oct 1992 Page 40

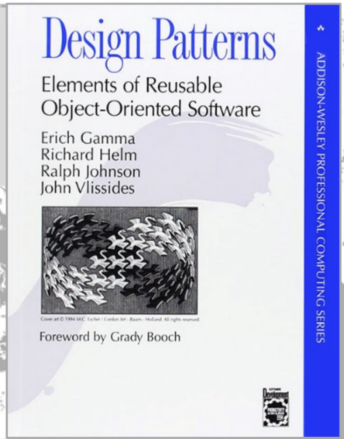
Foundations for the Study of Software Architecture

Dewayne E. Perry — AT&T Bell Laboratories, 600 Mountain Avenue, Murray Hill, New Jersey 07974

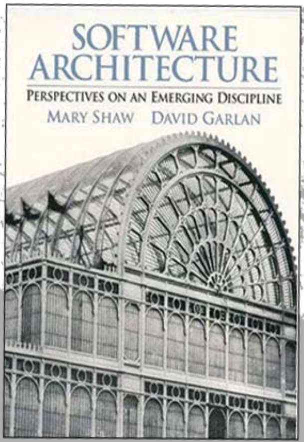
Alexander L. Wolf — Department of Computer Science, University of Colorado, Boulder, Colorado 80309

Perry and Wolf paper

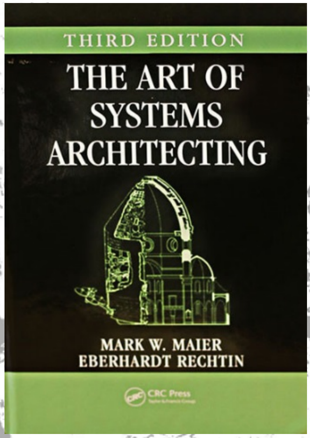
BOOKS



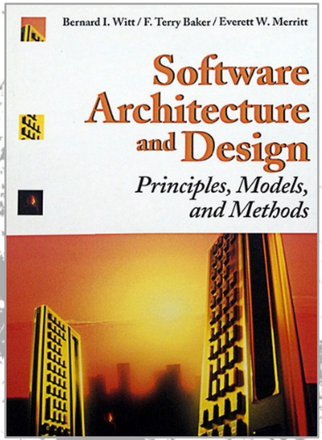
Gamma/Helm/Johnson/Vlissides



Shaw/Garlan



Rechtin/Maier



Witt/Baker/Merritt

SAAM

EVALUATION

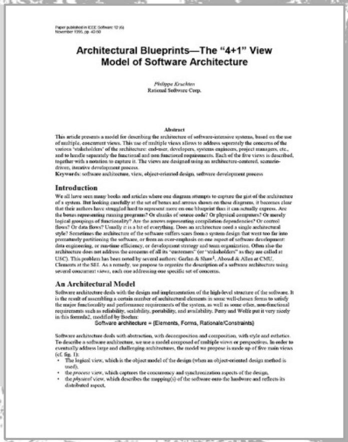

SAAM: A Method for Analyzing the Properties of Software Architectures

Rick Kazman, Len Bass, Gregory Abowd, Mike Webb

Kazman/Bass/Abowd/Webb

4+1 Views

REPRESENTATION


Architectural Blueprints—The “4+1” View Model of Software Architecture

Philippe Kruchten, Rational Software Corp.

Kruchten

Acme, Darwin, Wright


A Formal Approach to Software Architecture

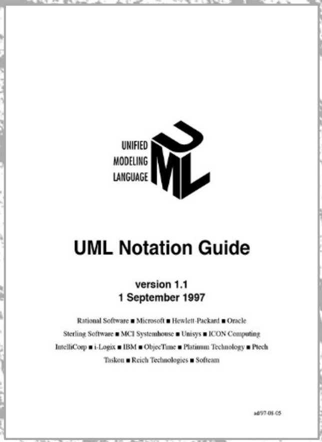
Robert J. Allen

May 1997

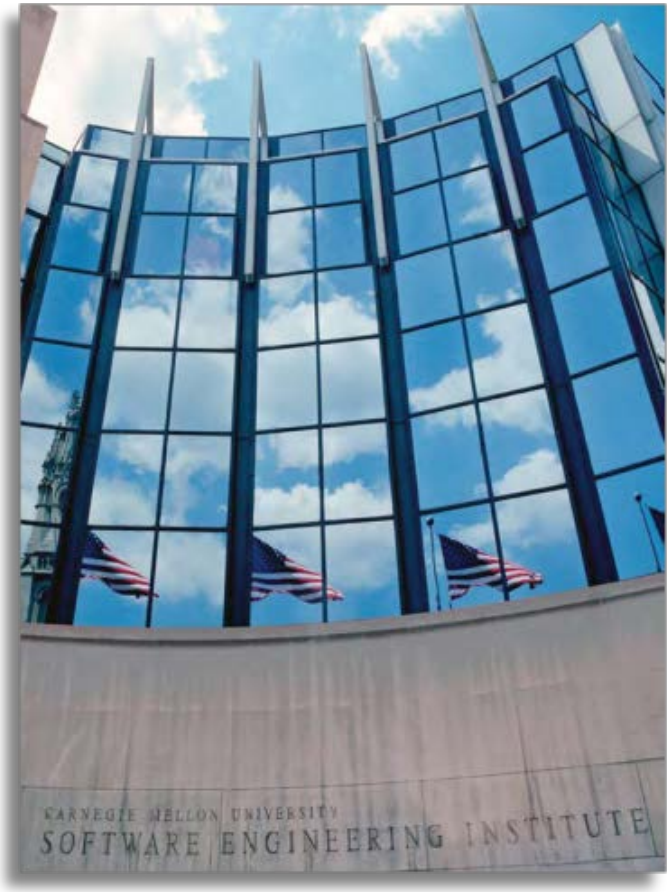
CMU-CS-97-144

Garlan/Jackson/Shaw/Boehm

UML 1.1



SEI's Role



The SEI developed principles, methods, foundations, techniques, tools, and materials in support of creating, fostering, and stimulating widespread transition of architecture-centric engineering.

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 and tactics

Component-based approaches

Company-specific product lines

Model-based approaches

Aspect-oriented approaches

Frameworks and platforms

Standard interfaces

Standards

SOA

Persisting Themes:

- **Modularity**
- **Commonality vs Variability**

Community Gatherings

Research

- WICSA
- CompArch
- European Conference on Software Architecture
- QoSA

- SATURN
- SEI Software Architecture Educators' Workshop Series

Practice

- Software Architect
- ArchConf
- O'Reilly Software Architecture Conference

+ tracks in other research and practitioner events

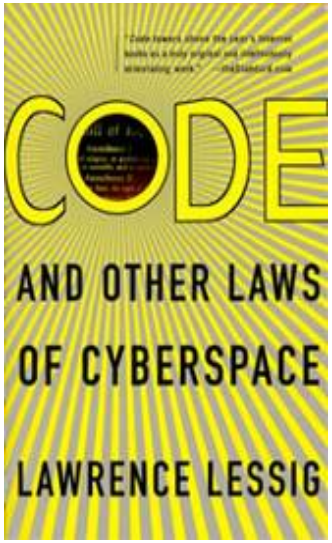
Related Shifts

Incremental approaches

Light-weight software development approaches

Open source

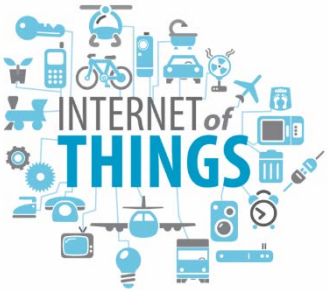
Distributed development environments

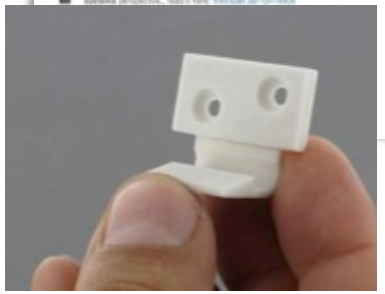


What Changed?

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

Technology Trends





Software Development Trends

docker

GitHub

Application frameworks

Distributed development environments

Cloud strategies

NoSQL

NewSQL

Machine Learning

Static analysis tools

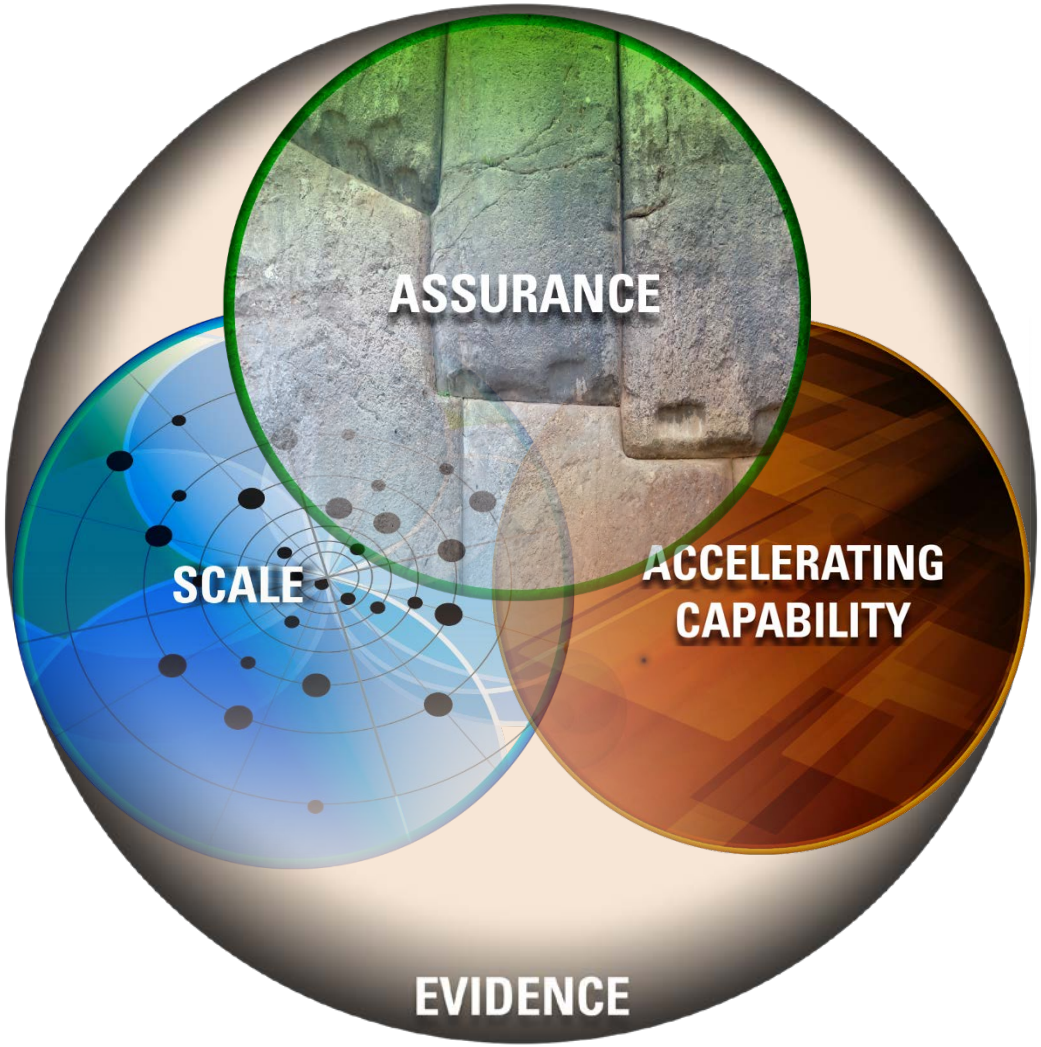
Dashboards

DevOps

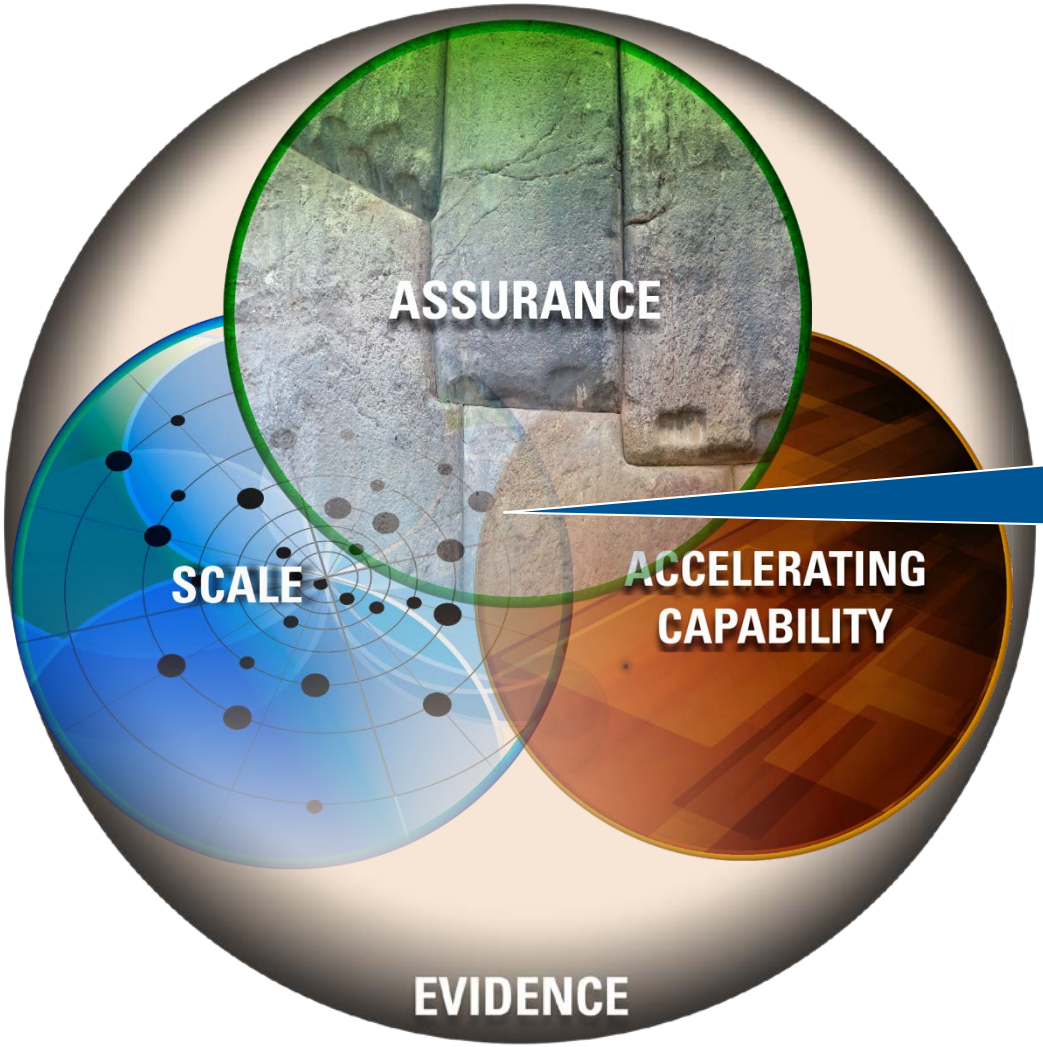
Containers

Microservices

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?

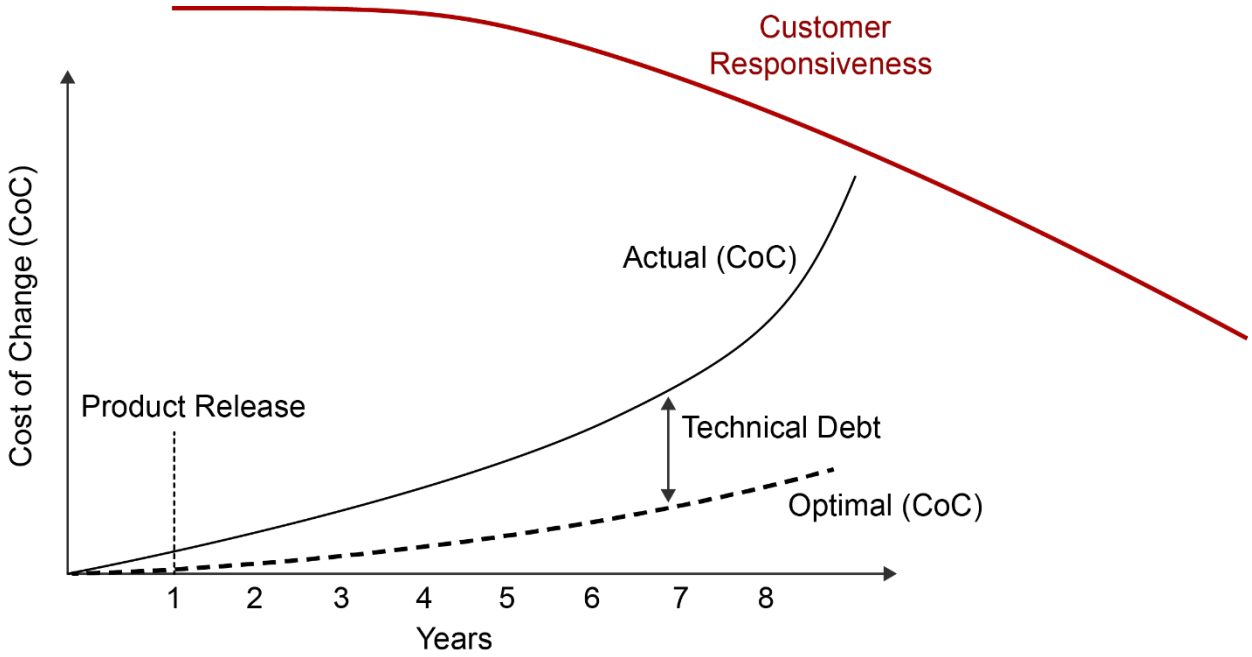
There is a difference between being agile and doing agile.

Agility is enabled by architecture – not stifled by it.

Managing technical debt is key.

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.



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

Graph: Jim Highsmith, Oct 19 2010 **http://jimhighsmith.com/the-financial-implications-of-technical-debt/**

In Research and Practice

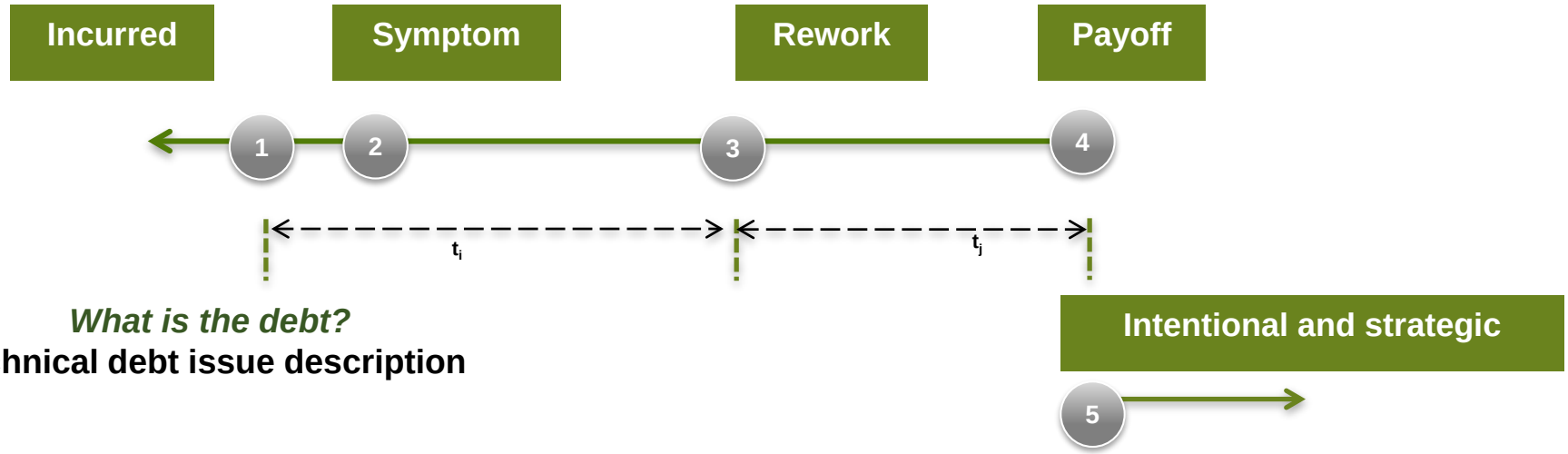
Dagstuhl Workshop on Technical Debt

April 2016

“Two developers ask forgiveness of technical debt at the beginning of the sprint”

Jean-Francois Millet 1857-1859: *Classic Programmer Paintings*

Timeline for Managing Technical Debt

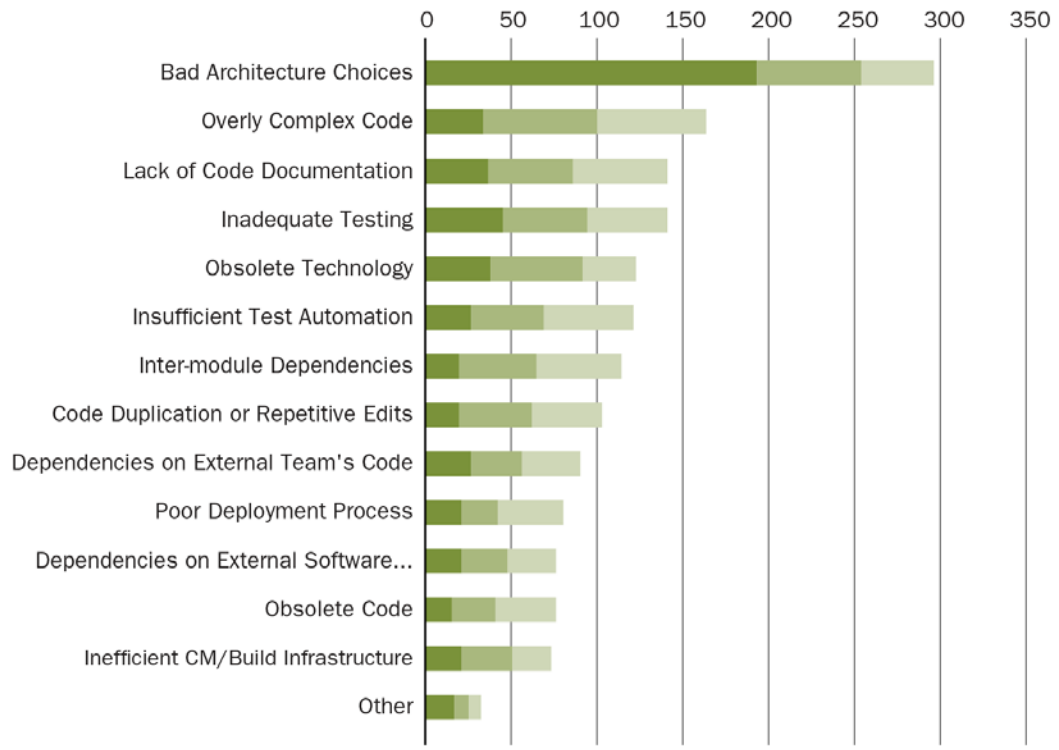


What is the debt?
Technical debt issue description

How does debt accumulate?
Static and architecture analysis

When to pay back debt?
Architecture-focused release planning

Architecture and Technical Debt

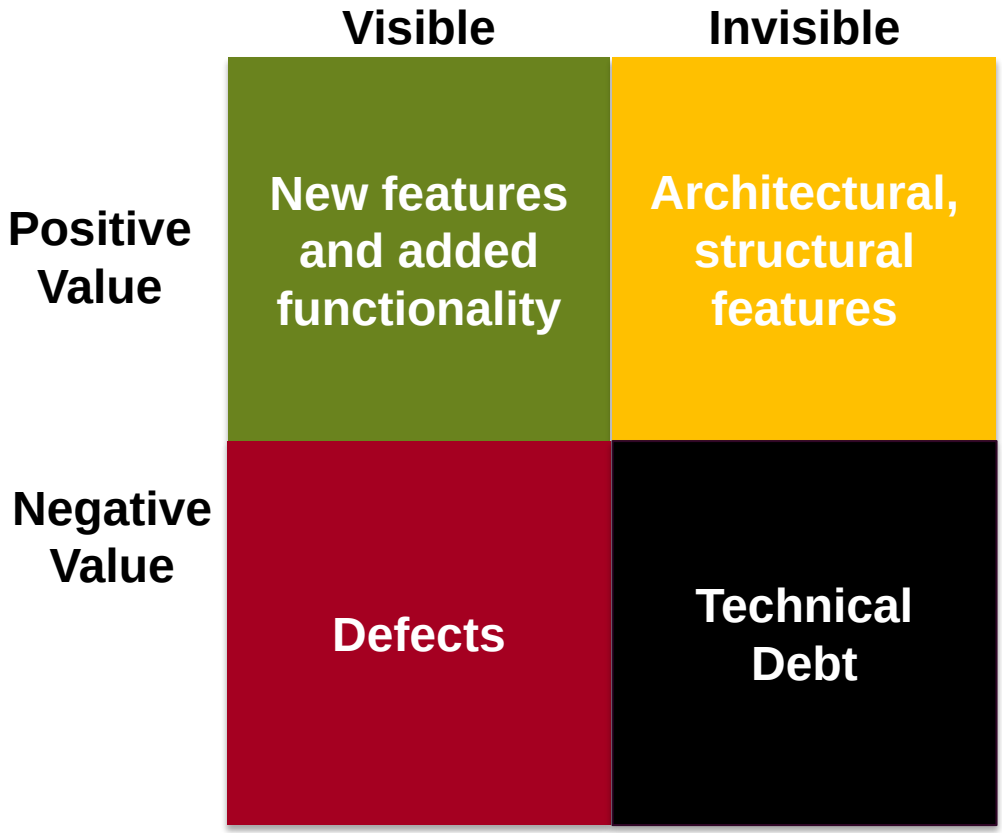


Bad architectural choices rated as the top contributor to technical debt, followed by overly complex code and inadequate testing among over 1800 developers we surveyed. 56% of the respondents ranked architecture among their top three pain points.

A Field Study of Technical Debt

https://insights.sei.cmu.edu/sei_blog/2015/07/a-field-study-of-technical-debt.html

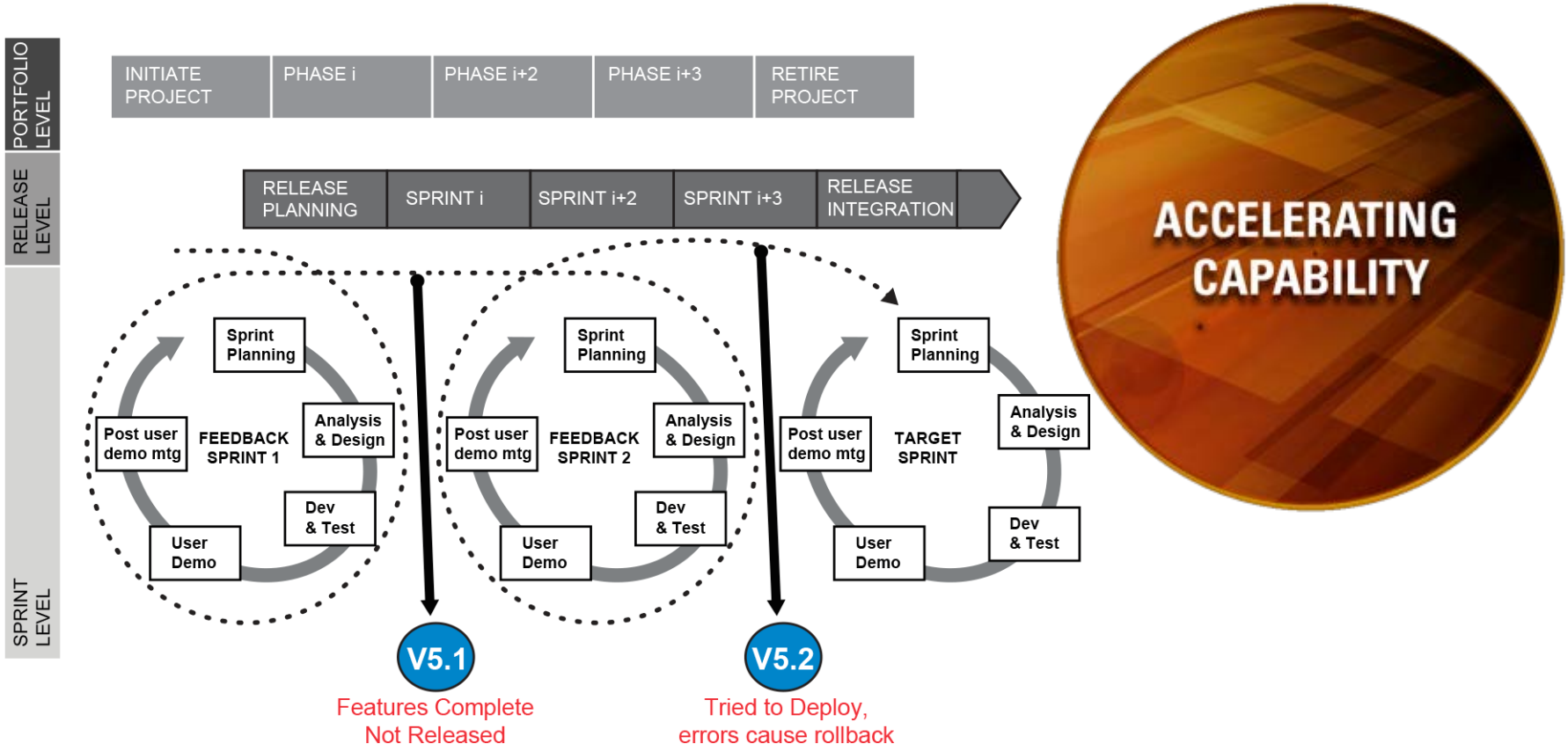
What color is your backlog?



Philippe Kruchten, Robert L. Nord, Ipek Ozkaya: Technical Debt: From Metaphor to Theory and Practice. IEEE Software 29(6): 18-21 (2012)

Continuous Deployment

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



DevOps in 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 manual, repetitive tasks
  - static analysis
  - automation for monitoring architectural health
  - performance dashboards
  - containers

Architecture and DevOps

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

Architectural choices need to facilitate continuous delivery.

Microservices

“Army of contractors migrating to Microservices”

Canaletto, 1733 or 1734 : *Classic Programmer Paintings*

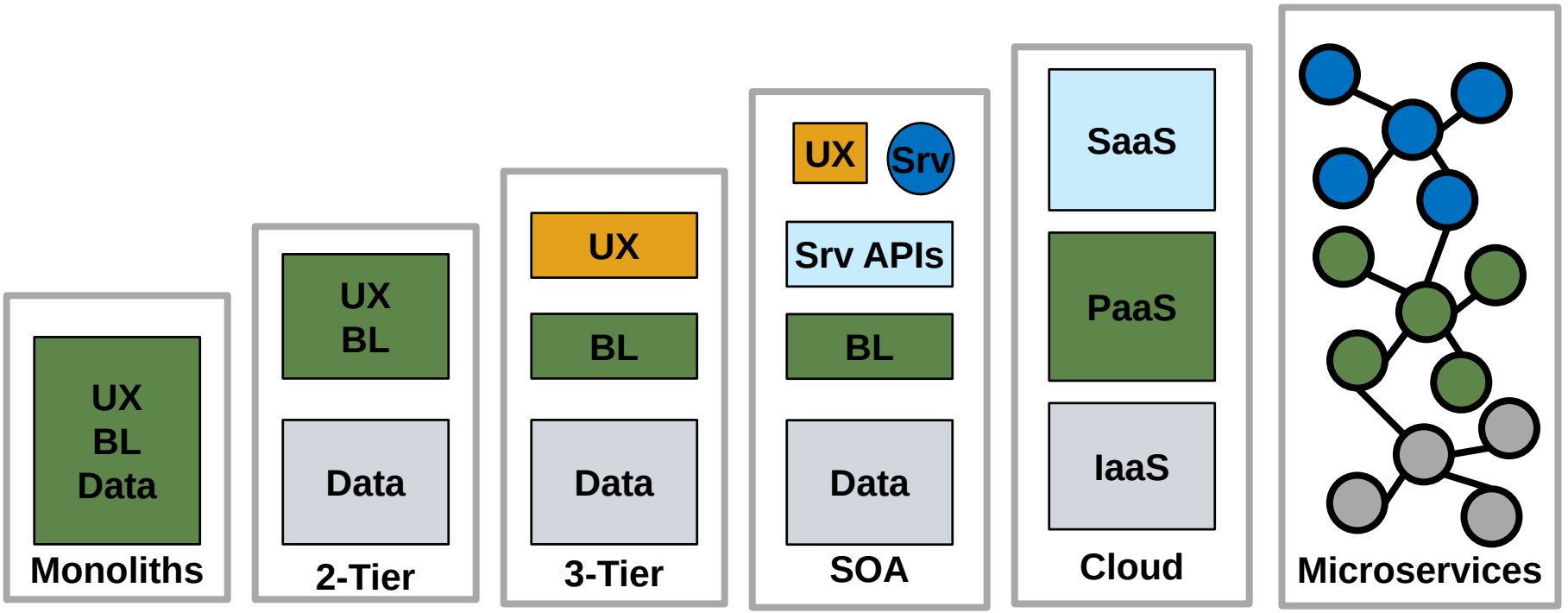
Architecture Evolution for Business Applications

Progress in

- decomposition management
- supporting change
- automation

Persisting Themes:

- **Modularity**
- **Commonality vs Variability**

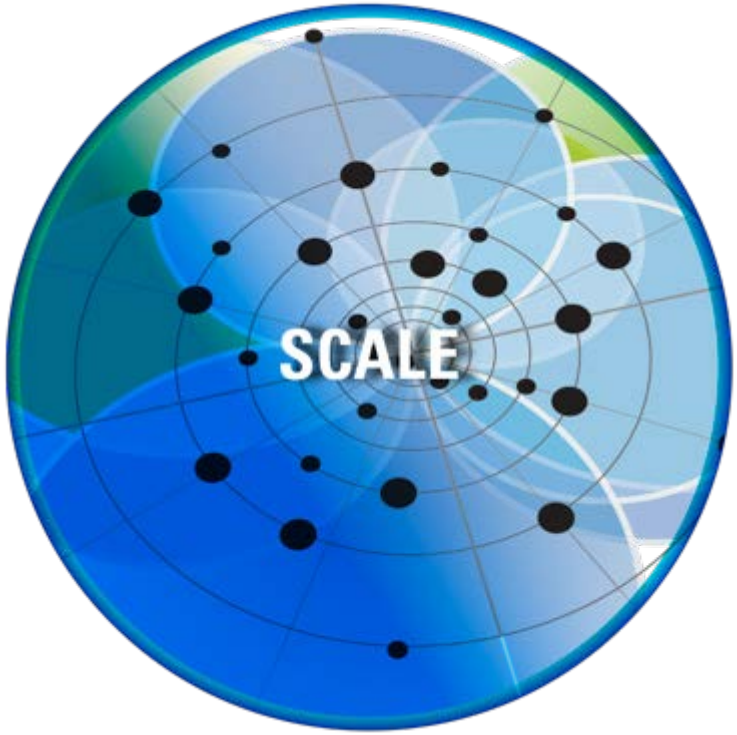


Architecture and Scale

Cloud strategies

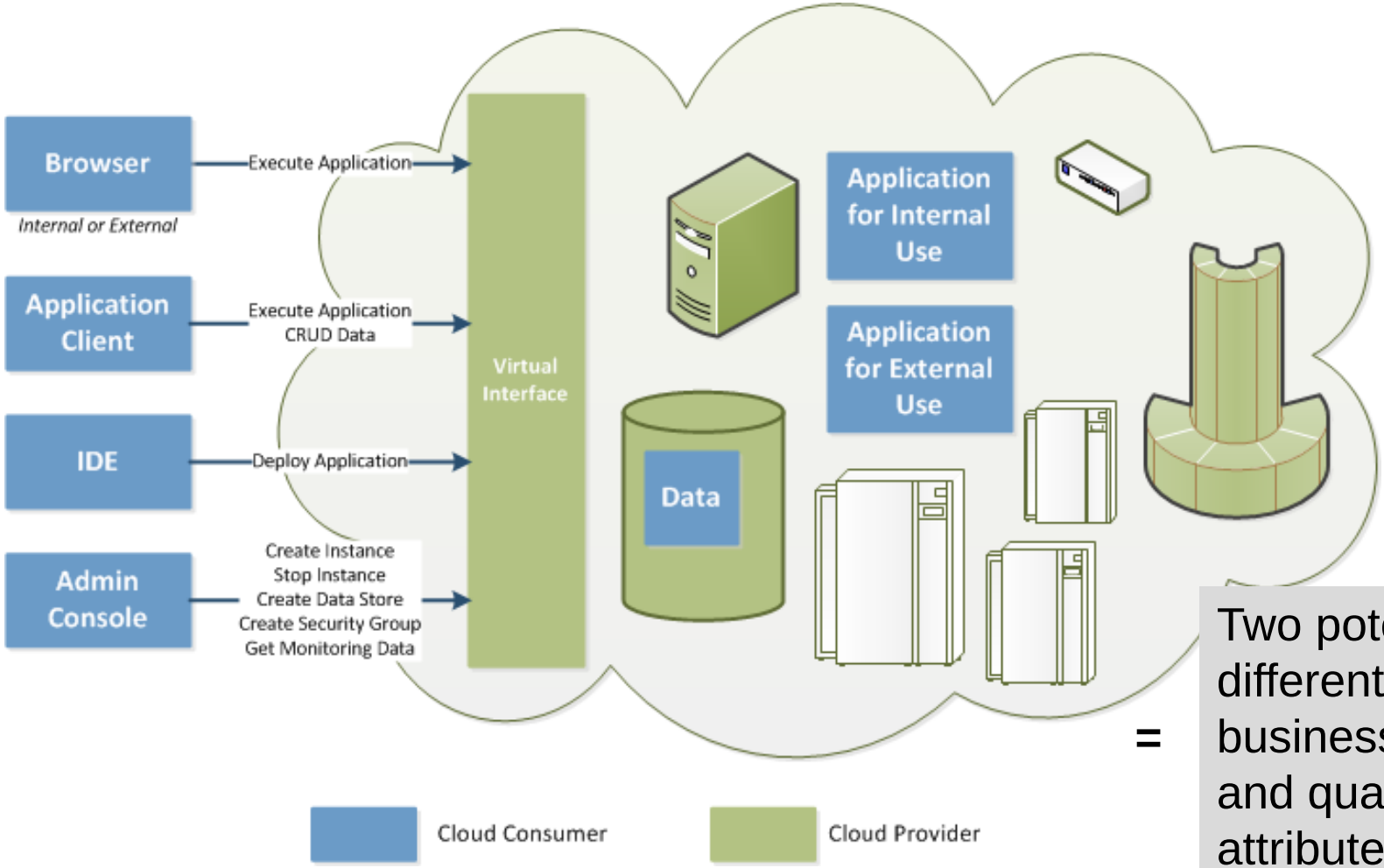
Cloud strategies for mobility

Big data



“Scale Changes Everything”

Two Perspectives of Software Architecture in Cloud Computing



=

Two potentially different sets of business goals and quality attributes

Mobile Device Trends

Architecture Trends: Cyber-Foraging

Nokia Siemens Networks
Liquid Applications

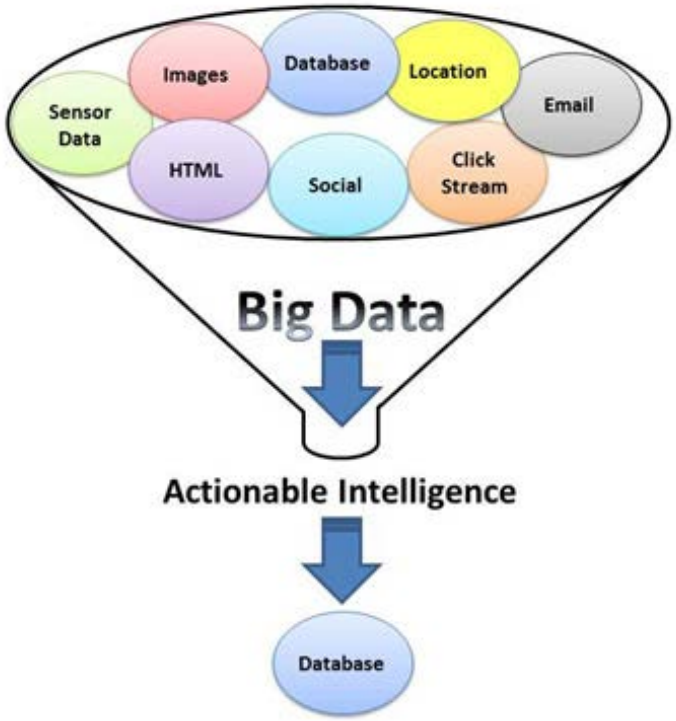
Cisco Systems
Fog Computing

Edge Computing

Using external resource-rich surrogates to augment the capabilities of resource-limited devices

- code/computation offload
- data staging

Big Data Systems



Two very distinct but related technological thrusts

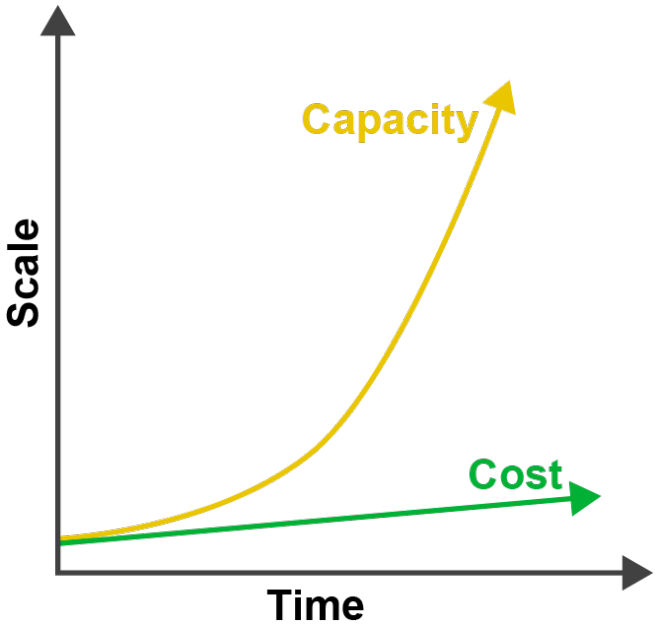
- Data analytics
- Infrastructure

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

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/NewSQL models are not created equal.

You can't manage what you don't monitor.

Architecture and Assurance

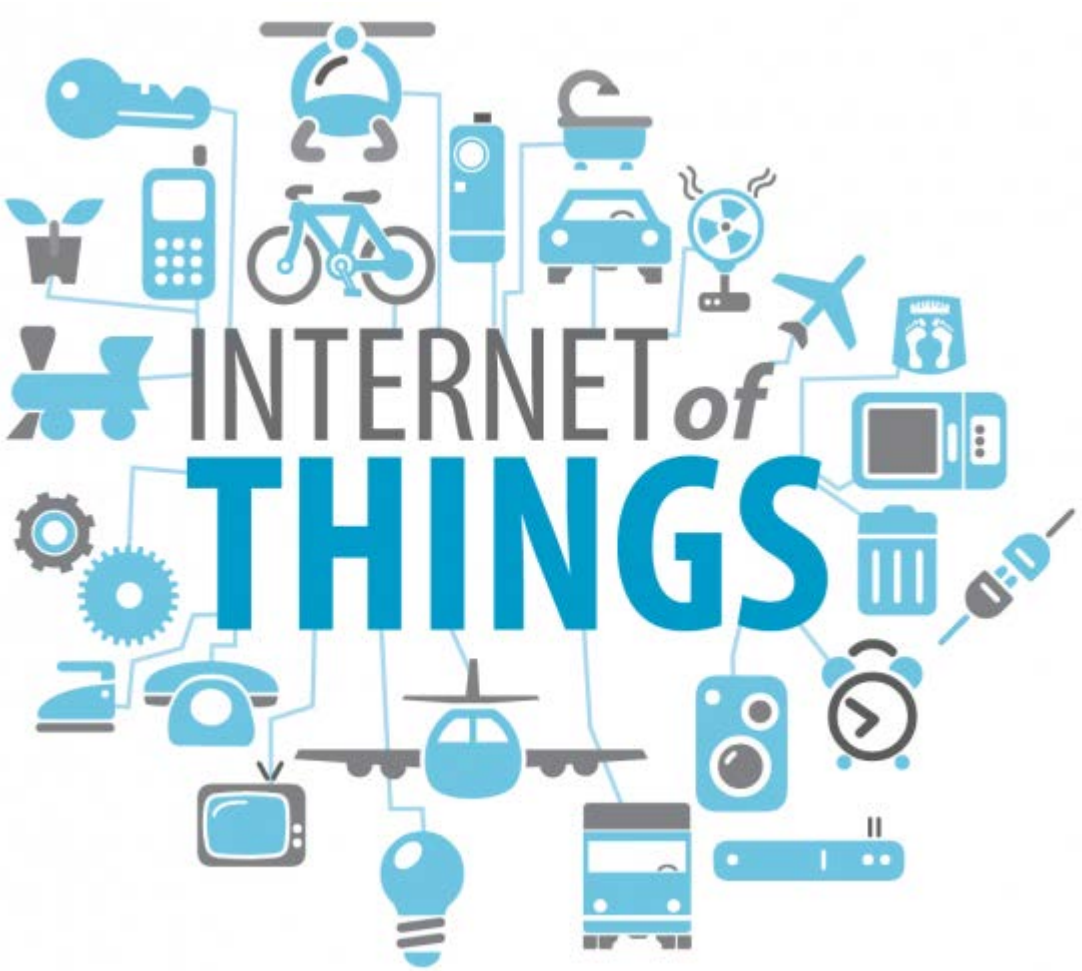
It’s not just about security, but functioning as intended and only as intended.

Supply chains, open source, frameworks, outsourcing introduce unknowns.

Tool chains that generate code, configuration files, etc. introduce unknowns.

Consequences include operational failures, security and privacy compromises, reputational impact, etc.

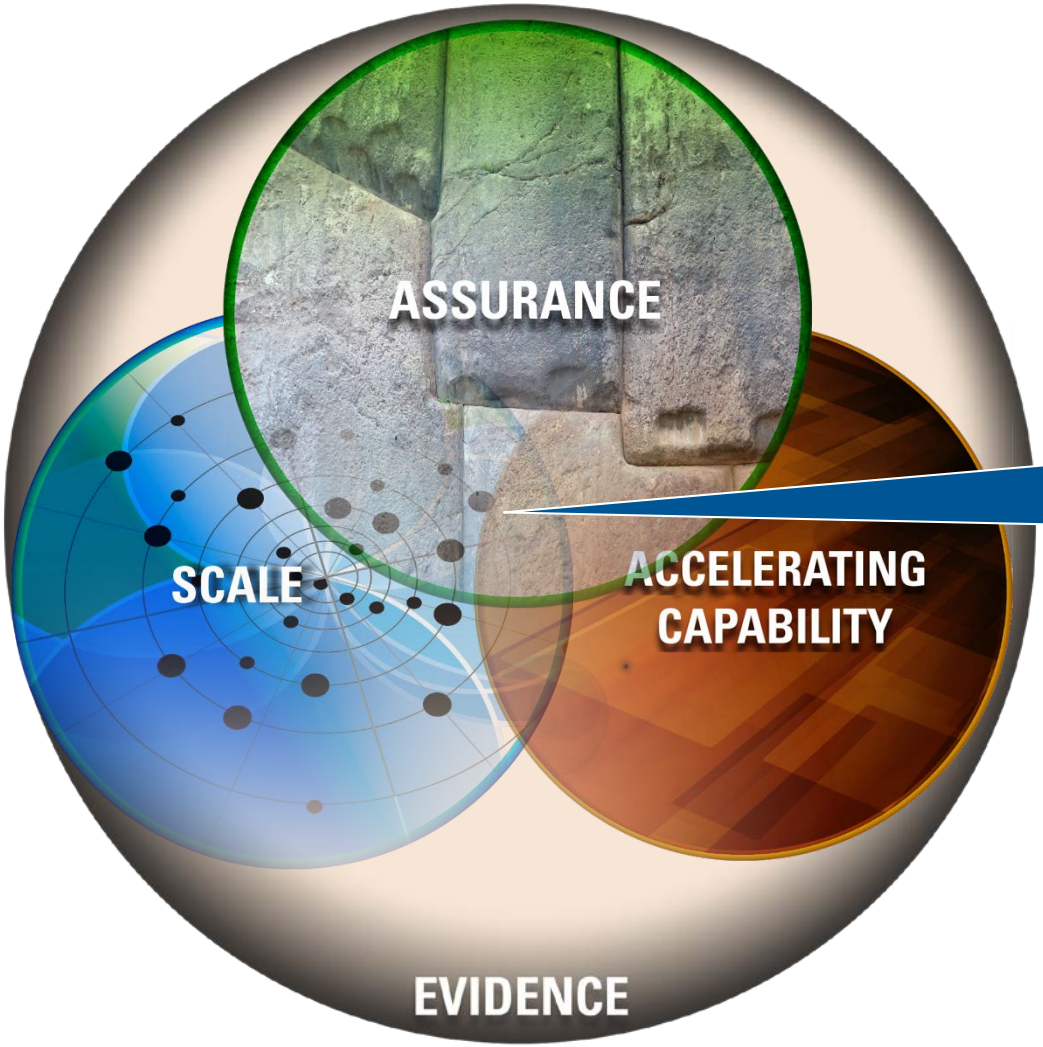
Assurance is Imperative



Analysis and Architectural Models

- evaluation is not a phase – ongoing analysis is required
- capture architecture in a form amenable to ongoing analysis
  - range from informal (e.g., visio diagrams) to formal (e.g., with precisely defined execution semantics)
  - In safety critical systems formality is warranted.
    - Example: SAE Architecture Analysis & Design Language (AADL) Standard Suite (AS-5506 Series) - single annotated architecture model permits formal analysis across multiple quality attributes
- More tooling and automation is needed to satisfy assurance needs.

Software Development: Making Decisions



There is tension and a need for educated decisions.

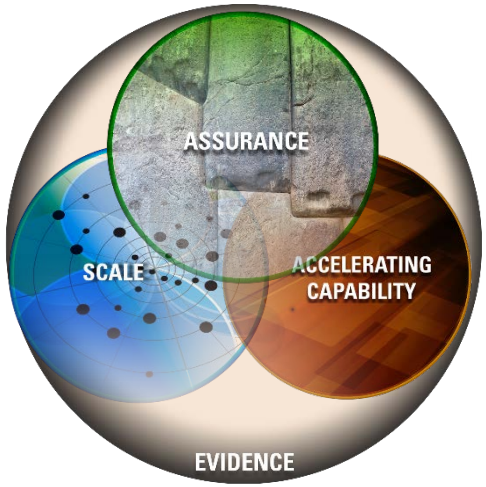
Architecture is the enabler for tradeoff analyses.

Conclusion

- Software architecture has strong foundations.
- Much progress has been made.
- Changes stemming from connectivity have precipitated a new world – new technologies, approaches, and challenges.
- Software architecture principles and their importance persist.
- But…much remains to be done.
- The future is in your hands.. be mindful.

Persisting Themes:

- **Modularity**
- **Commonality vs Variability**



thank you

This Is the Work of Many

At the SEI and CMU (past and present)

Len Bass
Joe Batman
Felix Bachmann
Mario Barbacci
Stephany Bellomo
Paul Clements
Peter Feiler
David Garlan
James Ivers
Rick Kazman
John Klein
Mark Klein
Philippe Kruchten
Grace Lewis
Ipek Ozkaya
Rod Nord
Mary Shaw
and many more…

And so many others in the professional community..

Contact Information

Linda Northrop
SEI Fellow
Telephone: 1+ 412-268-7638
Email: lmn@sei.cmu.edu
@LindaNorthrop

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

U.S. Mail:
Software Engineering Institute
Carnegie Mellon University
Pittsburgh, PA 15213-3890