



Modern Trends through an Architecture Lens

Linda Northrop

Software Engineering Institute
Carnegie Mellon University
Pittsburgh, PA 15213

Copyright 2018 Carnegie Mellon University. All Rights Reserved.

This material is based upon work funded and supported by the Department of Defense under Contract No. FA8702-15-D-0002 with Carnegie Mellon University for the operation of the Software Engineering Institute, a federally funded research and development center.

The view, opinions, and/or findings contained in this material are those of the author(s) and should not be construed as an official Government position, policy, or decision, unless designated by other documentation.

References herein to any specific commercial product, process, or service by trade name, trade mark, manufacturer, or otherwise, does not necessarily constitute or imply its endorsement, recommendation, or favoring by Carnegie Mellon University or its Software Engineering Institute.

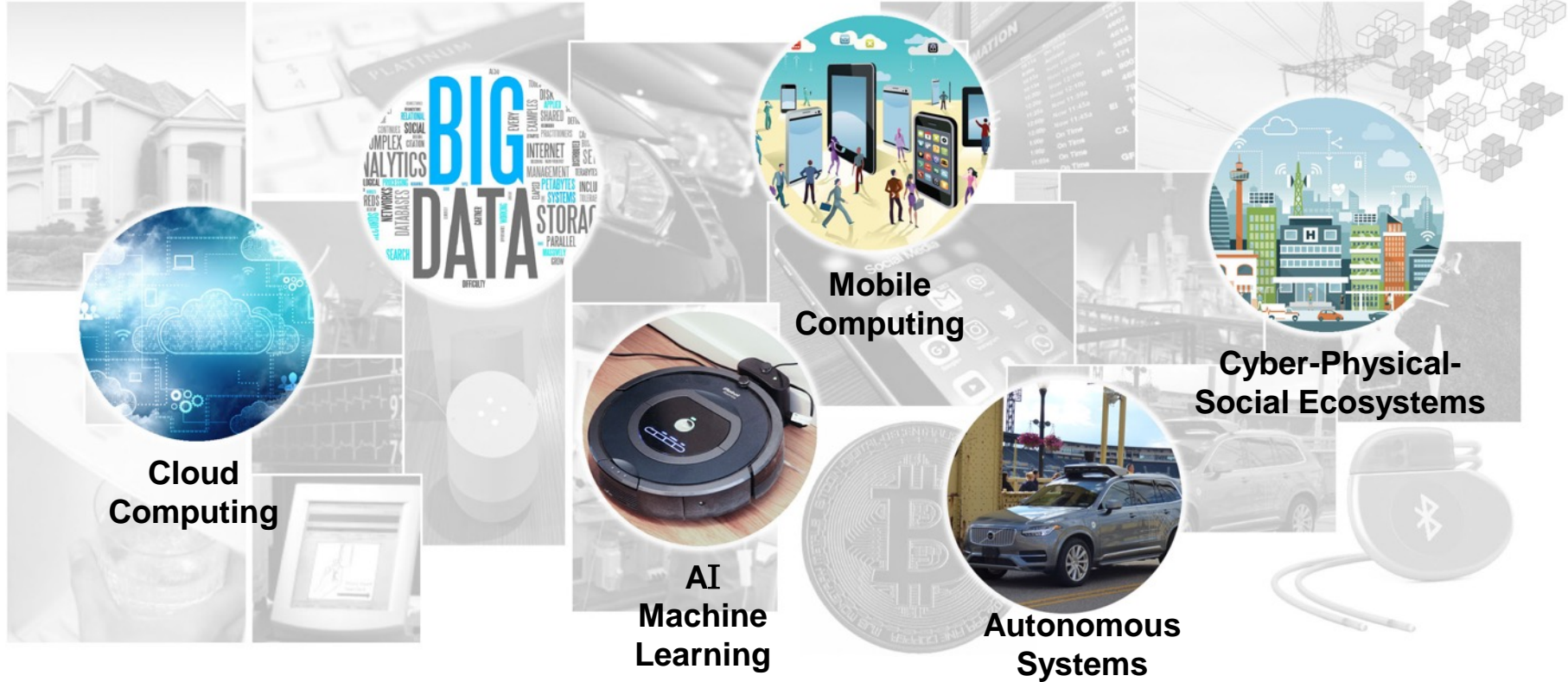
NO WARRANTY. THIS CARNEGIE MELLON UNIVERSITY AND SOFTWARE ENGINEERING INSTITUTE MATERIAL IS FURNISHED ON AN "AS-IS" BASIS. CARNEGIE MELLON UNIVERSITY MAKES NO WARRANTIES OF ANY KIND, EITHER EXPRESSED OR IMPLIED, AS TO ANY MATTER INCLUDING, BUT NOT LIMITED TO, WARRANTY OF FITNESS FOR PURPOSE OR MERCHANTABILITY, 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.

[DISTRIBUTION STATEMENT A] This material has been approved for public release and unlimited distribution. Please see Copyright notice for non-US Government use and 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.

DM18-0689

Modern Technology Trends



Software Development Trends



Agile approaches

DevOps

Scripting languages

Dashboards

Application frameworks

Distributed development environments

Open source libraries

Containers

Microservices

NoSQL



docker

GitHub



Downside



Liability in a Self-Driving Car Forces Tesla to Confront Its Limits

let expressed concerns that the Florida death could cast a pall over their field — and prompt federal regulators to craft to write more “stricter” rules on the technology “any of their own late to the” executives at the rival company to comment published.

Phillips, of Atlanta, that most car already tempered predicted a con- for the avail- ing able to be all, too.”

“While the investigation into the accident is just beginning, the repercussions could be profound for Tesla.”

The company has been preparing to sharply increase its manufacturing production for the introduction next year of its own expensive Model 3 sedan, which already has a long waiting list of customers who have preordered the vehicles. At the same time, Mr. Musk is trying to engineer the merger between Tesla and SolarCity and complete a new factory under construction in Nevada that will produce batteries meant to provide energy storage both for cars and homes.

In the past, Mr. Musk has been able to move quickly past issues related to Tesla’s quality and safety, such as when the company modified its cars to prevent fires from catching fire in high-

When the company would disclose more information about the accident, or about any plans for possibly alerting vehicle owners about the dangers of maintaining the Autopilot feature.

A spokesman for the National Highway Traffic Safety Administration, Bryan Thomas, declined to say why the agency waited until late June to begin a formal inquiry into an accident that happened in May, or why the agency did not require Tesla to notify owners about a possible problem.

“The investigation into the accident is just beginning, the repercussions could be profound for Tesla.”

The company has been preparing to sharply increase its manufacturing production for the introduction next year of its own expensive Model 3 sedan, which already has a long waiting list of customers who have preordered the vehicles. At the same time, Mr. Musk is trying to engineer the merger between Tesla and SolarCity and complete a new factory under construction in Nevada that will produce batteries meant to provide energy storage both for cars and homes.

In the past, Mr. Musk has been able to move quickly past issues related to Tesla’s quality and safety, such as when the company modified its cars to prevent fires from catching fire in high-

That type of response is not unusual for Mr. Musk, who tends to attack suggestions that Tesla has ever fallen short of its goals.

“This company is very personal to Elon Musk and he is very passionate and protective of it,” said Mr. Montoya of Edmunds.com. “But when anything goes wrong, he becomes very defensive.”

Regulators are now proceeding with the next step in their investigation by preparing to make an official “information request” for Tesla to provide more details on the accident and the self-driving equipment in the vehicle.

Mr. Thomas, the spokesman for the federal safety agency, declined to say how long the process might take.

Mr. Musk has recently been publicly drumming up support for Tesla’s proposed merger with SolarCity, while continuing to ride as Tesla’s chief shareholder.

On May 31, for example, he spent more than three hours discussing Tesla’s history and outlining its future before an overflow crowd of shareholders at the company’s annual meeting.

Over the course of the meeting, he lashed extensively about setting new industry standards for manufacturing at Tesla’s California plant, and building the world’s biggest battery factory in the Nevada desert.

When asked by one Tesla owner about whether he expected to keep adding new technology to the Model S, Mr. Musk responded as if that was a foregone conclusion. “We are always going to keep improving the product,” he said.

But what he did not mention at all was the Florida accident, which had occurred three weeks earlier.

A carmaker remains vague about a crash that occurred nearly two months ago.



Mired in Legacy Systems



Today's Software Workforce

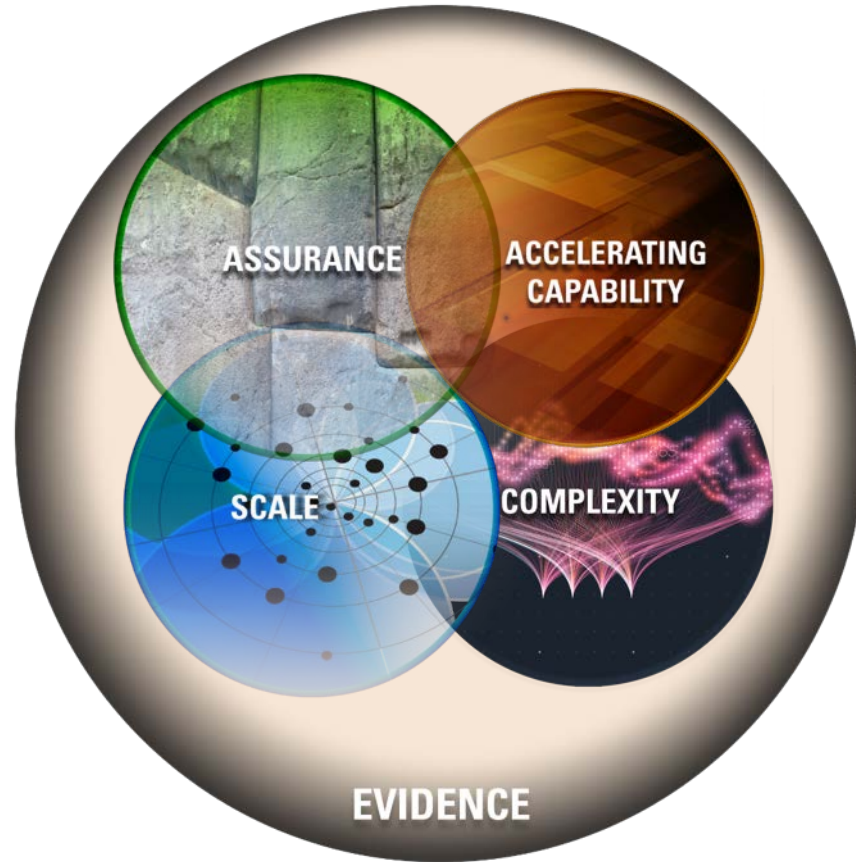


Different Abilities Have Diverse Needs

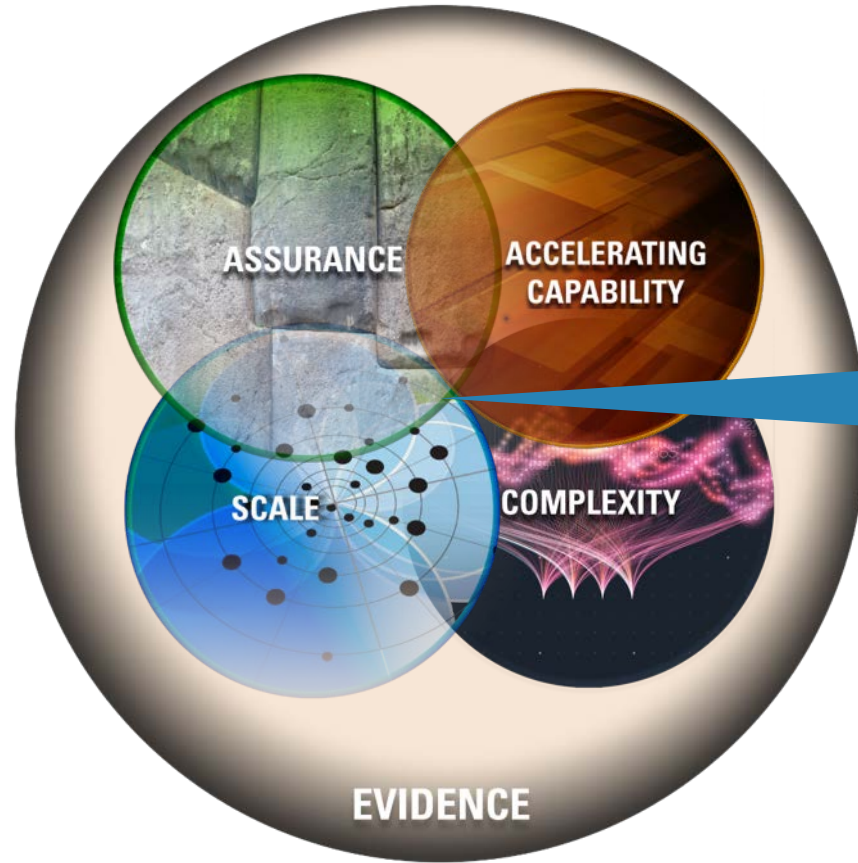


<https://www.wpsbc.org/>

Software Engineering Challenges



The Intersection and Architecture



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

Architecture is the enabler for tradeoff analyses.

Software Architecture

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



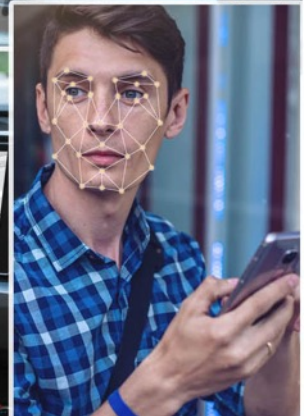
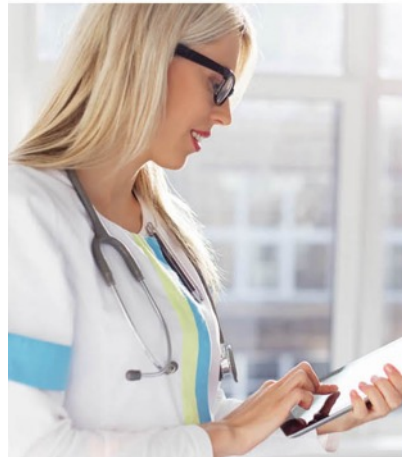
Quality Attributes

Quality attributes

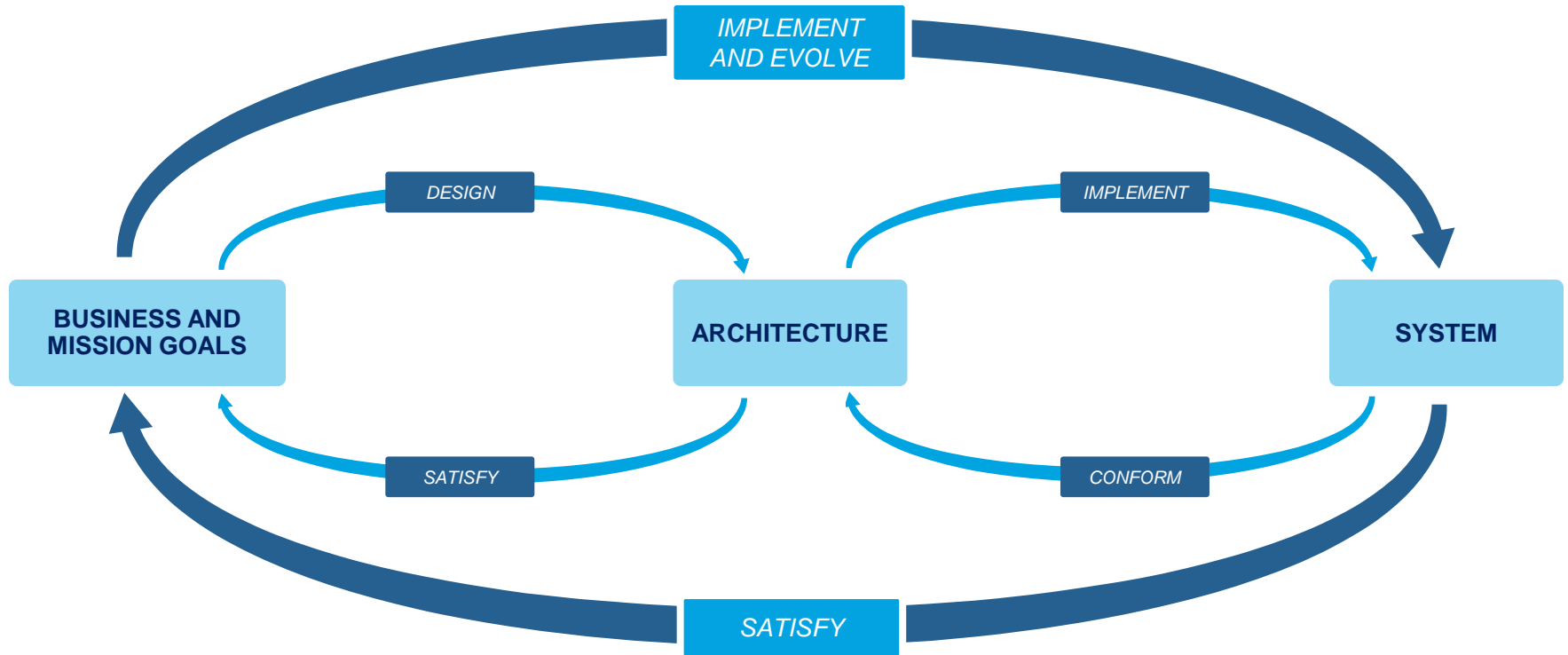
- properties of work products or goods by which stakeholders judge their quality
- stem from business and mission goals.
- need to be characterized in a system-specific way

Quality attributes include

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



Central Role of Architecture



Architectural 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

Microservices

Persisting Themes

- Modularity
- Commonality vs Variability

What Now?

Is structure still important?

Are old architectural principles still relevant?

What are the architectural drivers in today's systems?

What kind of new architectural styles are needed to be able to ensure intended behavior?

What kinds of design paradigms will help us ensure the safety, security, and reliability of systems with artificial intelligence and autonomy?

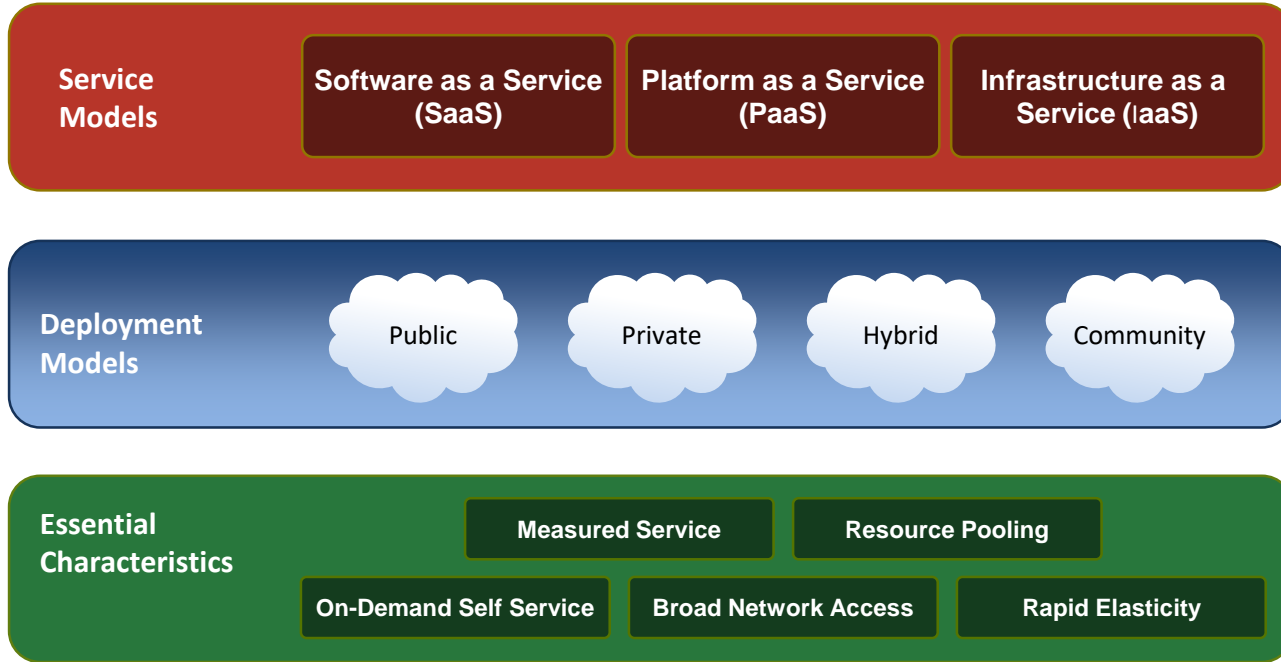
Should such systems be tested with different approaches?

Can the design of these systems be engineered to ensure their testability?

How can the design of software assist in ensuring its intended ethical use?

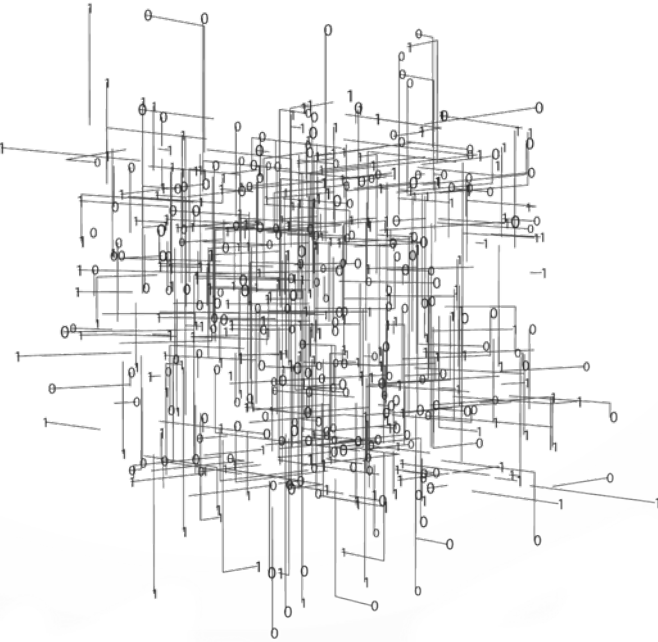


Cloud Computing Models and Essential Characteristics



Source: National Institute of Standards and Technology (NIST), 2011

Architecture Perspective



Shared responsibility

Two potentially different sets of business goals and quality attributes

- SLA is an architectural decision
- Portability tradeoffs
- Tempo differences
- Runtime tradeoffs

Architectural tradeoffs involve cost

Testing challenges that require architectural support

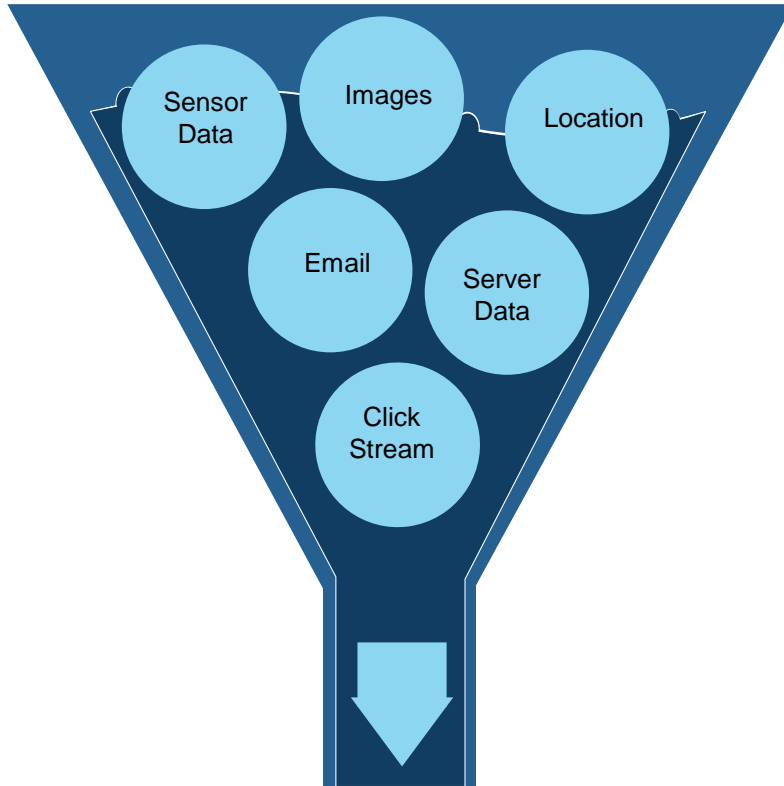
- Controllability
- Observability

Mobile Device Trends



Today's UI is increasingly mobile.

Big Data Systems



Involves

- Data analytics
- Infrastructure

Analytics is typically a massive data reduction exercise – “data to decisions.”

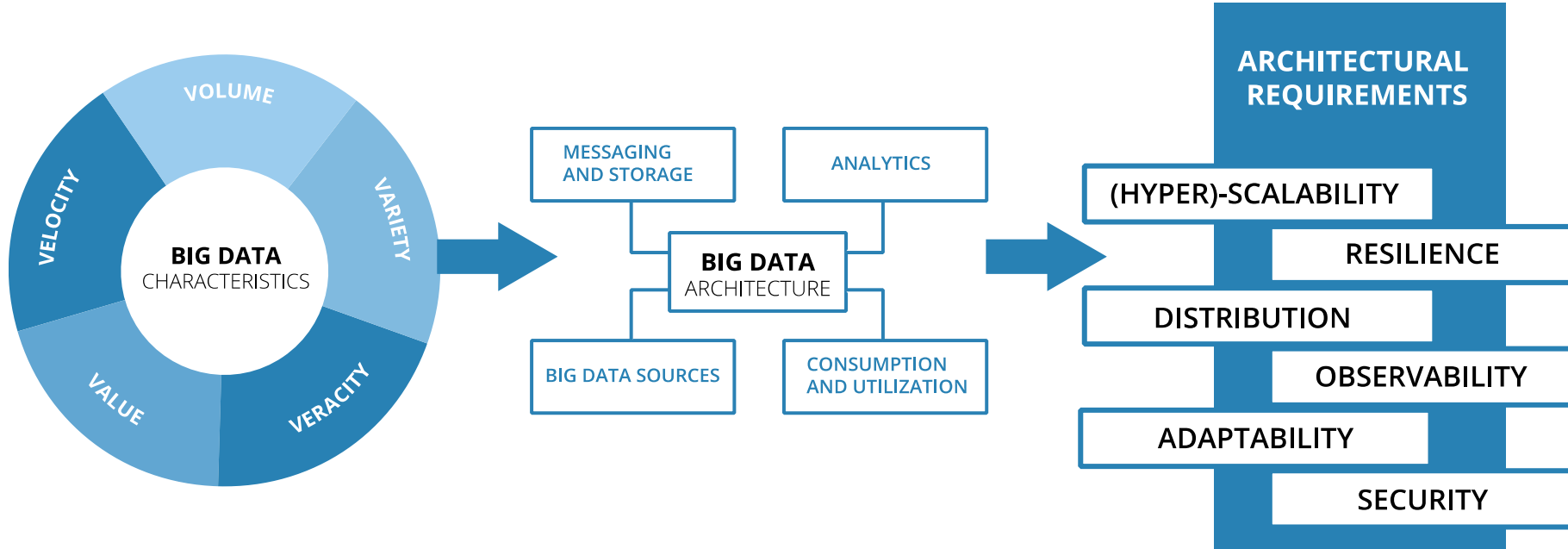
Computation infrastructure necessary to ensure the analytics are

- fast
- scalable
- secure
- easy to use

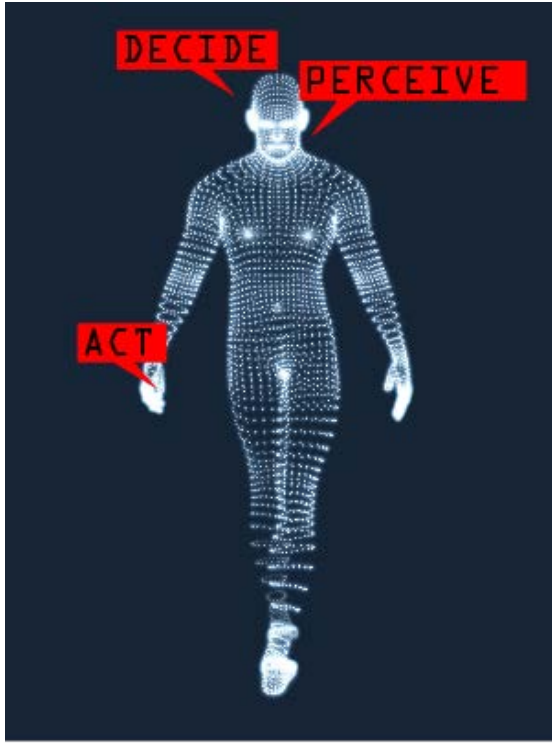
Big Data State of the Practice: from Pioneers to Diverse Industries



Big Data Software Architecture



Artificial Intelligence (AI)



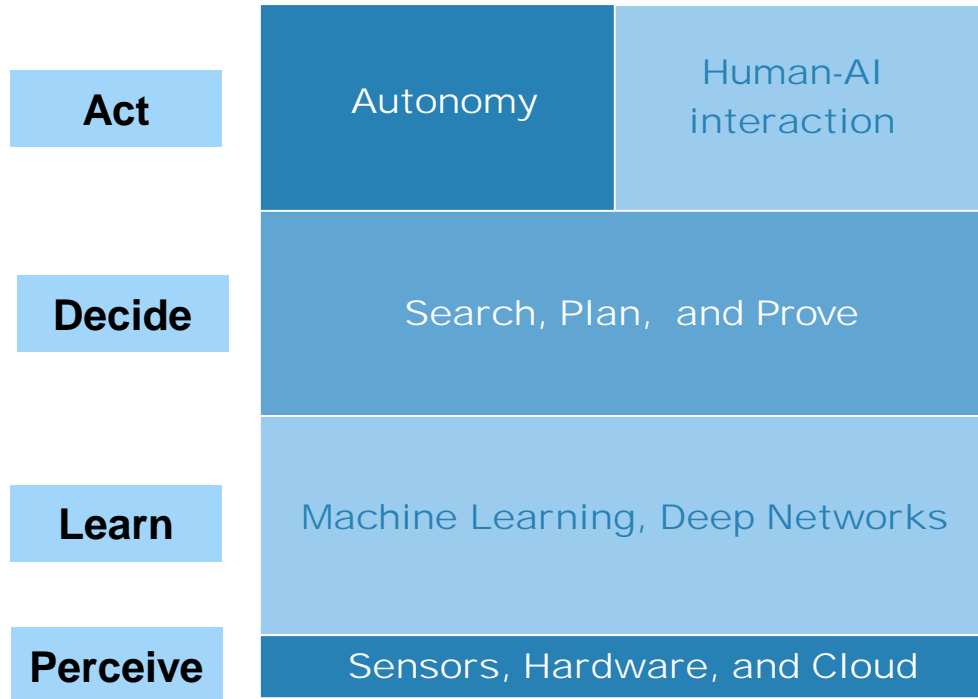
AI is creating a revolution in system capability.

- Data analytics
- Cooperative autonomous systems
- UX/collaboration modalities
- Cyber autonomy and counter-autonomy
- Bug repair, self-healing, and self-adaptive systems

There are also reasonable fears.

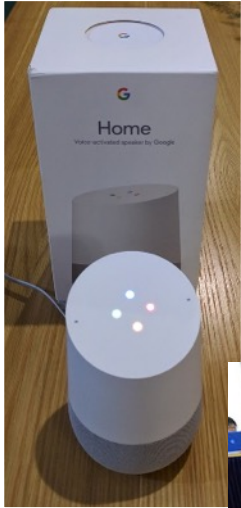
“A Vision for Software Development,” Andrew Moore, Carnegie Mellon University, Jan 6, 2018

AI Stack



“A Vision for Software Development,” Andrew Moore, Carnegie Mellon University, Jan 6, 2018

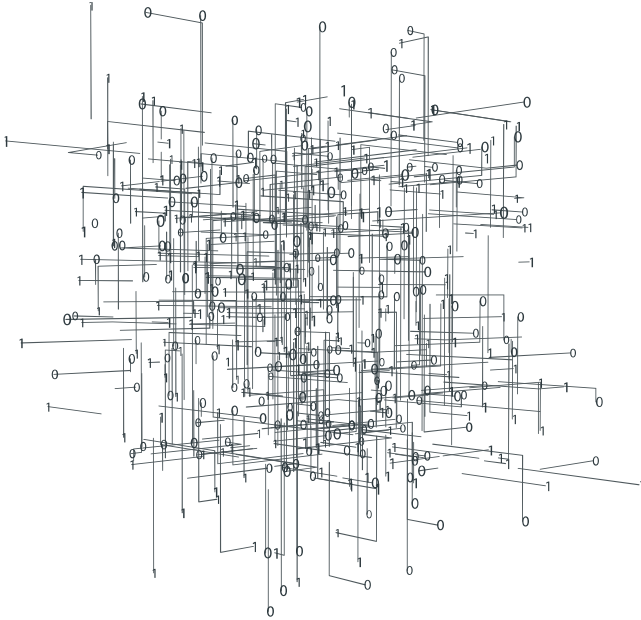
Machine Learning (ML) Systems Today



Machine learning: learning to predict by extrapolating from data

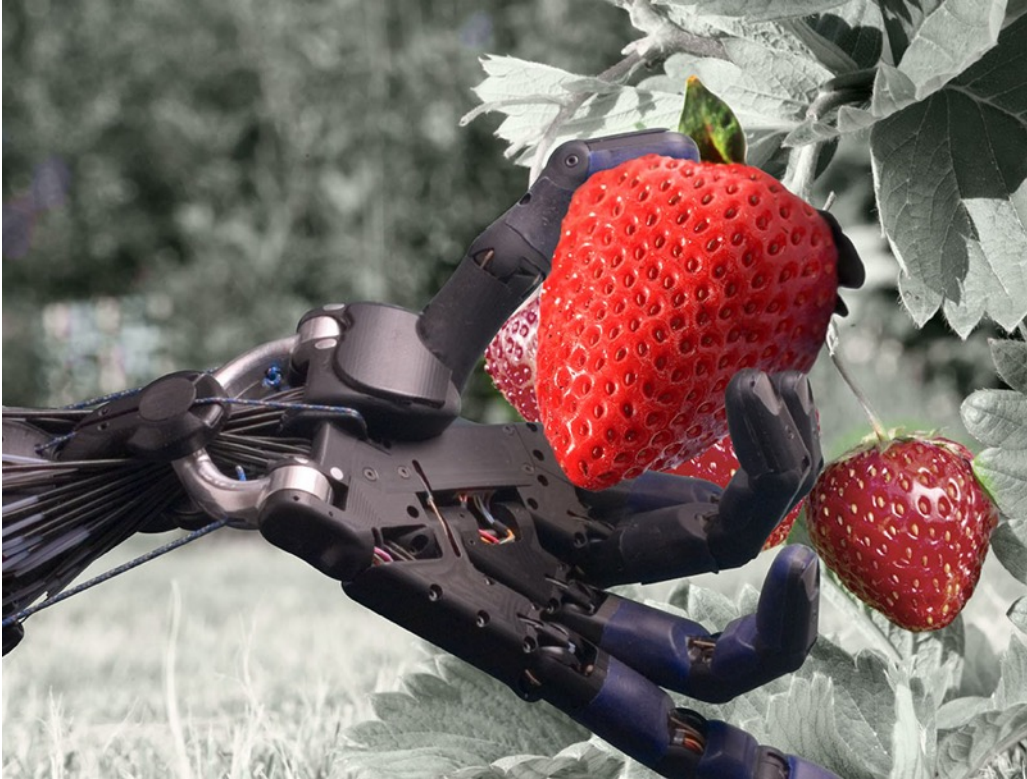
- Can provide rapid response to dramatically changing contexts
- Effectiveness is highly variable across different domains
- Algorithms are readily accessible
- Overall, today still a cottage industry

ML, Software Engineering, and Architecture

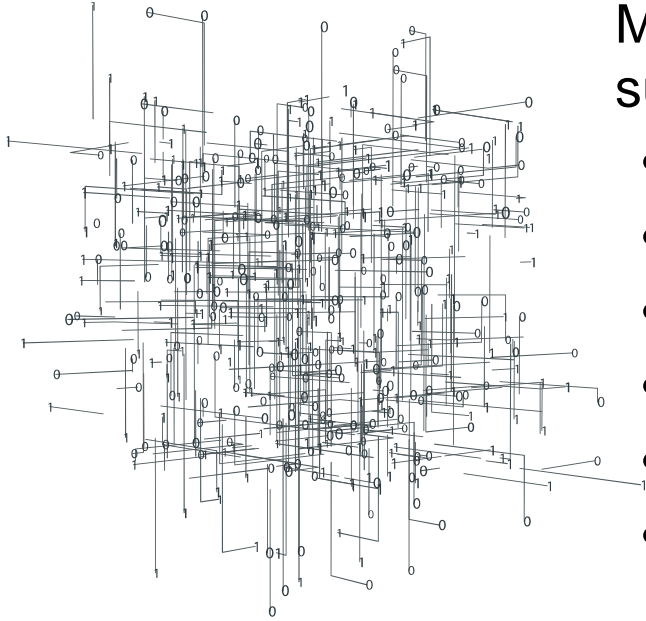


1. Correctness will not be possible.
 - ML has an experimental mindset.
2. Holistic testing is impossible
 - uncertainty and error will be part of the output
3. Deductive reasoning from the code and metadata is not and will not be effective.
4. Data and its attributes must be first class.
5. Divide and conquer doesn't work.
6. Quality attribute focus
 - reliability
 - observability

Autonomous Systems



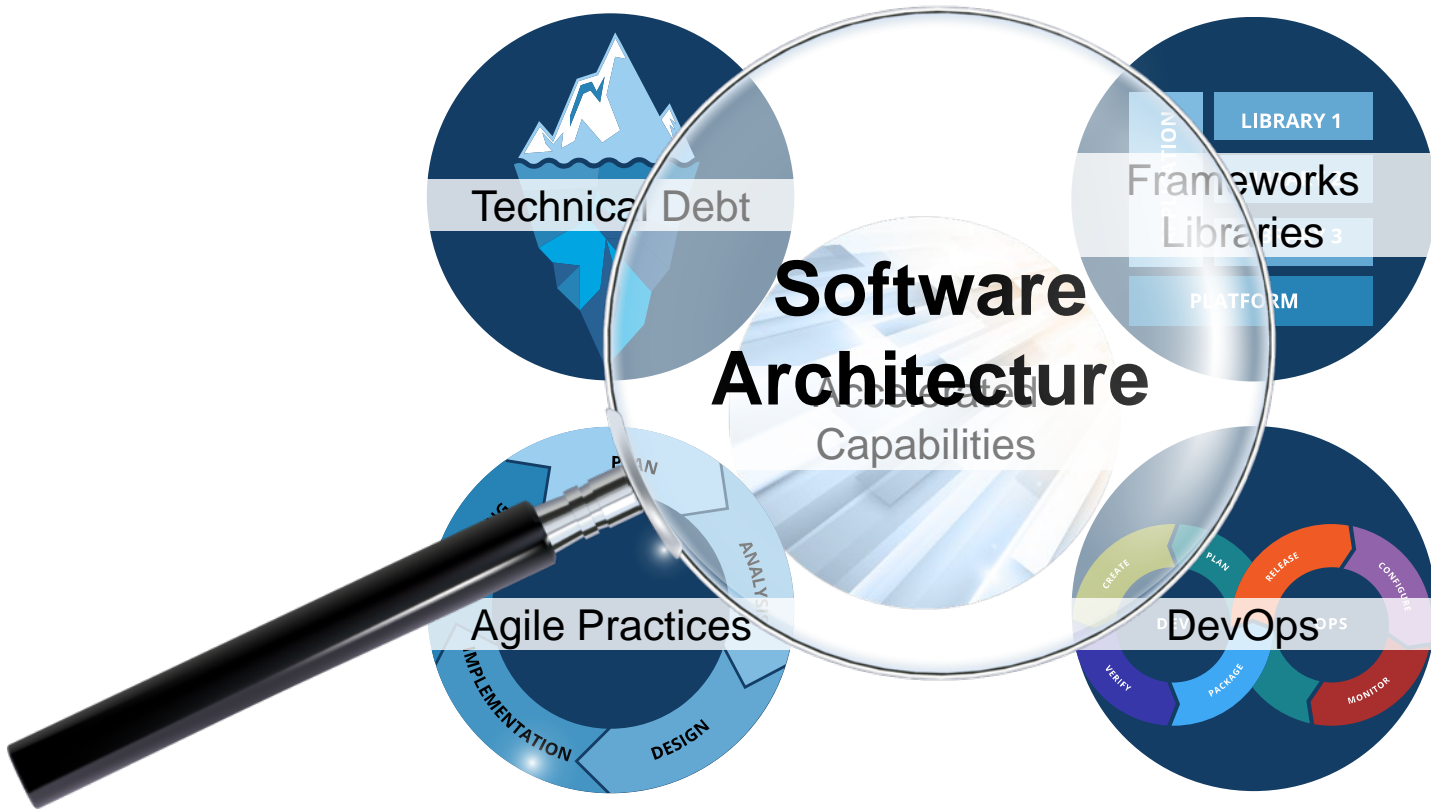
Autonomous Systems, Software Engineering, and Architecture



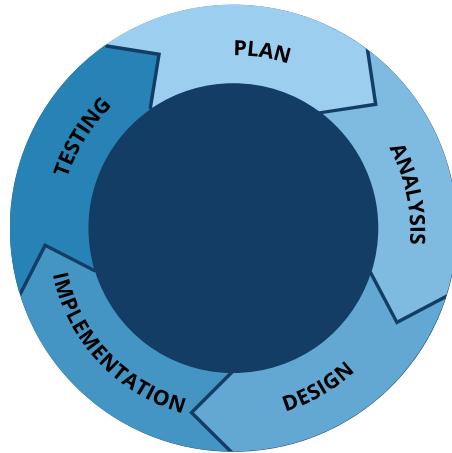
ML issues plus structural support for

- Human/Machine collaboration
- Safety
- Timing
- Security
- Adaptation
- Edge case handling





Incremental Development and Architecture



Architecture design can be done incrementally.

There is a difference between being agile and doing agile.

Agility is enabled by architecture – not stifled by it.

Architecture has a role to play in supporting agile at scale.

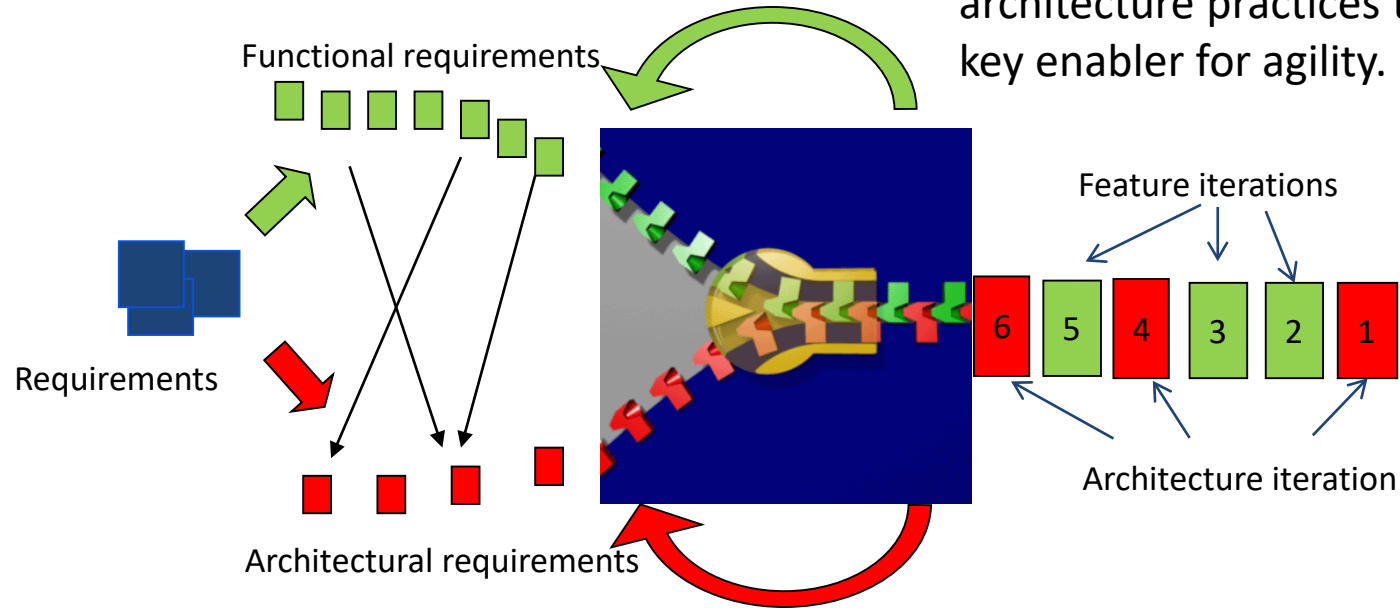
How much architecture design is enough?

The need is for modular evolution.

Automation can be key to acceleration.

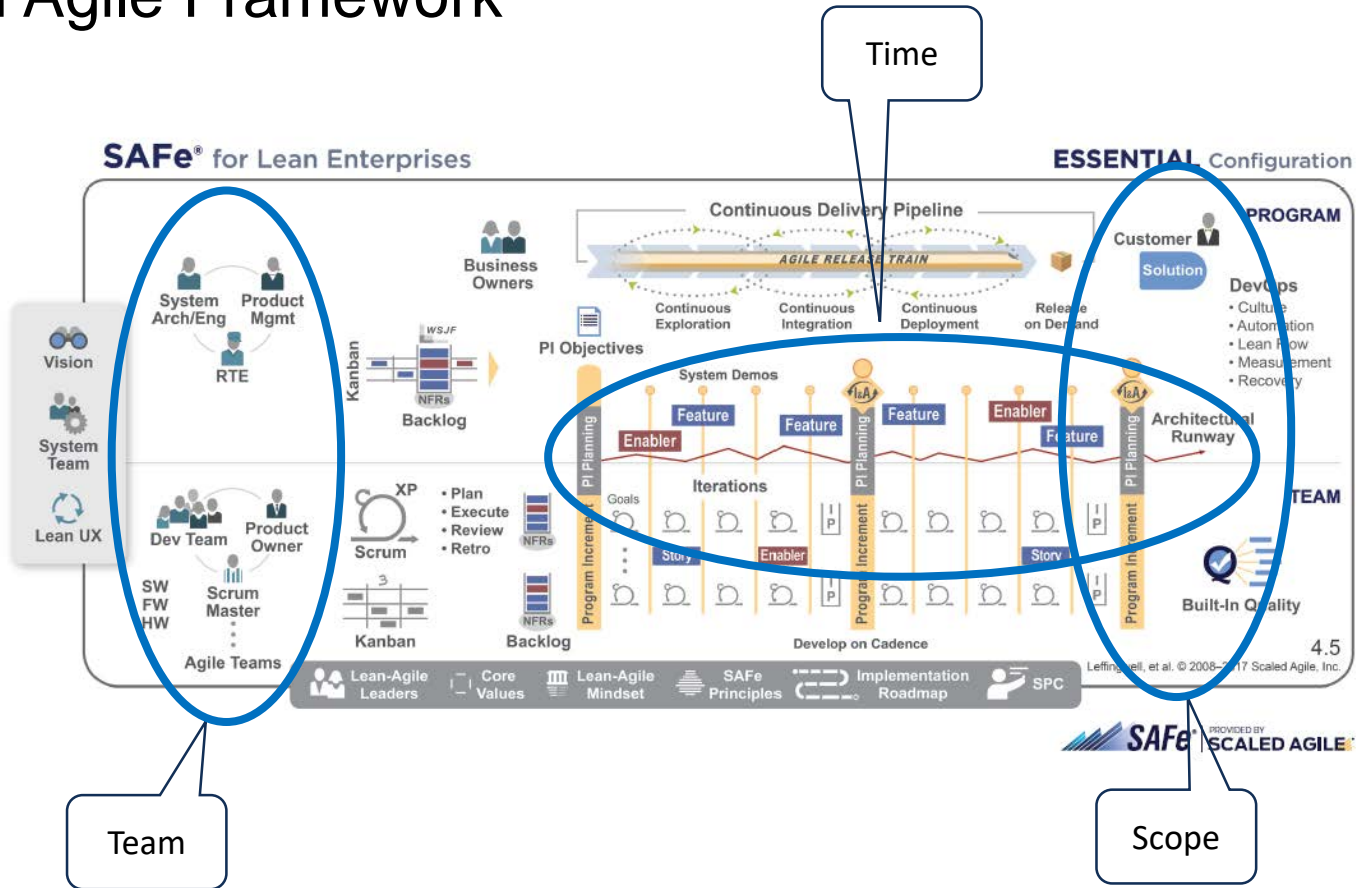
Integrated Agile/Architecture Practices

Successful project teams find architecture practices to be a key enabler for agility.



Nord, R.L., Ozkaya, I. and Kruchten, P. Agile in Distress: Architecture to the Rescue. T. Dingsøyr et al. (Eds.): *XP 2014 Workshops*, LNBIP 199, pp. 43–57, 2014. Springer International Publishing Switzerland 2014
“A Study of Enabling Factors for Rapid Fielding: Combined Practices to Balance Speed and Stability,” by Bellomo, Nord, and Ozkaya. ICSE 2013.

Scaled Agile Framework

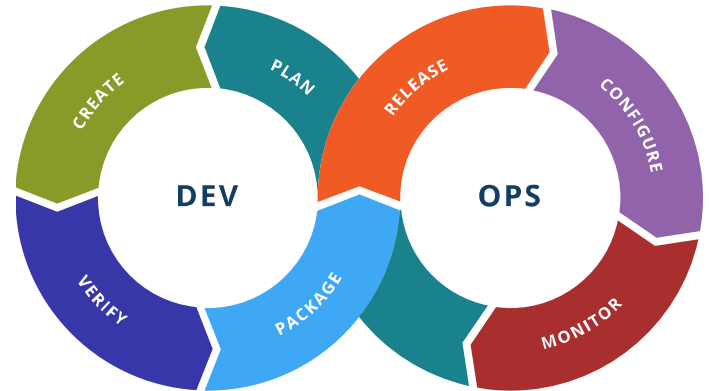


Available at <http://v4.scaledagileframework.com/>

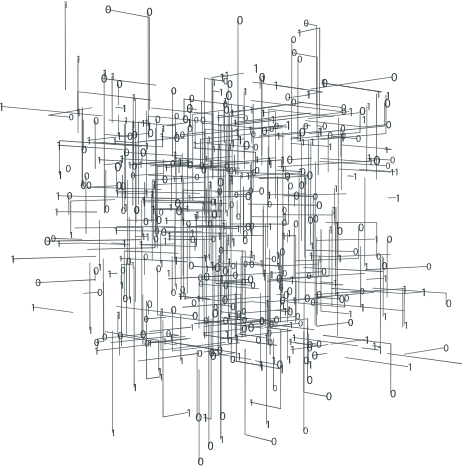
DevOps

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



DevOps and Architecture



Architect the system for deployability.

Architect the tool chain.

- Integrate security into DevOps.

Architect the IaC.

Implement the architecture you design.

- Write custom checks for implementation conformance.
- Automate tests for quality attributes.
- Collect data to monitor health of the architecture.



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

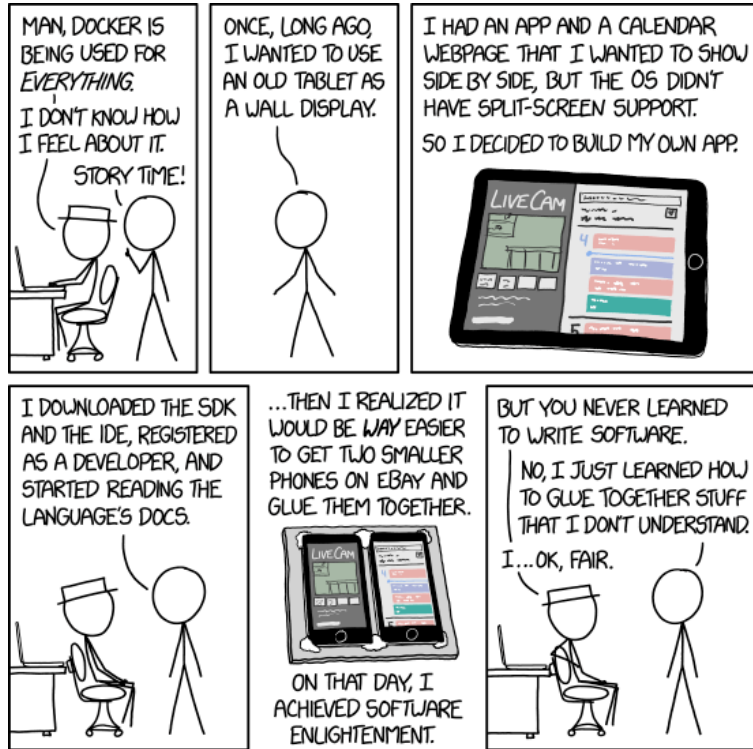
Deployability Tactics

DevOps Tactics

Availability	Modifiability	Performance	Testability
Monitor	Encapsulate	Increase Resources	Sandbox
Exception Detection	Defer Binding	Increase Currency	Specialized Interfaces
Exception Handling	Abstract Common Services	Schedule Resources	Record/Playback
Voting		Reduce Overhead	
Rollback		Maintain Multiple Copies of Computations	
Active Redundancy		Maintain Multiple Copies of Data	
Reconfiguration		Limit Event Response	
		Prioritize Events	
		Manage Sampling Rate	

Bellomo, S., Kazman, R., Ernst, N., Nord, R.: Toward Design Decisions to Enable Deployability: Empirical Study of Three Projects Reaching for the Continuous-Delivery Holy Grail. In: *First International Workshop on Dependability and Security of System Operation*, pp. 32–37. IEEE Press, New York (2014)

Frameworks, Libraries, Containers, etc.



Containers | xkcd.com

Reuse abounds.

Rapid construction through assembly.

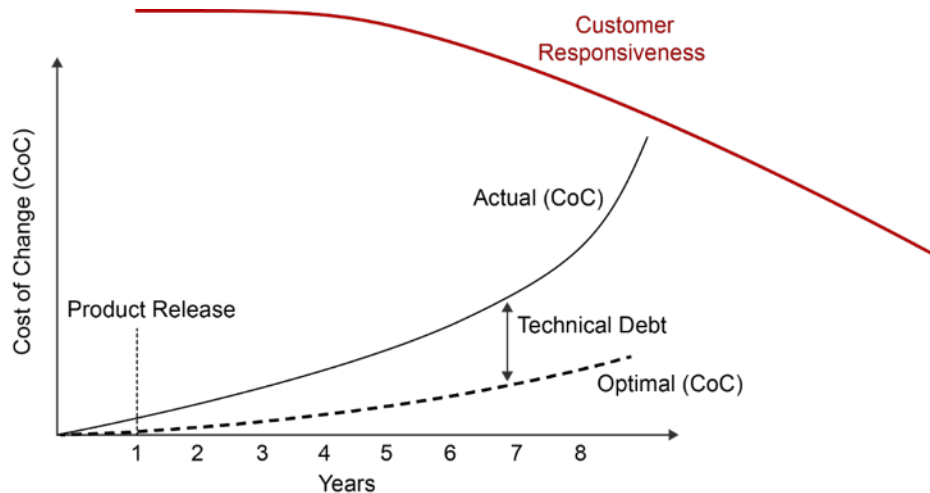
Architecture is implicit.

Undesirable behavior can occur.

Debilitating technical debt can occur.

Technical Debt*

Technical debt* is a collection of design or implementation choices that are expedient in the short term, but that can make future changes more costly or impossible.



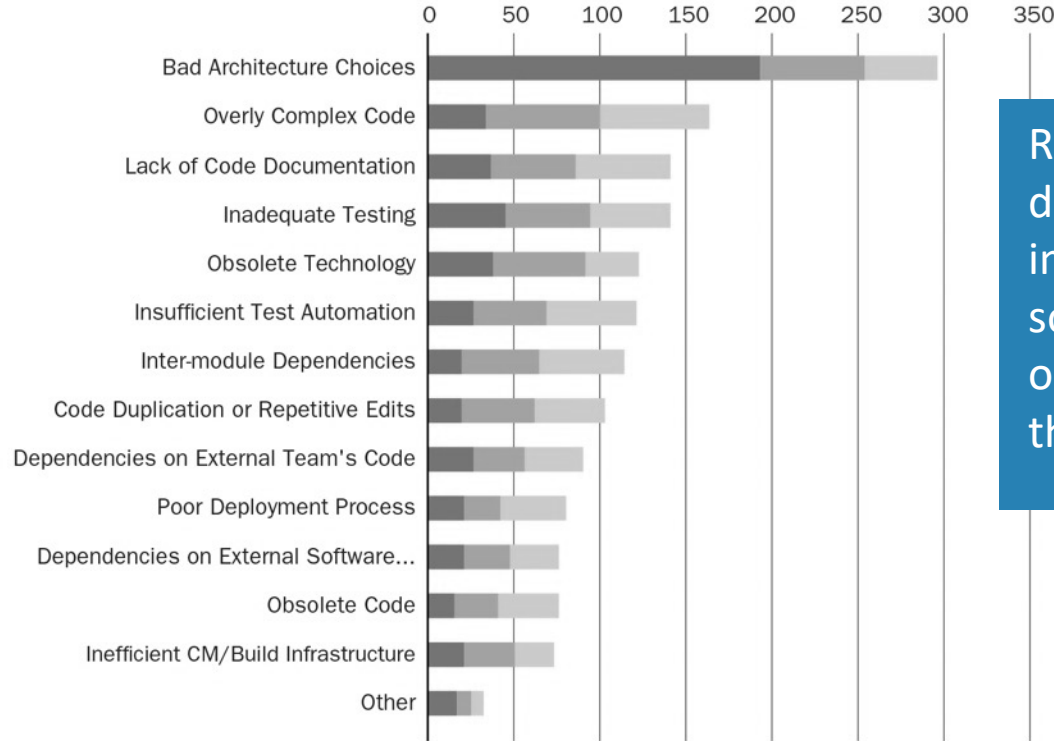
Exists in

- Code
- Build scripts
- Data model
- Automated test suites
- Structural decisions

- 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/>

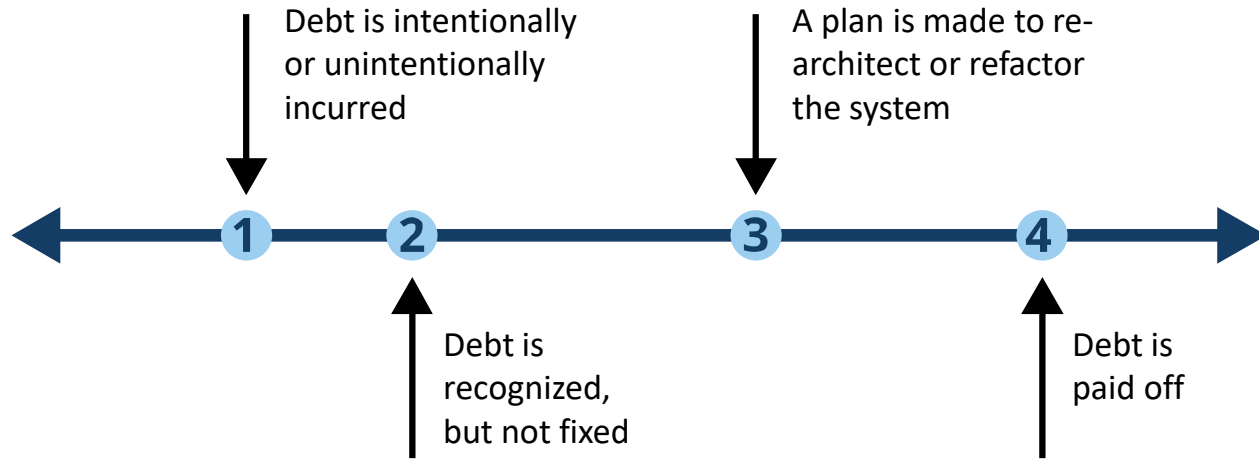
Software Architecture and Design Tradeoffs Matter



Results from over 1800 developers from two large industry and one government software development organization list architecture as the most costly technical debt.

“Measure it? Manage it? Ignore it? Software Practitioners and Technical Debt” N. Ernst, S. Bellomo, I. Ozkaya, R. Nord, I. Gorton, Int. Symp on Foundations of Software Engineering 2015

Technical Debt Timeline



All systems have technical debt.
The impact depends on how you manage it.

Architecture and Assurance



Security concerns are paramount.

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

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

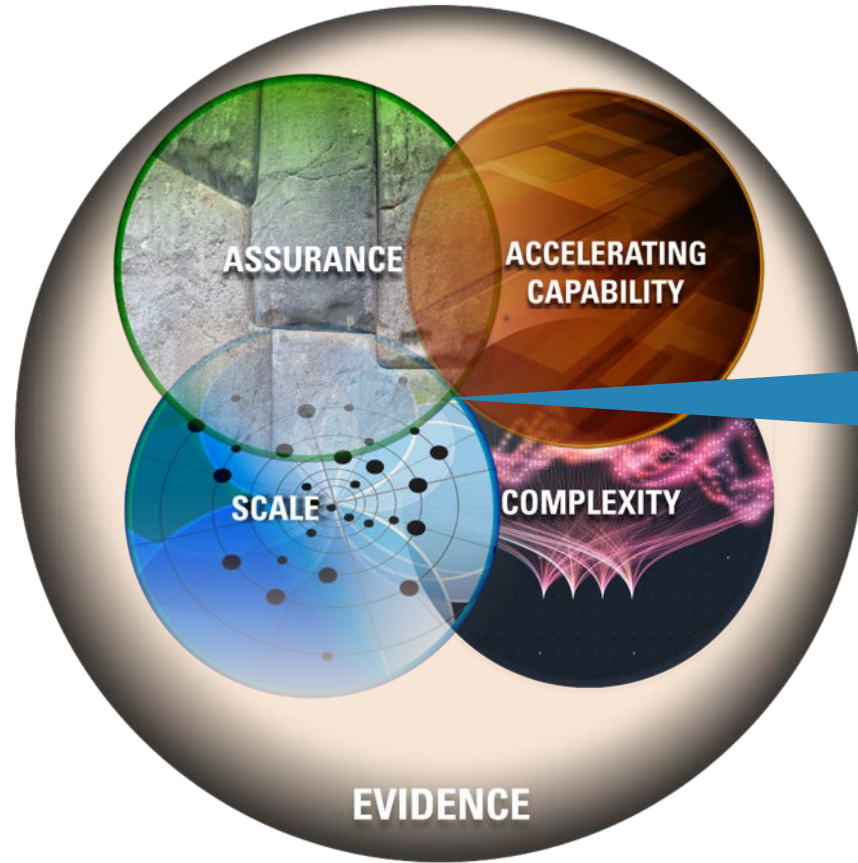
Tool chains that generate code, configuration files, etc. introduce unknowns.

Autonomy, machine learning, and connected physical systems introduce unknowns.

Humans in the system introduce unknowns.

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

So Where Are We?



There are tradeoffs, tension, and needs for educated decisions and measurement.

Architecture is still the enabler for tradeoff analyses, but there is a changed architectural workforce and new architectural needs.

Today's Software Architecture Workforce

Differs widely by organization and domain

Reveals a democratization of the architecture

Has a spectrum of experience

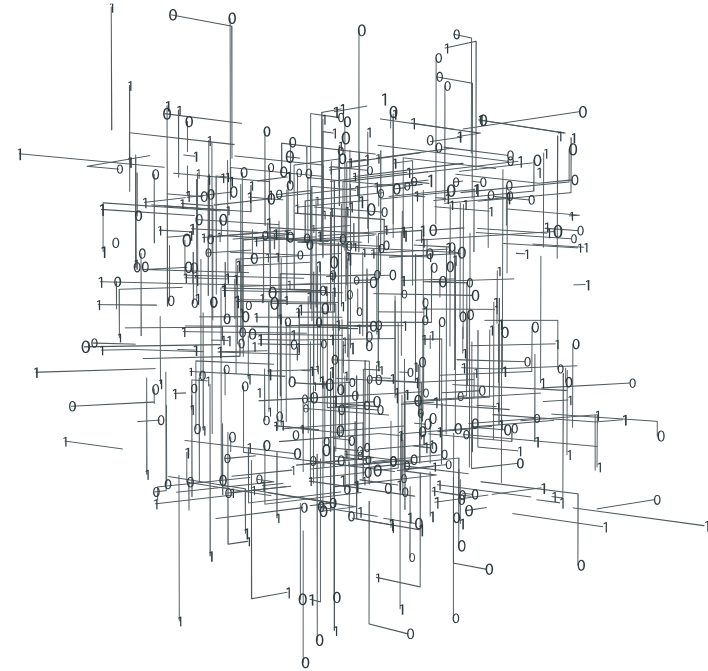
Tension between wisdom of the crowd and experience

And yet...

- How do quality attributes get distributed across team(s)?
- Technical debt is accruing due to lack of architectural thinking.
- More design horse power (not less) needed for complex systems and specialized domains.
- More talent (not less) needed, some from other disciplines.



Net Sum Architectural Needs



Tradeoffs, decisions, structure persists.

Security needs are heightened.

Different quality attributes at the fore.

New focus on

- Evolution
- Runtime
- Data
- ML
- Automation

Evolution and Runtime

Evolution

- Explicitly design for continuous evolvability and adaptability in order to deal with uncertainty and not incur prohibitive technical debt
- Decisions will reflect changing principles, policies, and algorithms

Runtime

- Architecture needs to be seen at runtime
- Observability: mechanisms to support continuous monitoring
- Recovery, auto-scaling, managed roll-out
- Dynamic adaptation to support environmental changes and tradeoff priorities
- Configuration changes at runtime without performance hits
- Human-in-the-system models
- Situational awareness and explanation

Data and ML

Data

- Data and its attributes must be first class citizens
- Relax current design heuristics; e.g., how to decouple components and data
- Software analysis tools will need to reason about data

ML

- Certainty will give way to probability
- Ability to articulate the tradeoffs in ML
- Criteria for whether ML is a good solution for a given problem
- Architecture patterns that allow post mortem of ML systems

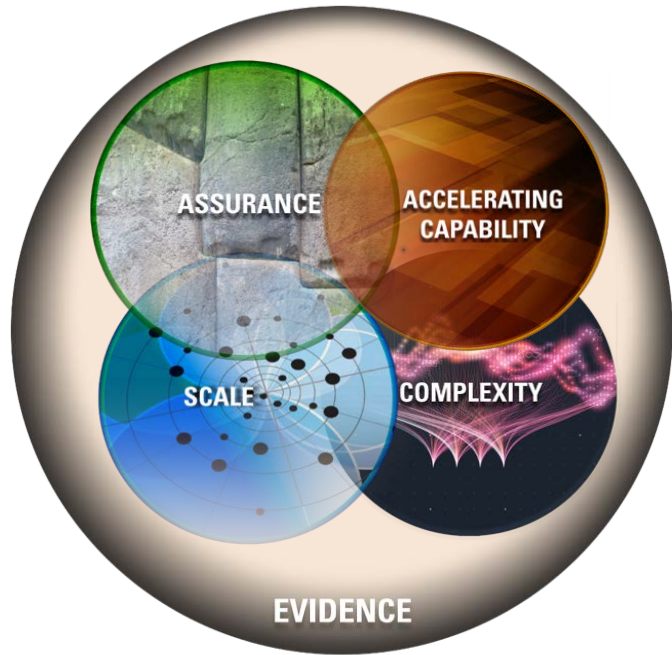
Automation

Tools to support design and architecture

- At design time for discovery, envisioning, and collaboration
- At run time for observation and environmental monitoring
- To embed design alternatives with code as part of the build system
- To detect and manage technical debt
- Move from explicit decisions to principles with guide rails
 - guide rails are manifest in the code
 - “smart” frameworks; architecture hoisting

ML to collaborate with designers and to understand the impact of design decisions

Conclusion



Structural decisions continue to be made.
Tradeoffs continue to be made.
Software architecture importance persists.

But...

- The focus must fit today's environment and needs.
- Architects need to embrace uncertainty.
- New tooling is essential.

thank you

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

