Carnegie Mellon University
Software Engineering Institute

Modern Trends through an Architecture Lens

Linda Northrop

Software Engineering Institute
Carnegie Mellon University
Pittsburgh, PA 15213

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.

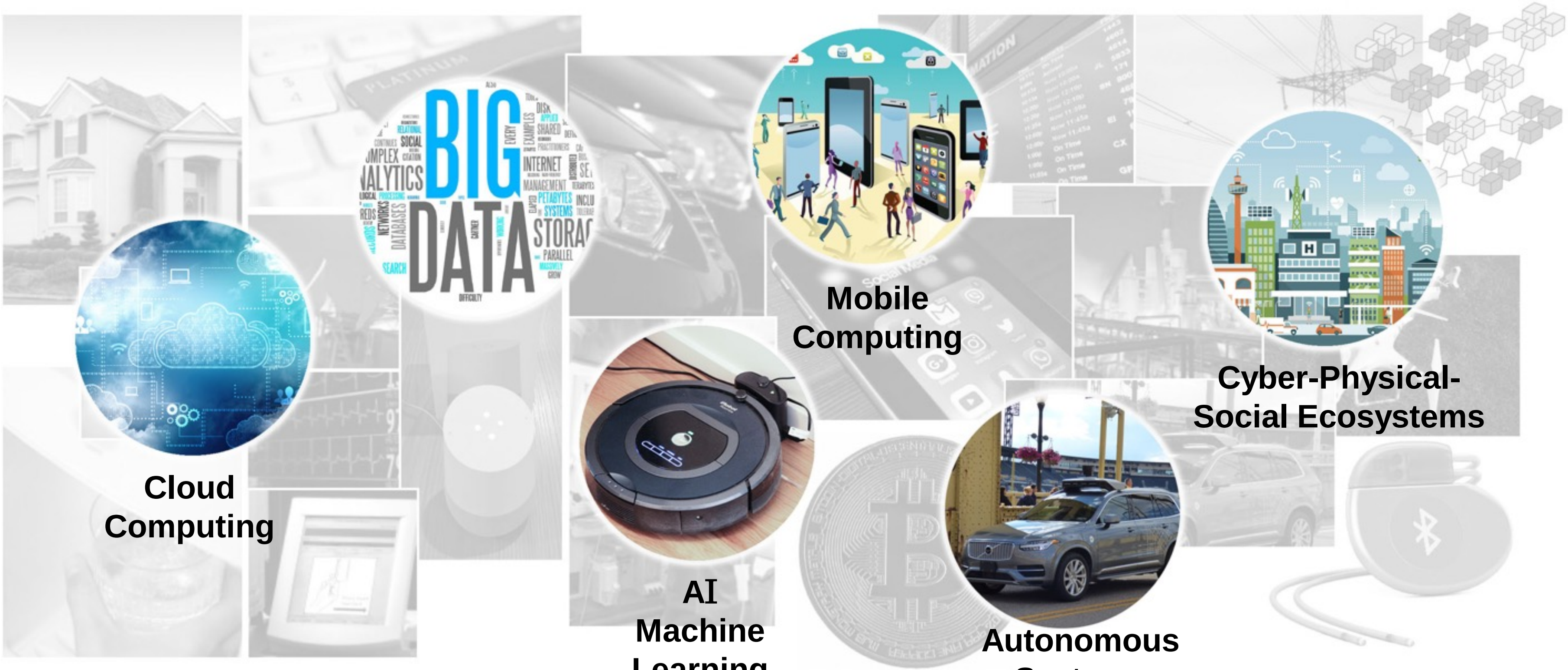
References herein to any specific commercial product, process, or service by trade name, trade mark, manufacturer, or otherwise, does not necessarily constitute or imply its endorsement, recommendation, or favoring by Carnegie Mellon University or its Software Engineering Institute.

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.

DM18-0689

Society’s Dependence on Software

Modern Technology Trends



Cloud Computing

Mobile Computing

AI Machine Learning

Cyber-Physical-Social Ecosystems

Autonomous Systems

Software Development Trends

AGILE
SOFTWARE DEVELOPMENT
MANIFESTO
INDIVIDUAL AND INTERACTIONS
OVER PROCESSES AND TOOLS
WORKING SOFTWARE
OVER COMPREHENSIVE DOCUMENTS
CUSTOMER COLLABORATION
OVER CONTRACT NEGOTIATION
RESPONDING TO CHANGE
OVER FOLLOWING A PLAN

FREE CULTURE
HOW BIG MEDIA USES TECHNOLOGY AND THE LAW TO LOCK DOWN CULTURE AND CONTROL CREATIVITY
LAWRENCE LESSIG

Clustrix

Jenkins

mongoDB

Agile approaches

DevOps

Scripting languages

Dashboards

Application frameworks

Distributed development environments

Open source libraries

Containers

Microservices

NoSQL

docker

GitHub

hadoop

snowflake

ANGULAR by Google

APACHE Spark™

Downside

RANGE ROVER
SPORT
CYBERCRIME
VIRUS
THEFT
TECHNOLOGY ADDICTION
COMPUTER
IDENTITY
SNOOPING
CYBERATTACK
IDENTITY
CYBERWARFARE
Snooping
IDENTITY THEFT
ATTACK
TECHNOLOGY ADDICTION
VIRUS
CYBERWARFARE
CYBERCRIME
social dysfunction
SNOOPING
CYBERATTACK
COMPUTER VIRUS
ATTACK
COMPUTER
THEFT

B6 THE NEW YORK TIMES, SATURDAY, JULY 2, 2016

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

ket expressed concerns that the Florida death could cast a pall over their field — and prompt federal regulators to rush to write re-
ictive rules on the technology
e any of their cars take to the
Executives at the rival com-
eclined to comment pub-

Phillippi, of Aa-
ed that most car
already tempered
predicted a con-
le for the avail-
g vehicles.
raditional au-
say we are
ing able to
he car will
may be
off, too."
tion un-
d to re-
about
why it
acci-
nths
tors
re-
bout

when the company would disclose more information about the accident, or about any plans for possibly alerting vehicle owners about the dangers of misusing 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.

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 less 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 households.

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 batteries from catching fire in high-speed collisions.

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

But the company has been vague about whether the Florida accident was the result of a defect in the self-driving system, or the result of a driver who neglected to use it correctly.

For now, Tesla continues to characterize the system as a safety improvement.

"When used in conjunction with driver oversight, the data is unequivocal that Autopilot reduces driver workload and results in a statistically significant improvement in safety when compared to purely manual driving," the company said in a news release discussing the Florida crash.

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 his role as Tesla's chief cheerleader.

On May 31, for example, he spent more than three hours dissecting 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 talked 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.

usiast

imits

er, called attention to the
o on Twitter, it went viral.
Mr. Brown seemed to be elated.
"He had said, 'For something to

Mired in Legacy Systems

Today’s Software Workforce

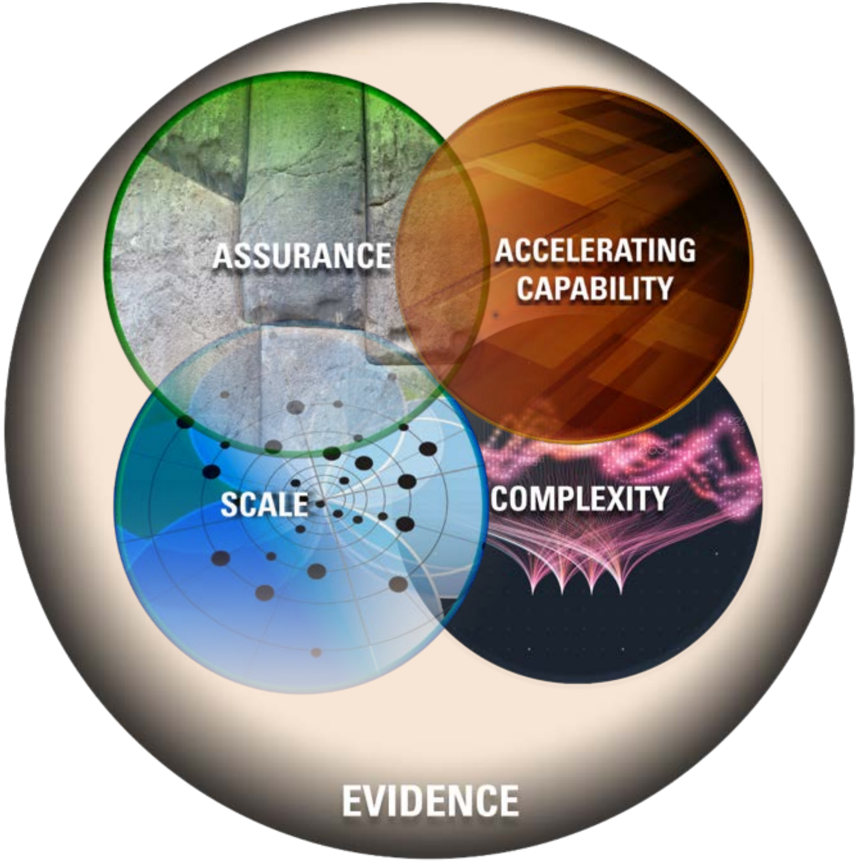
Holly Peterson

Jacqueline Cowan

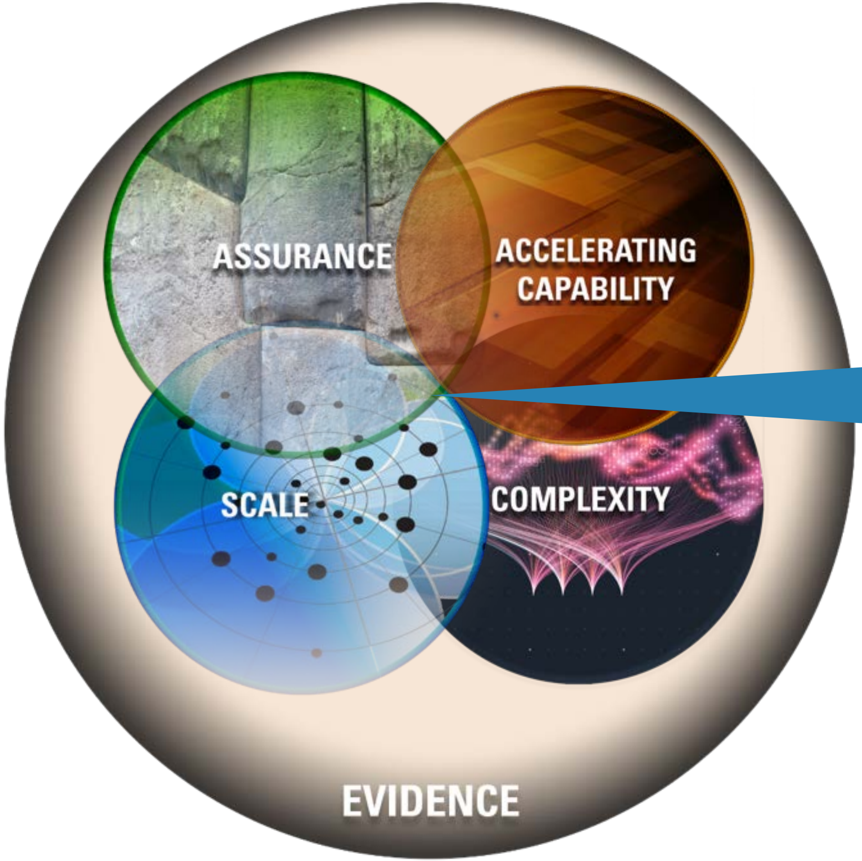
Different Abilities Have Diverse Needs

https://www.wpsbc.org/

Software Engineering Challenges



The Intersection and Architecture

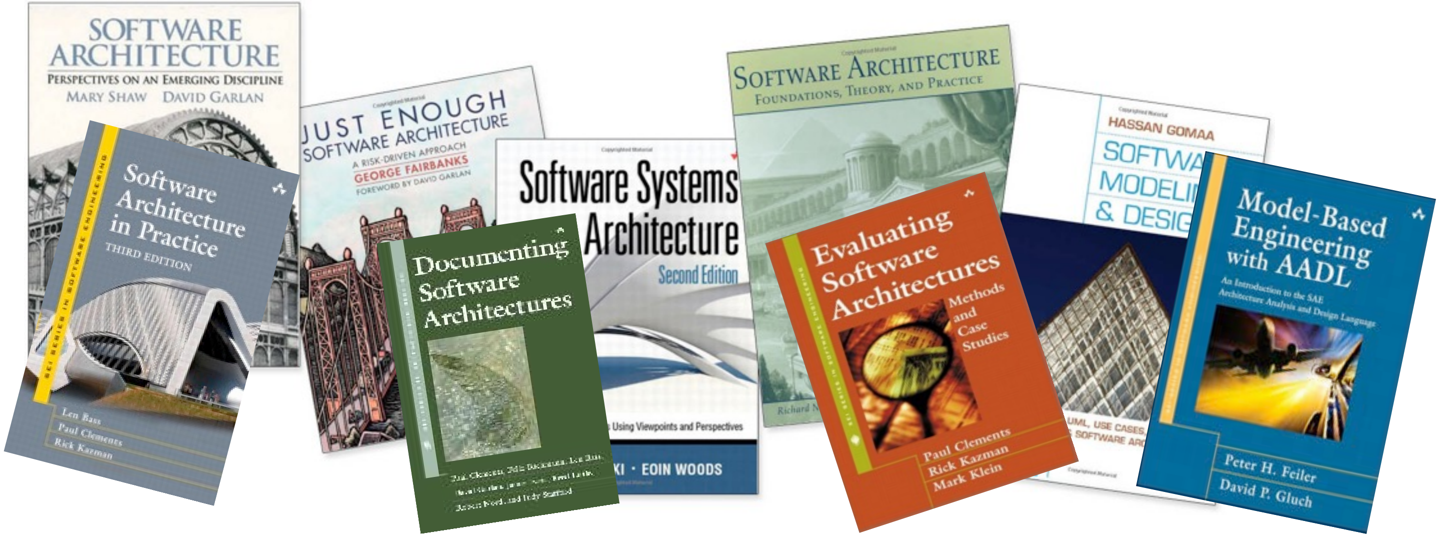


At the intersections there are difficult tradeoffs to be made - in structure, technology, process, and cost.

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


SOFTWARE ARCHITECTURE
PERSPECTIVES ON AN EMERGING DISCIPLINE
MARY SHAW DAVID GARLAN
Software Architecture in Practice
THIRD EDITION
SEI SERIES IN SOFTWARE ENGINEERING
Len Bass
Paul Clements
Rick Kazman
JUST ENOUGH SOFTWARE ARCHITECTURE
A RISK-DRIVEN APPROACH
GEORGE FAIRBANKS
FOREWORD BY DAVID GARLAN
Documenting Software Architectures
Software Systems Architecture
Second Edition
Using Viewpoints and Perspectives
EOIN WOODS
SOFTWARE ARCHITECTURE
FOUNDATIONS, THEORY, AND PRACTICE
Evaluating Software Architectures
Methods and Case Studies
Paul Clements
Rick Kazman
Mark Klein
HASSAN GOMAA
SOFTWA
MODELI
& DESIG
UML, USE CASES,
SOFTWARE AR
Model-Based Engineering with AADL
An Introduction to the SAE Architecture Analysis and Design Language
Peter H. Feiler
David P. Gluch

Quality Attributes

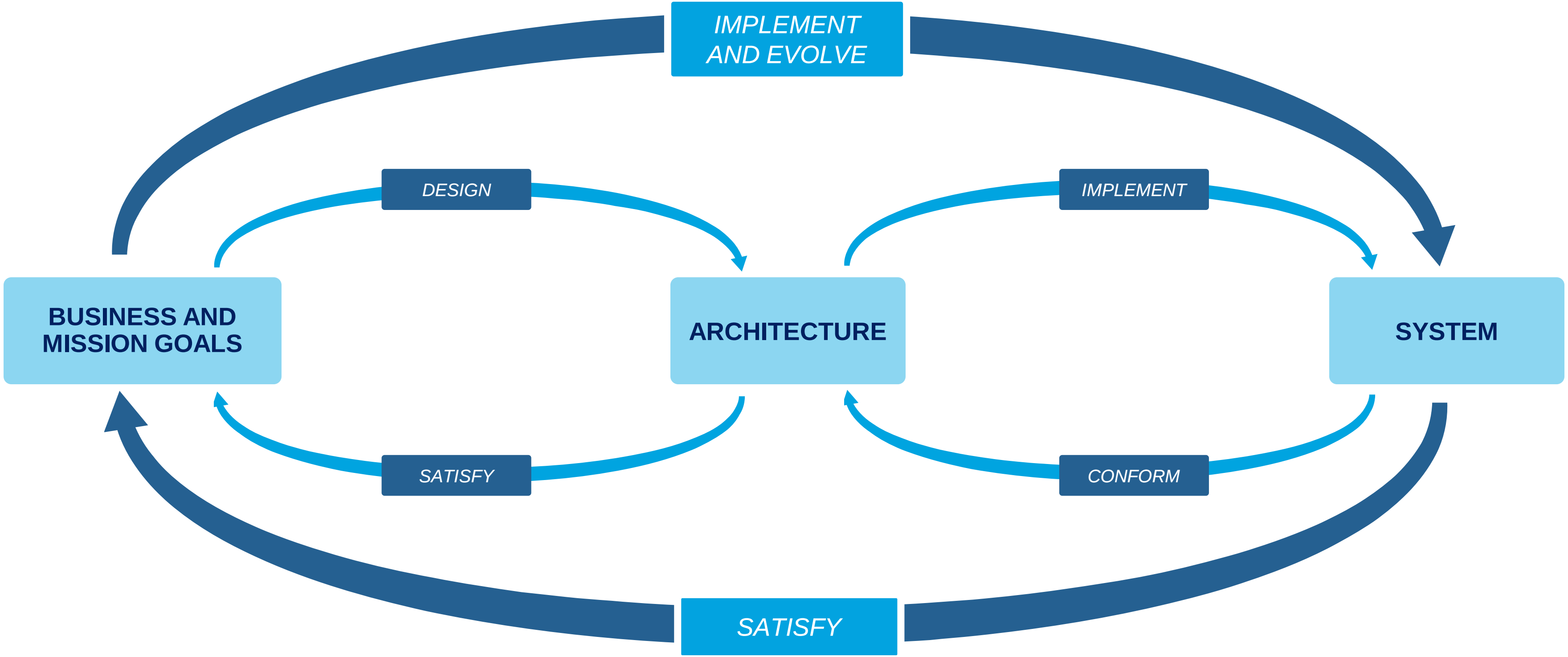
Quality attributes

- 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


IMPLEMENT AND EVOLVE

DESIGN

IMPLEMENT

BUSINESS AND MISSION GOALS

ARCHITECTURE

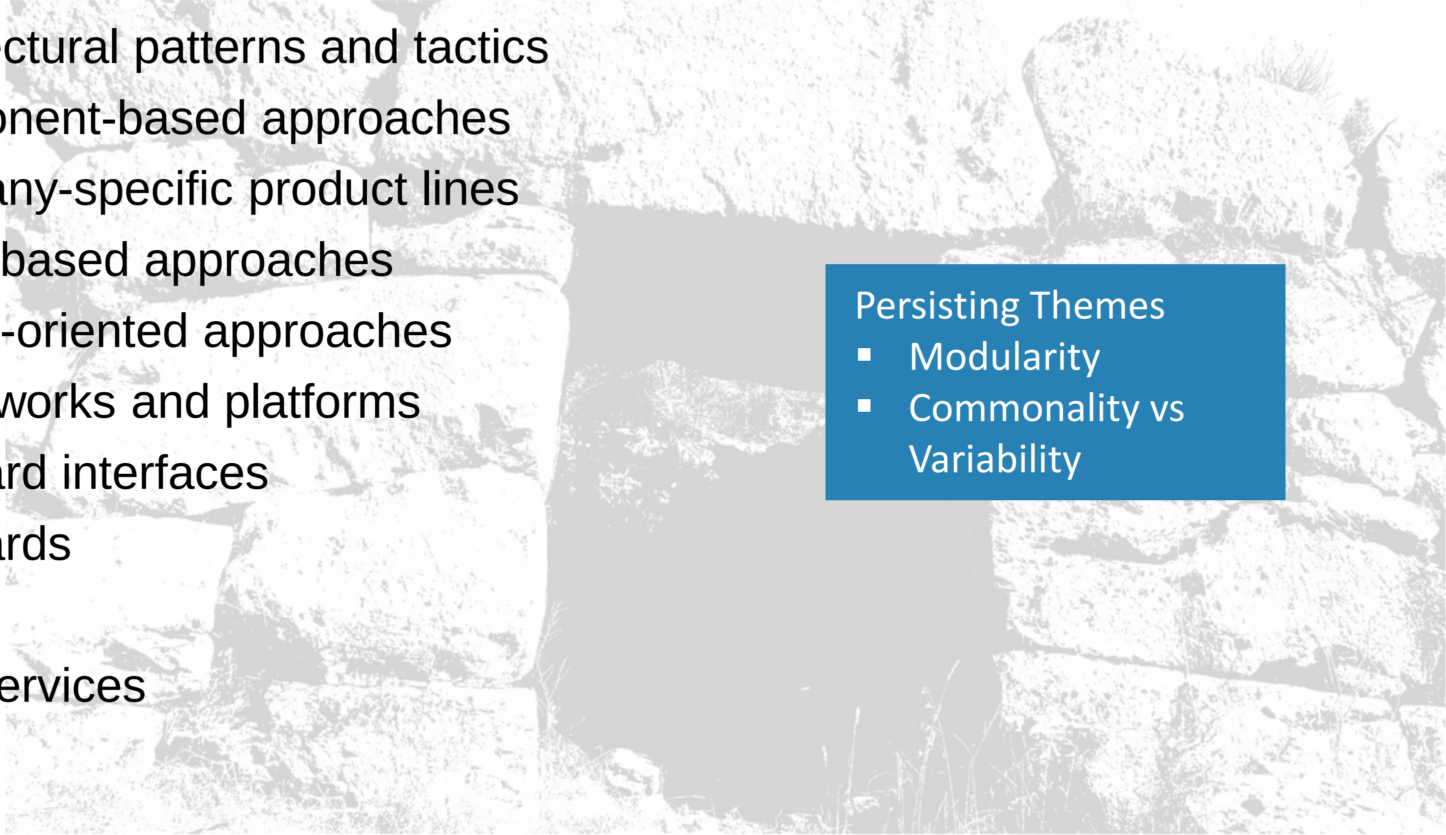
SYSTEM

SATISFY

CONFORM

SATISFY

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

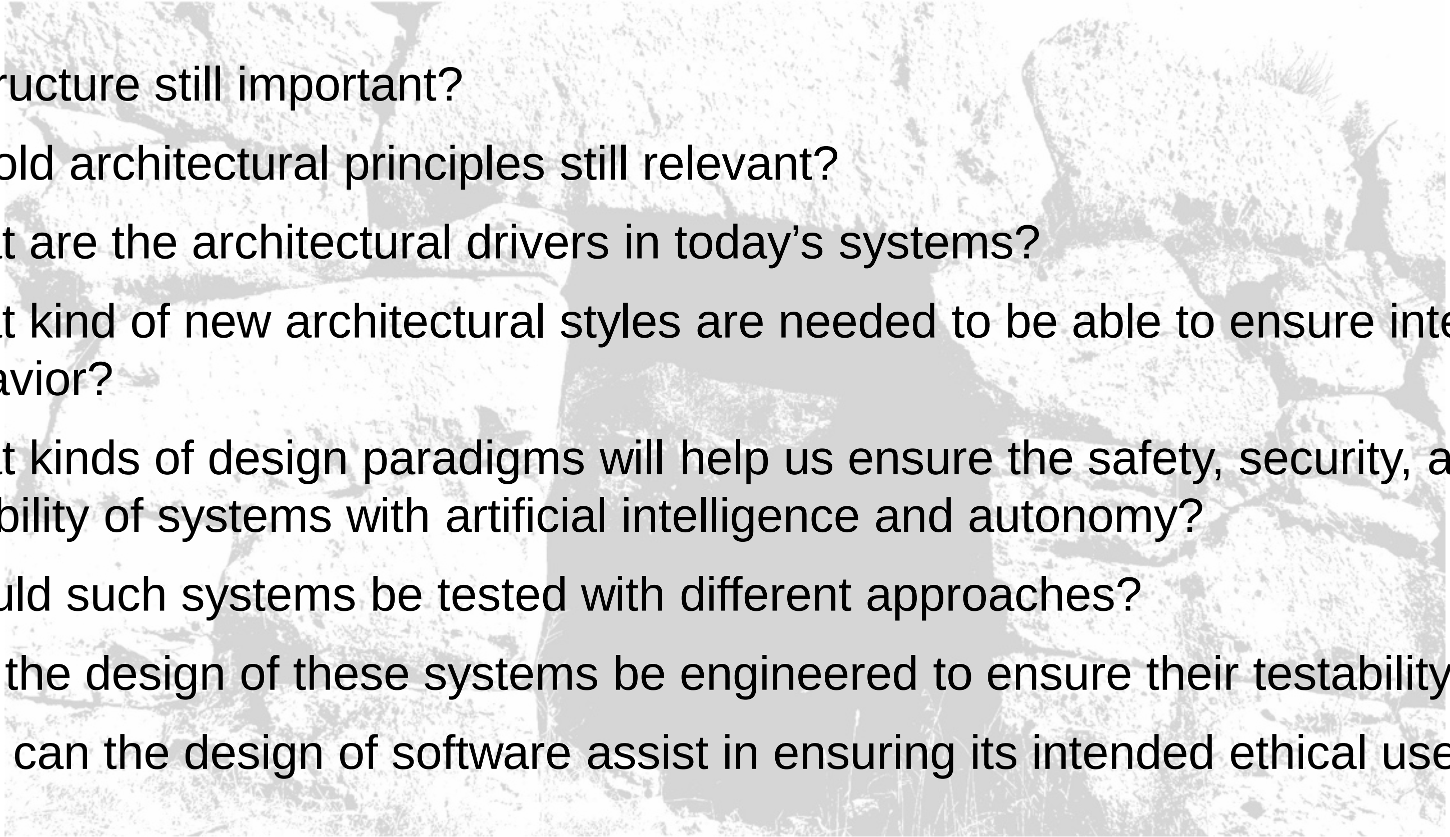
SOA

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?

Mobile Computing

BIG DATA

Cloud Computing

Software Architecture

SCALE

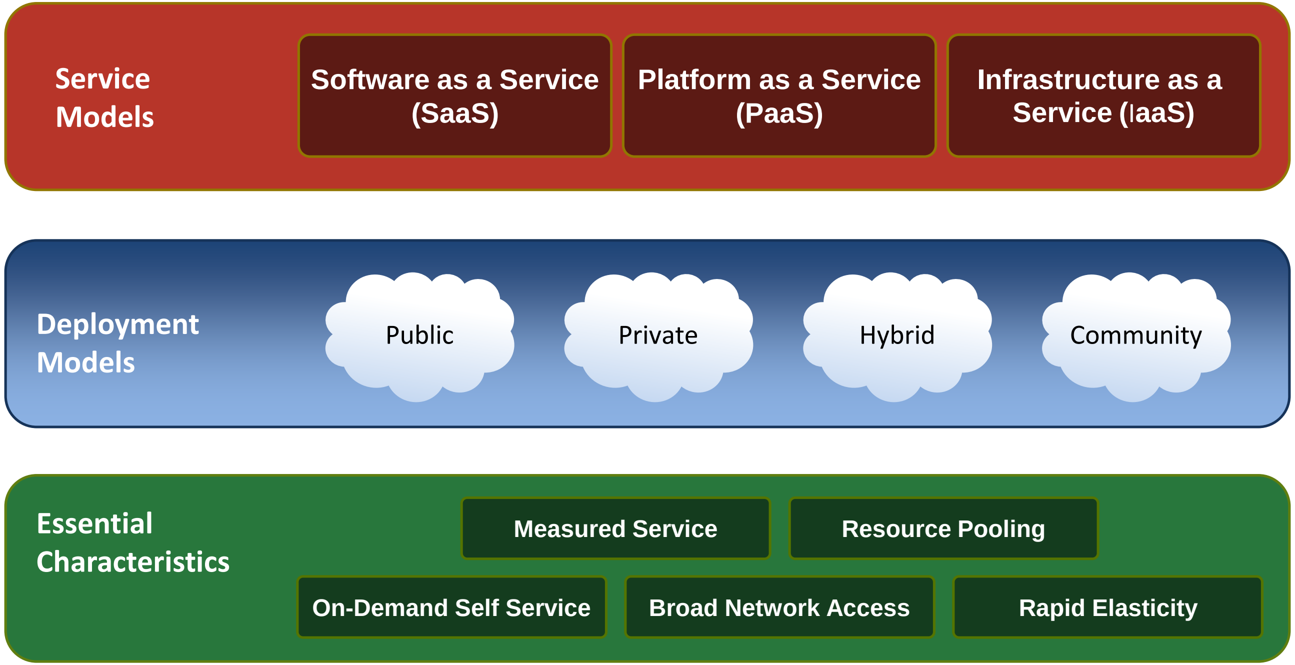
COMPLEXITY

AI Machine Learning

Cyber-Physical Social Ecosystems

Autonomous Systems

Cloud Computing Models and Essential Characteristics



Service Models

Software as a Service (SaaS)

Platform as a Service (PaaS)

Infrastructure as a Service (IaaS)

Deployment Models

Public

Private

Hybrid

Community

Essential Characteristics

Measured Service

Resource Pooling

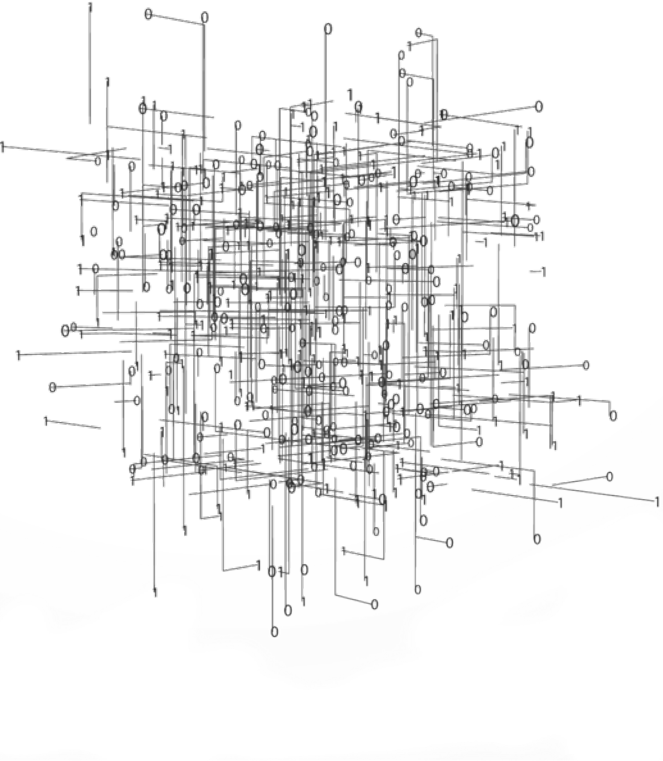
On-Demand Self Service

Broad Network Access

Rapid Elasticity

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

Special Cases

Migration to the cloud

- Challenges in interoperability, latency, legal, platform or language constraints, security, skill set, compliance, and portability
- Restructure to take advantage of cloud performance

Function as a Service (FasS)

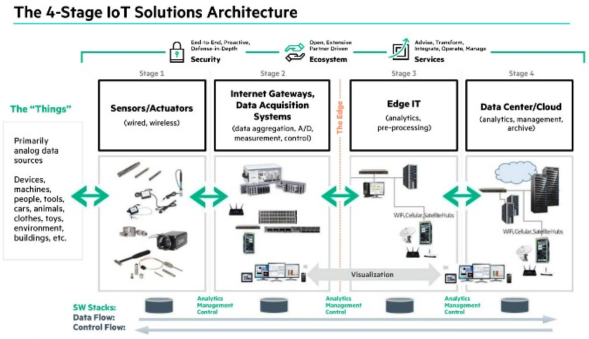
- Serverless architecture style
- Cloud-aligned architectures

Mobile Computing

Today’s UI is increasingly mobile.

Related Architecture Trends: Edge and Fog Computing

Smart Parking
Public Safety
Smart Buildings
Smart Health
Waste Management
Open Data
Education
Traffic Management
Smart Environment
Electromagnetic Emissions
Internet of Things
Smart Energy
Gas & Water Leak Detection
Smart Street Lights
SMART CITY
Air Pollution
Water Quality
Smart Home
Intelligent Shopping
Electric Vehicle Charging


The 4-Stage IoT Solutions Architecture
End-to-End, Proactive, Defense-in-Depth
Security
Open, Extensive Partner Driven
Ecosystem
Advise, Transform, Integrate, Operate, Manage
Services
Stage 1
Stage 2
Stage 3
Stage 4
The "Things"
Primarily analog data sources
Devices, machines, people, tools, cars, animals, clothes, toys, environment, buildings, etc.
Sensors/Actuators
(wired, wireless)
Internet Gateways, Data Acquisition Systems
(data aggregation, A/D, measurement, control)
The Edge
Edge IT
(analytics, pre-processing)
Data Center/Cloud
(analytics, management, archive)
WiFi,Cellular,SatelliteHubs
WiFi,Cellular,SatelliteHubs
Visualization
SW Stacks:
Data Flow:
Control Flow:
Analytics Management Control
Analytics Management Control
Analytics Management Control

Edge and Fog Computing push cloud resources to the edge of the network

Cyber-foraging is the process of discovering Edge and Fog resources for computation offload and data staging

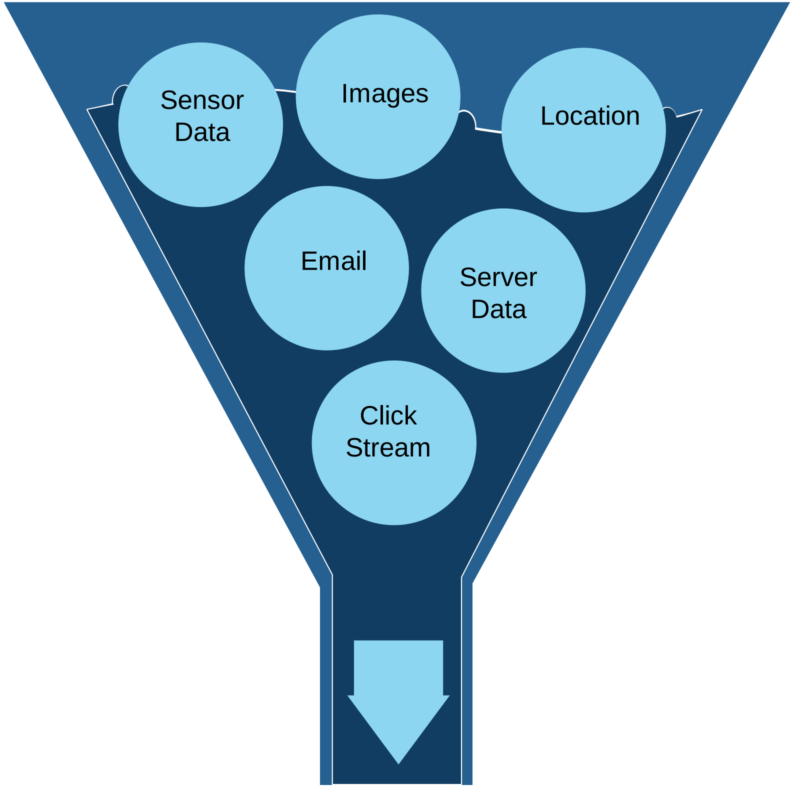
- Reduced network traffic
- Reduced latency
- Improved user experience

Architecture challenges

- Data and computation allocation to the right nodes at the right time
- Resource discovery
- Security and privacy

Source: https://www.hpe.com/us/en/insights/articles/the-intelligent-edge-what-it-is-what-its-not-and-why-its-useful-1704.html

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

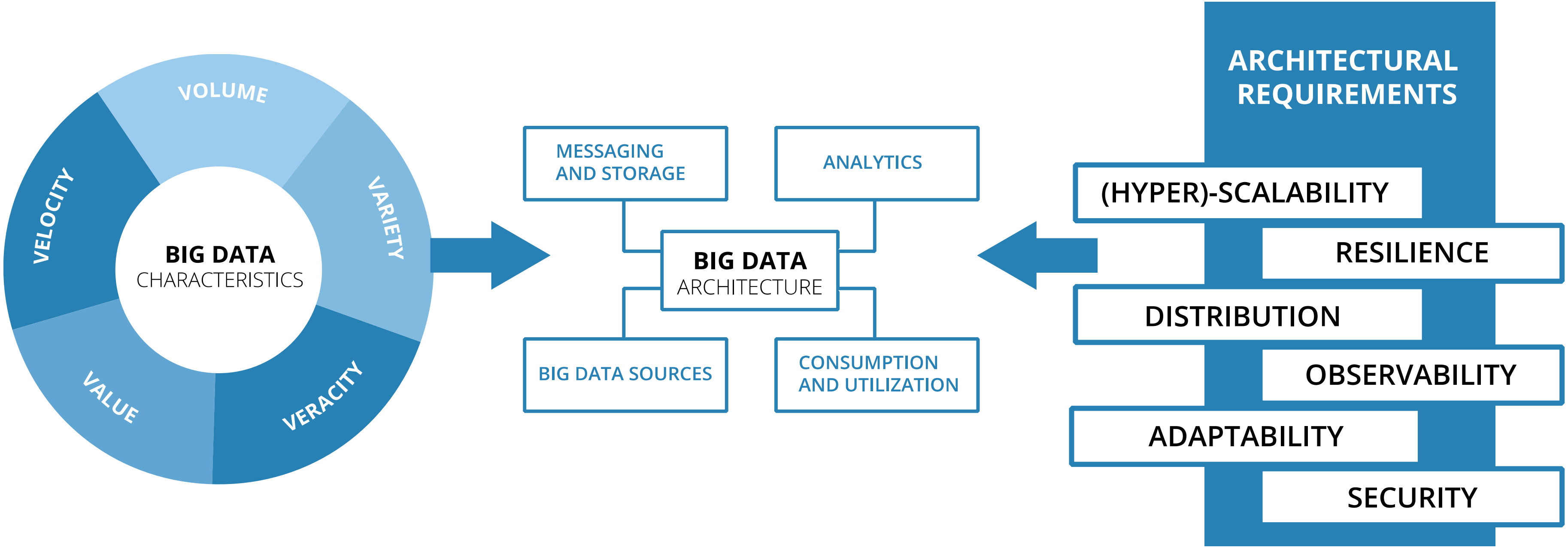
- fast
- scalable
- secure
- easy to use

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

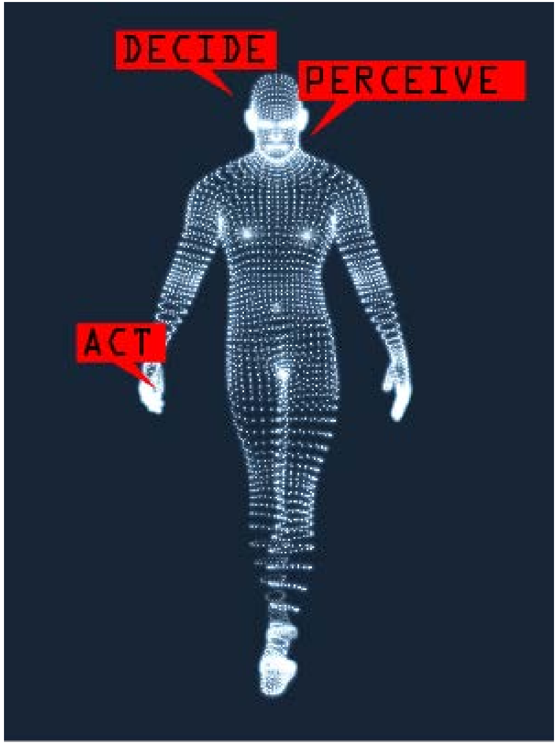
COMPUTING
edge
DATA
BIG DATA & ANALYTICS
Also in this issue:
> Interview with Award-Winning New York Times Journalist on Cybersecurity
> Should Architects Code?
MARCH 2018
IEEE
computer society

Data is a business asset.

Big Data Software Architecture



Artificial Intelligence (AI)



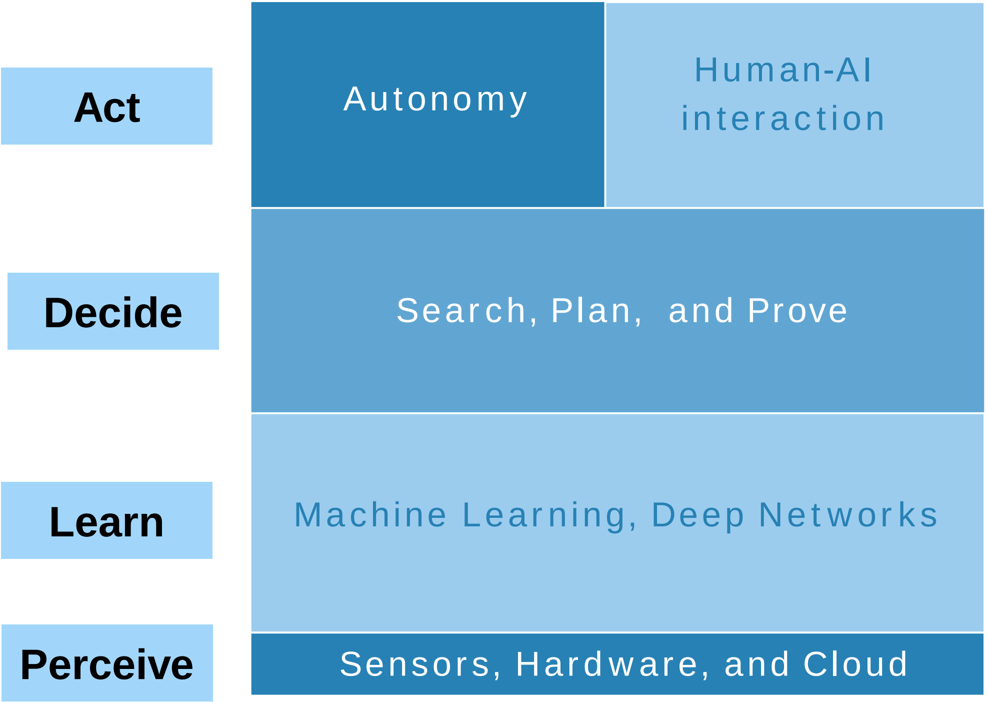
AI is creating a revolution in system capability.

- 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



Act

Decide

Learn

Perceive

Autonomy

Human-AI interaction

Search, Plan, and Prove

Machine Learning, Deep Networks

Sensors, Hardware, and Cloud

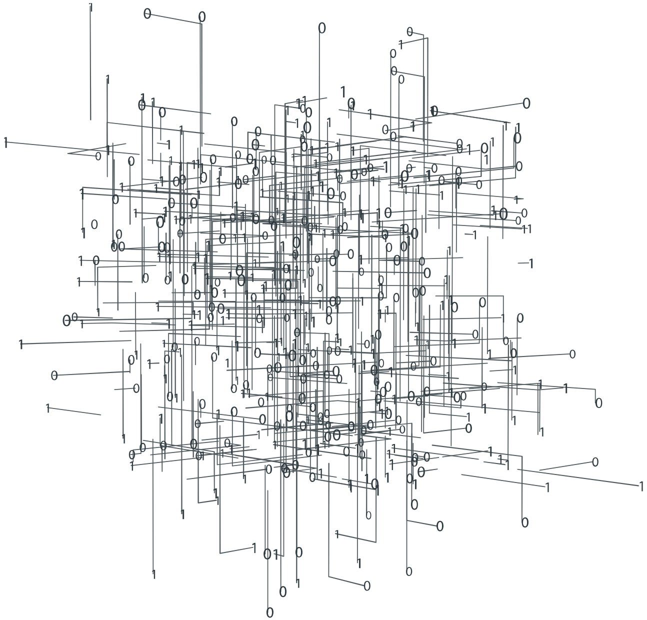
“”A Vision for Software Development,” Andrew Moore, Carnegie Mellon University, Jan 6, 2018

Machine Learning (ML) Systems Today

Machine learning: learning to predict by extrapolating from data

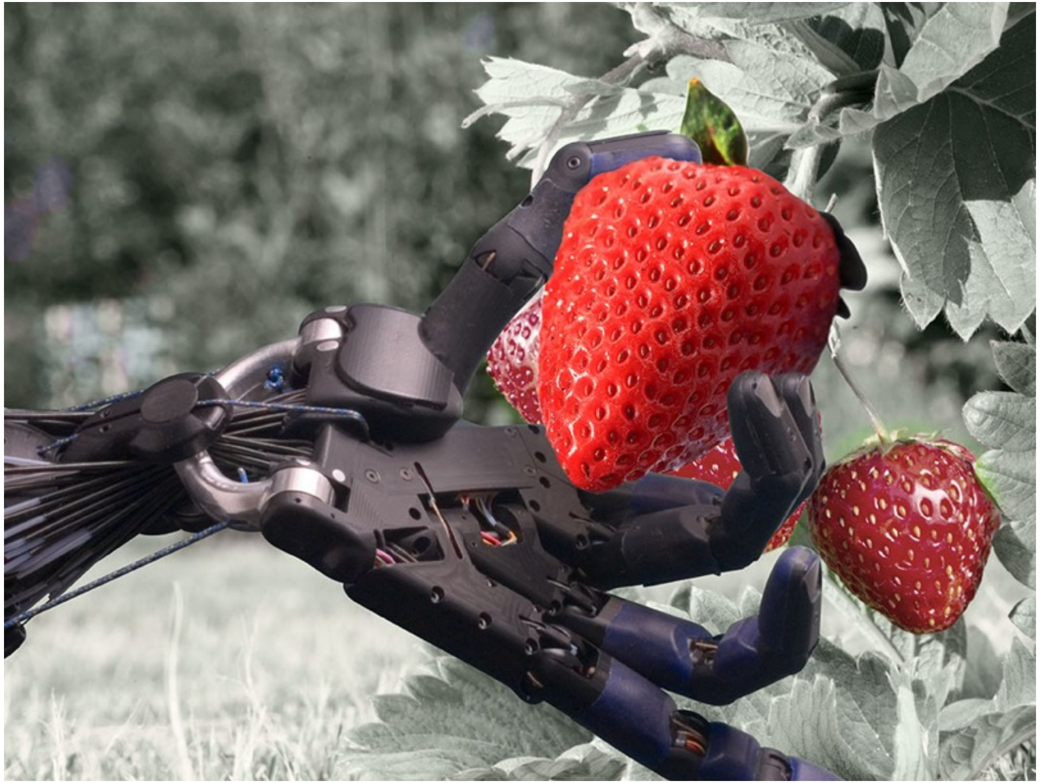
- Can provide rapid response to dramatically changing contexts
- Algorithms are readily accessible
- Effectiveness is highly variable across different domains
- Overall, today still a cottage industry

ML, Software Engineering, and Architecture

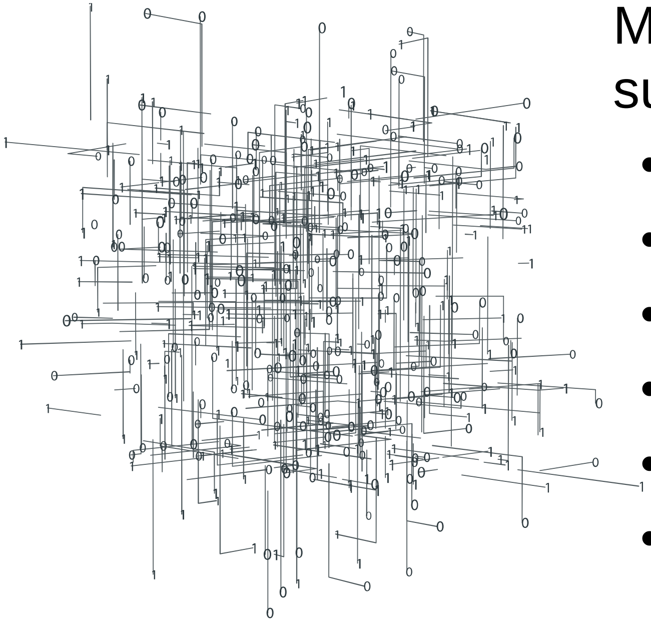


1. Correctness will not be possible.
   - ML has an experimental mindset.
2. Holistic testing is impossible
   - uncertainty and error will be part of the output
3. Deductive reasoning from the code and metadata is not and will not be effective.
4. Data and its attributes must be first class.
5. Divide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

Autonomous Systems



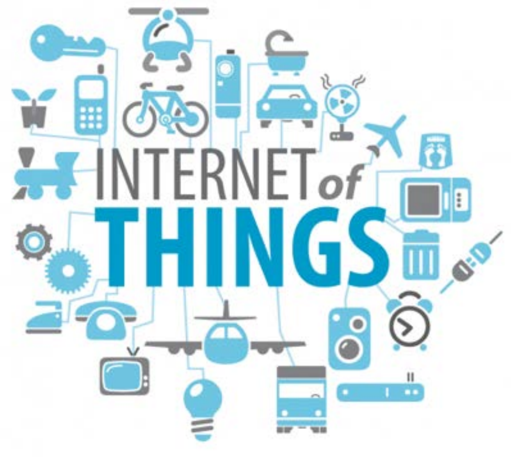
Autonomous Systems, Software Engineering, and Architecture

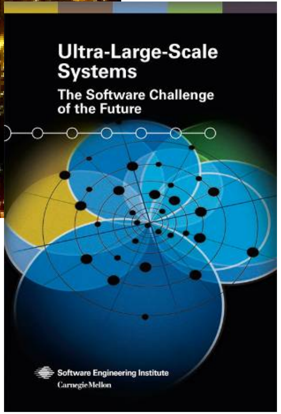


ML issues plus structural support for

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

Cyber-Physical-Social Ecosystems



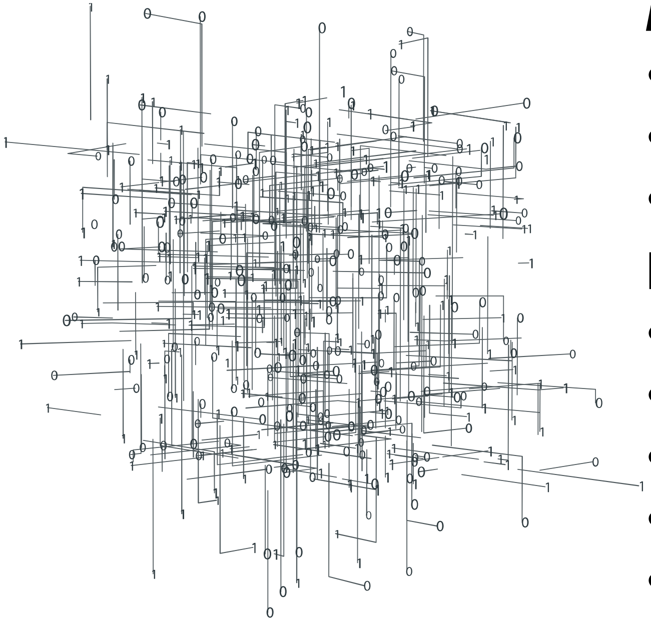


Ultra-Large-Scale Systems

The Software Challenge of the Future

Software Engineering Institute
Carnegie Mellon

Software Engineering and Architecture

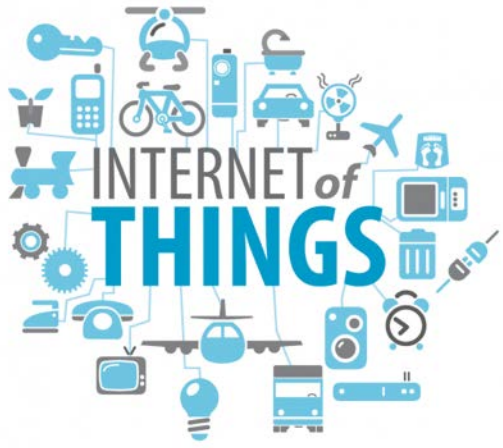


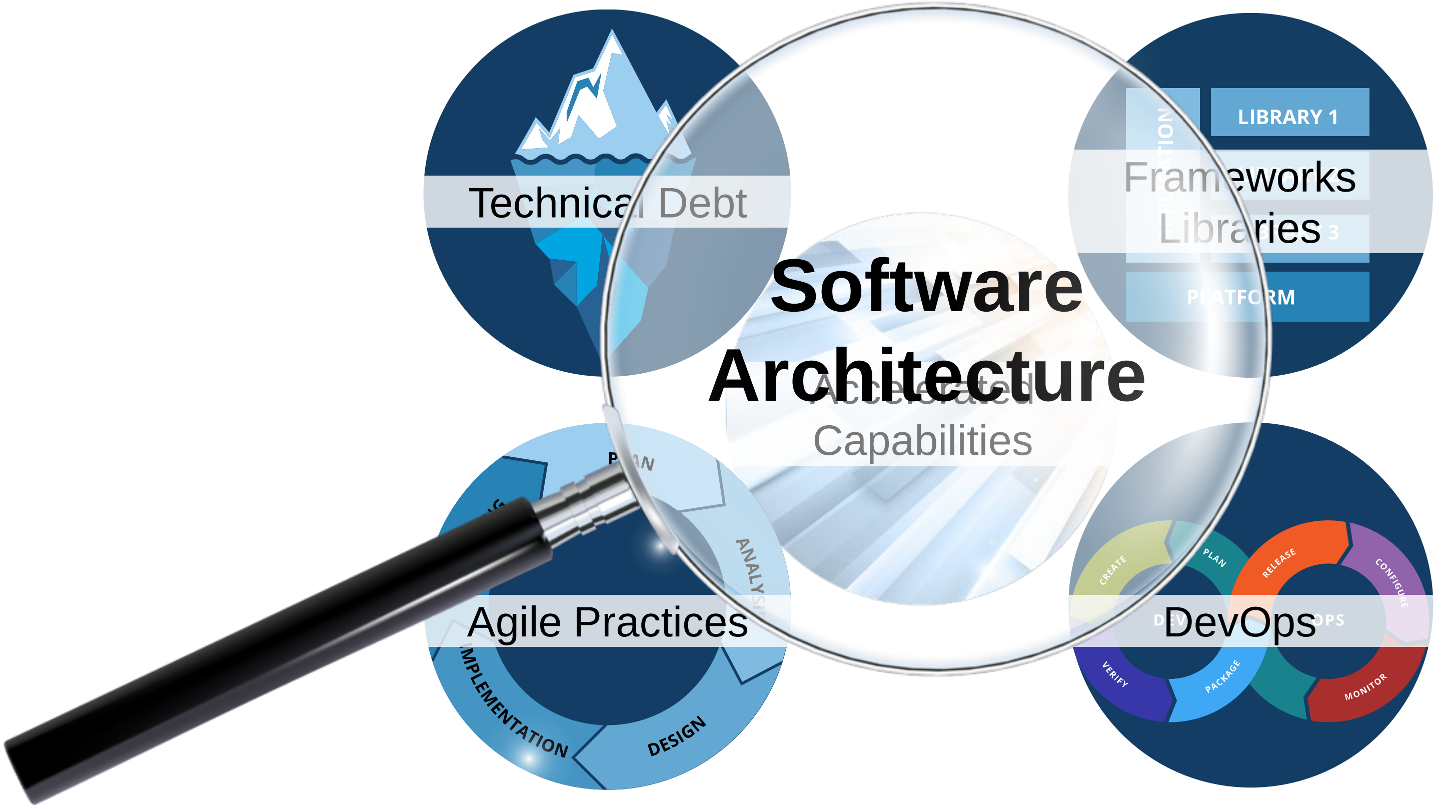
Design of and at all levels

- Governance
- Standards
- Certification

Platforms that admit heterogeneity and provide

- Interoperability
- Scalability
- Extensibility
- Timing
- Security
- Monitorability
- Resilience/self-adaptation





Technical Debt

Frameworks
Libraries

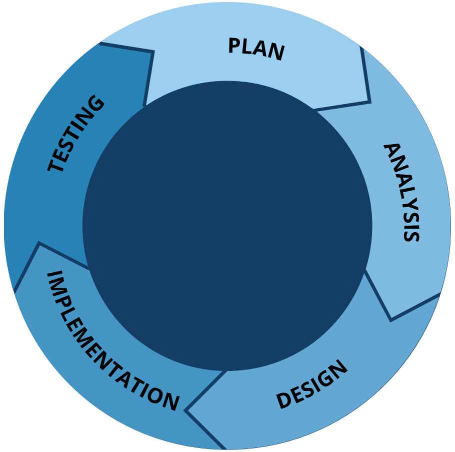
Software Architecture

Accelerated Capabilities

Agile Practices

DevOps

Incremental Development and Architecture



Architecture design can be done incrementally.

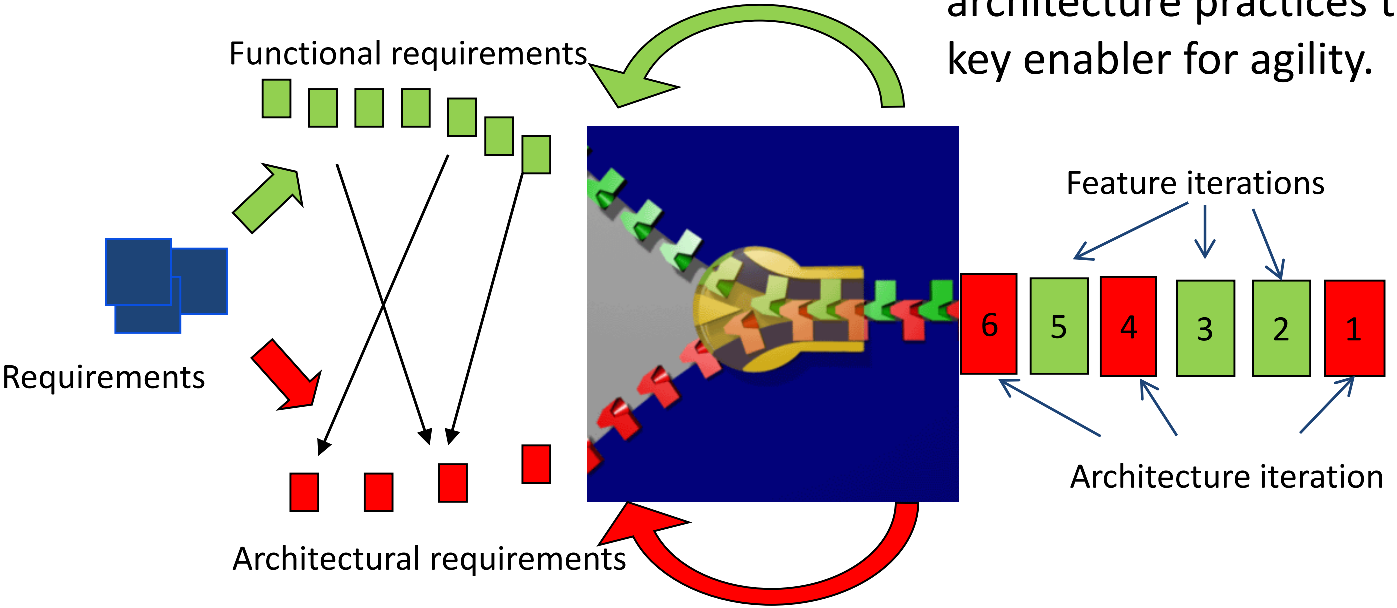
There is a difference between being agile and doing agile.

Agility is enabled by architecture – not stifled by it.

Architecture needs to be versatile, easy to evolve, and easy to modify, while resilient enough not to degrade after a few changes.

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

Integrated Agile/Architecture Practices



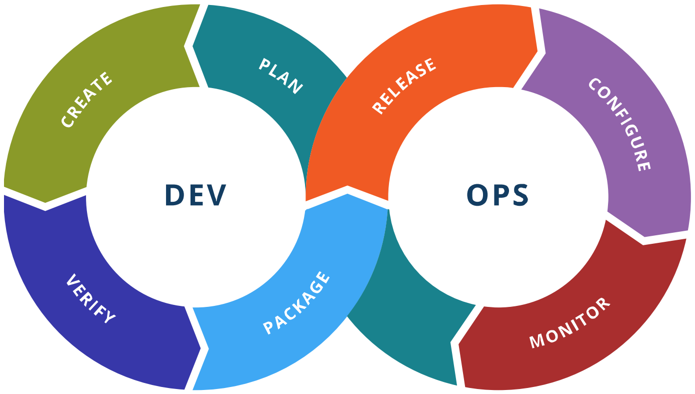
Successful project teams find architecture practices to be a key enabler for agility.

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.

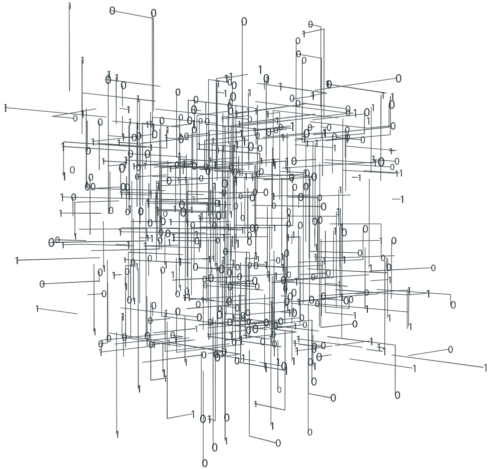
DevOps

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.

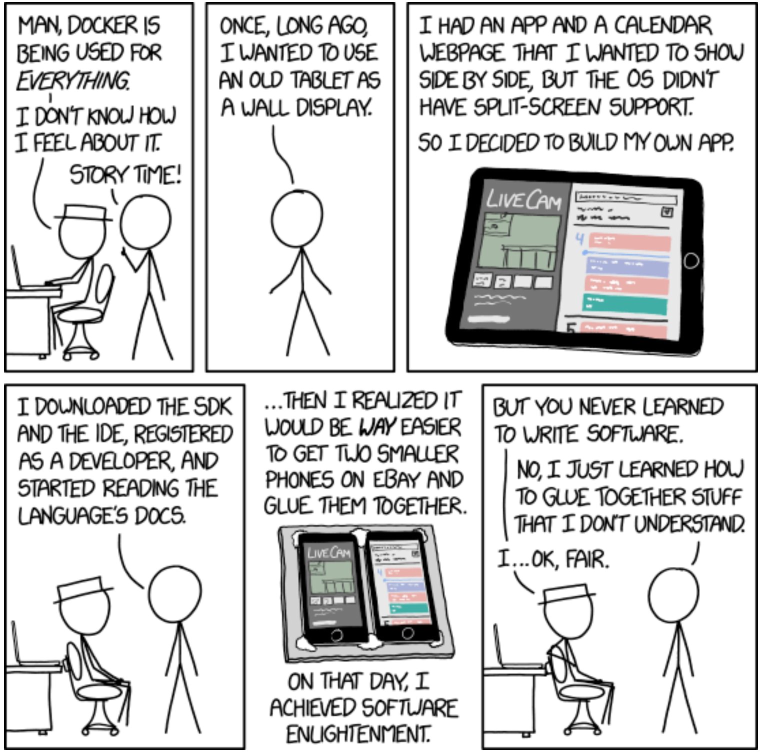
Deployability Tactics

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.


MAN, DOCKER IS BEING USED FOR EVERYTHING.
I DON'T KNOW HOW I FEEL ABOUT IT.
STORY TIME!
ONCE, LONG AGO, I WANTED TO USE AN OLD TABLET AS A WALL DISPLAY.
I HAD AN APP AND A CALENDAR WEBPAGE THAT I WANTED TO SHOW SIDE BY SIDE, BUT THE OS DIDN'T HAVE SPLIT-SCREEN SUPPORT.
SO I DECIDED TO BUILD MY OWN APP.
LIVECAM
I DOWNLOADED THE SDK AND THE IDE, REGISTERED AS A DEVELOPER, AND STARTED READING THE LANGUAGE'S DOCS.
...THEN I REALIZED IT WOULD BE WAY EASIER TO GET TWO SMALLER PHONES ON EBAY AND GLUE THEM TOGETHER.
LIVECAM
ON THAT DAY, I ACHIEVED SOFTWARE ENLIGHTENMENT.
BUT YOU NEVER LEARNED TO WRITE SOFTWARE.
NO, I JUST LEARNED HOW TO GLUE TOGETHER STUFF THAT I DON'T UNDERSTAND.
I...OK, FAIR.

Containers | xkcd.com

Reuse abounds.

Rapid construction through assembly.

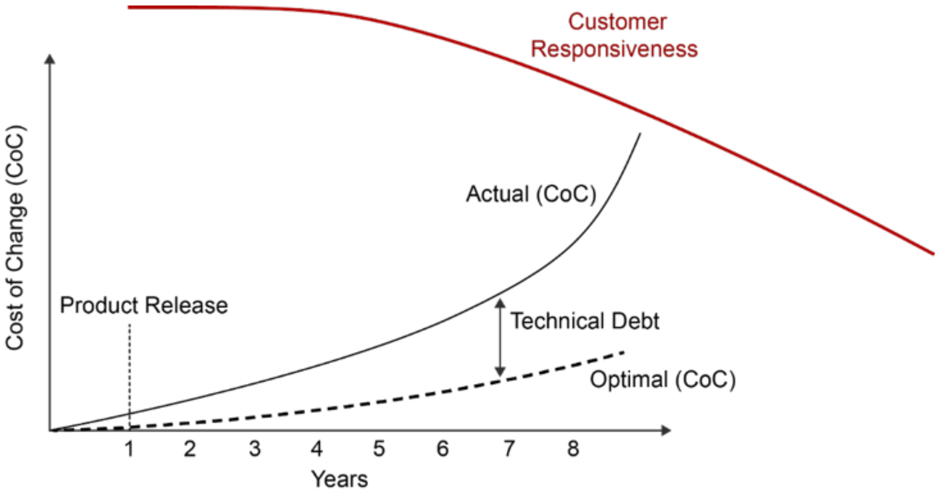
Architecture is implicit.

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.



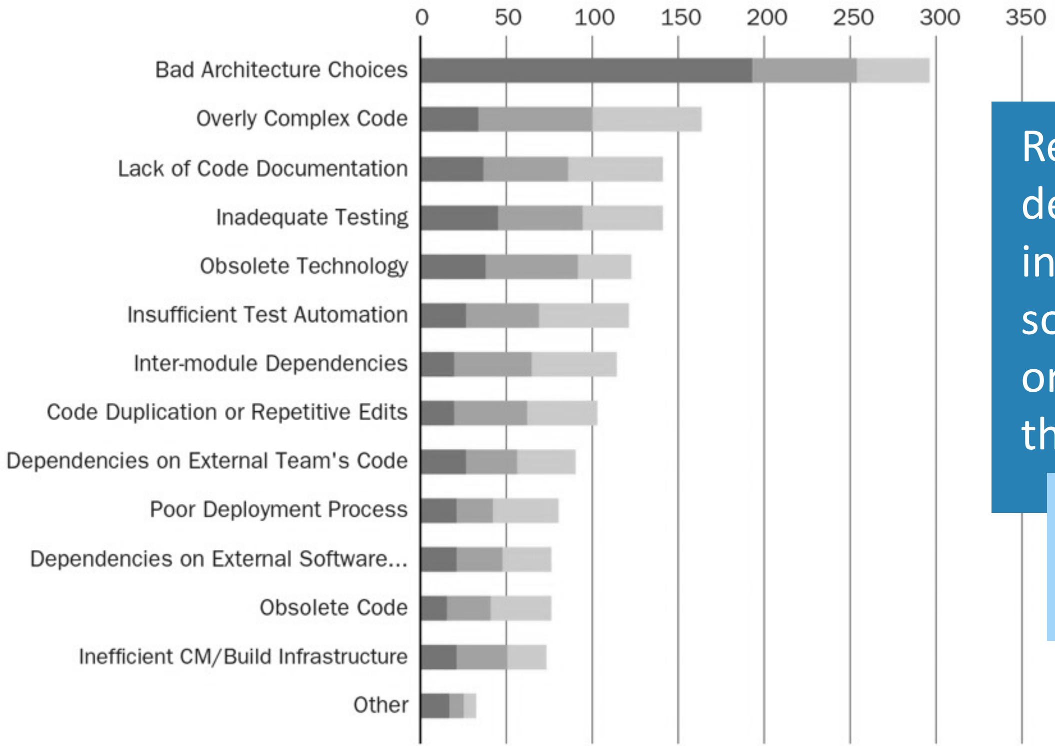
Exists in

- 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/

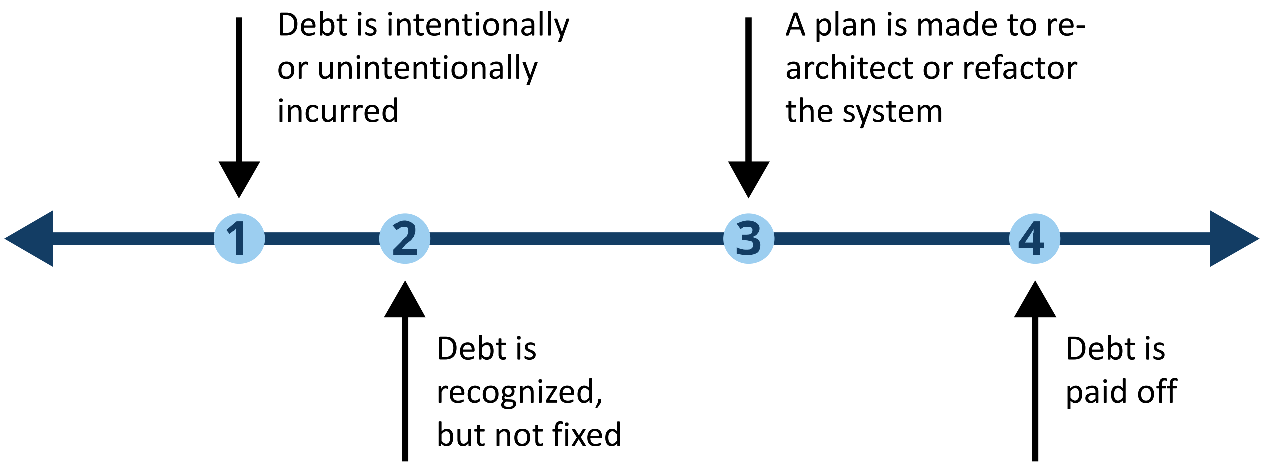
Software Architecture and Design Tradeoffs Matter



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


Debt is intentionally or unintentionally incurred
A plan is made to re-architect or refactor the system
1
2
3
4
Debt is recognized, but not fixed
Debt is paid off

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.

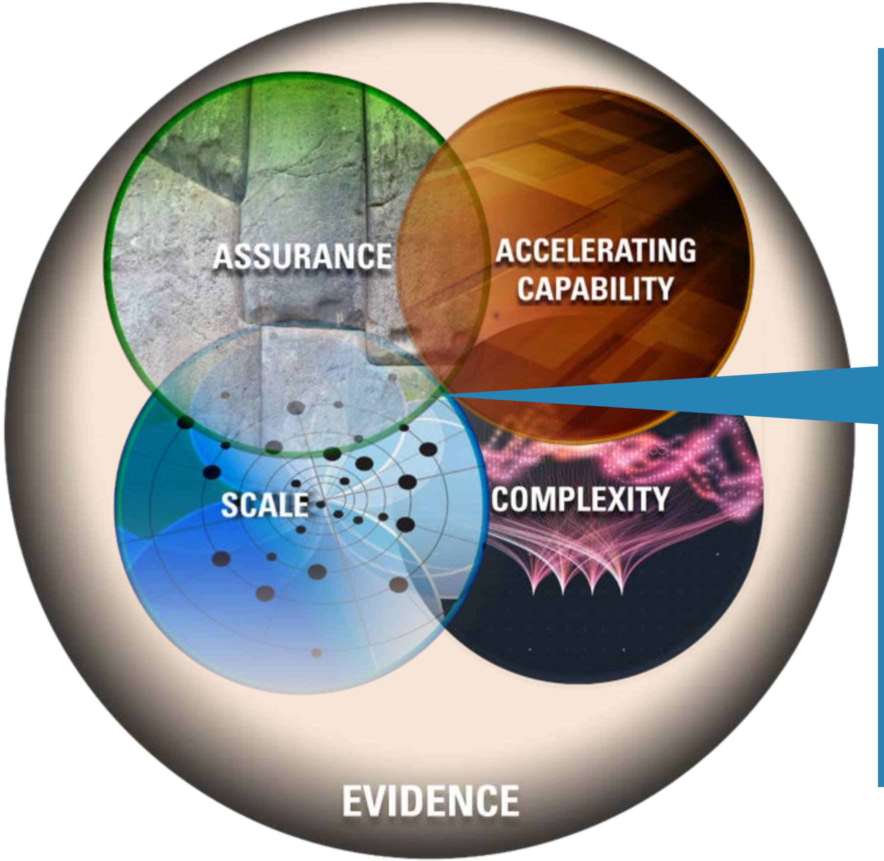
Tool chains that generate code, configuration files, etc. introduce unknowns.

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 and evidence, 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

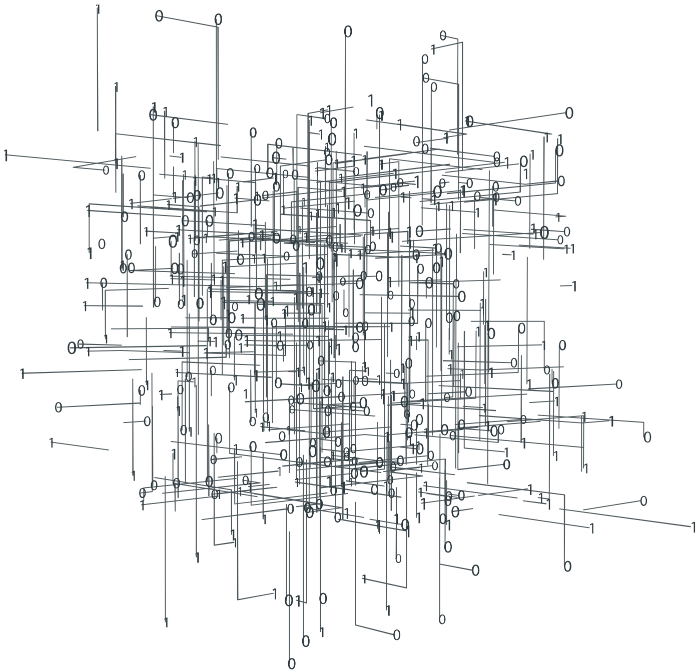
Has a spectrum of experience

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.

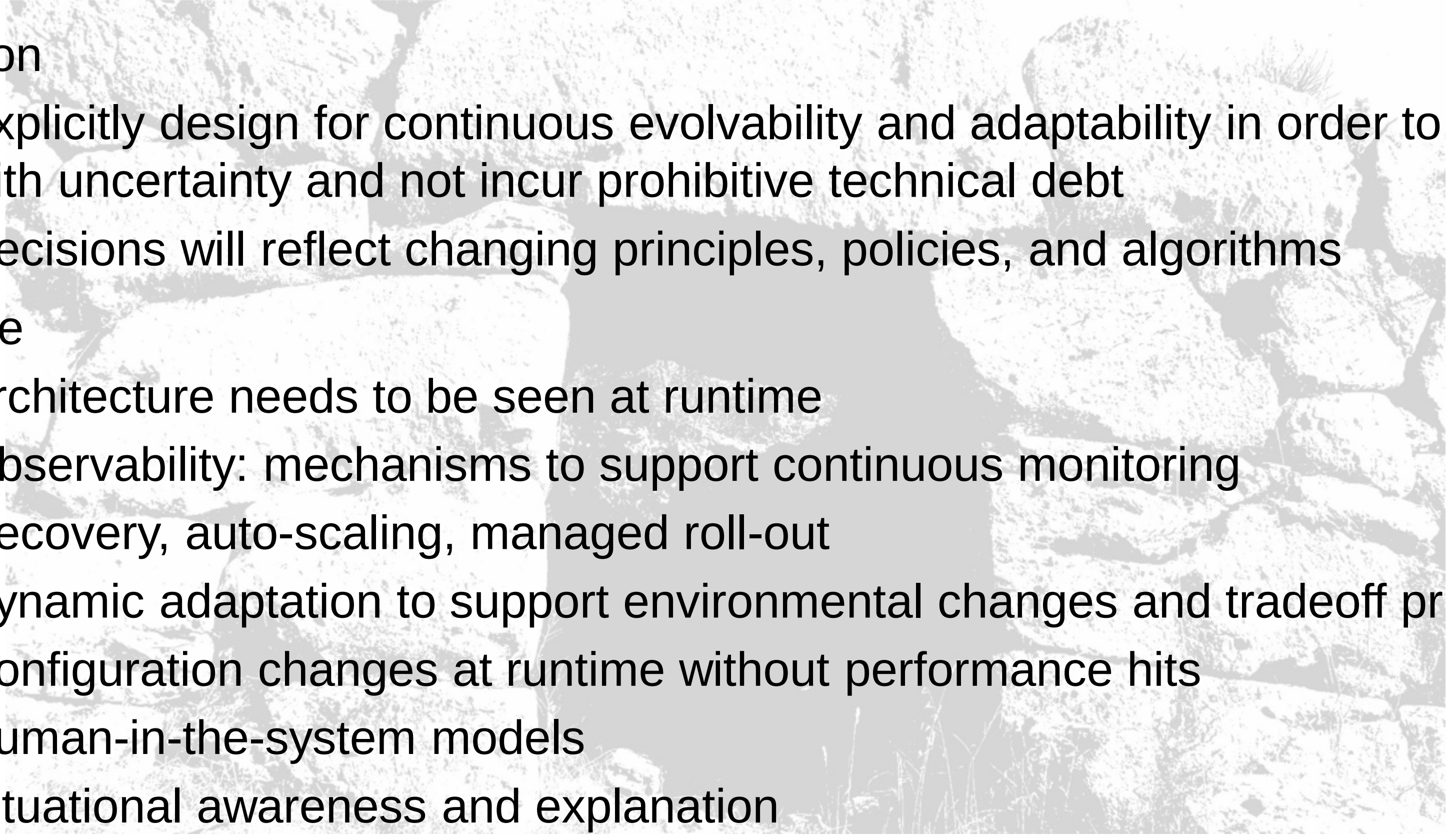
Security needs are heightened.

Different quality attributes at the fore.

New focus on

- Evolution
- Runtime
- Data
- ML
- Automation

Evolution and Runtime



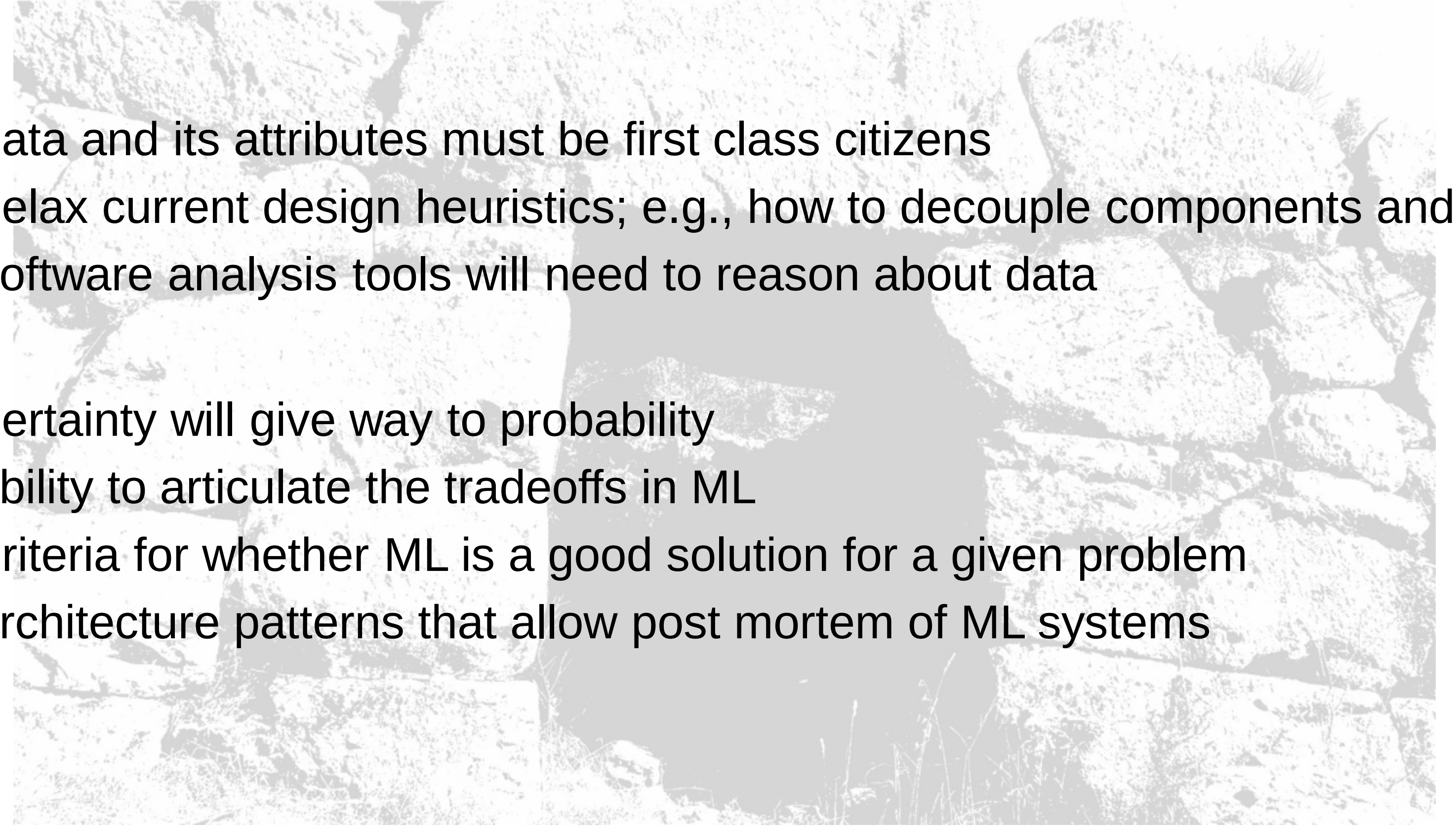
Evolution

- 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



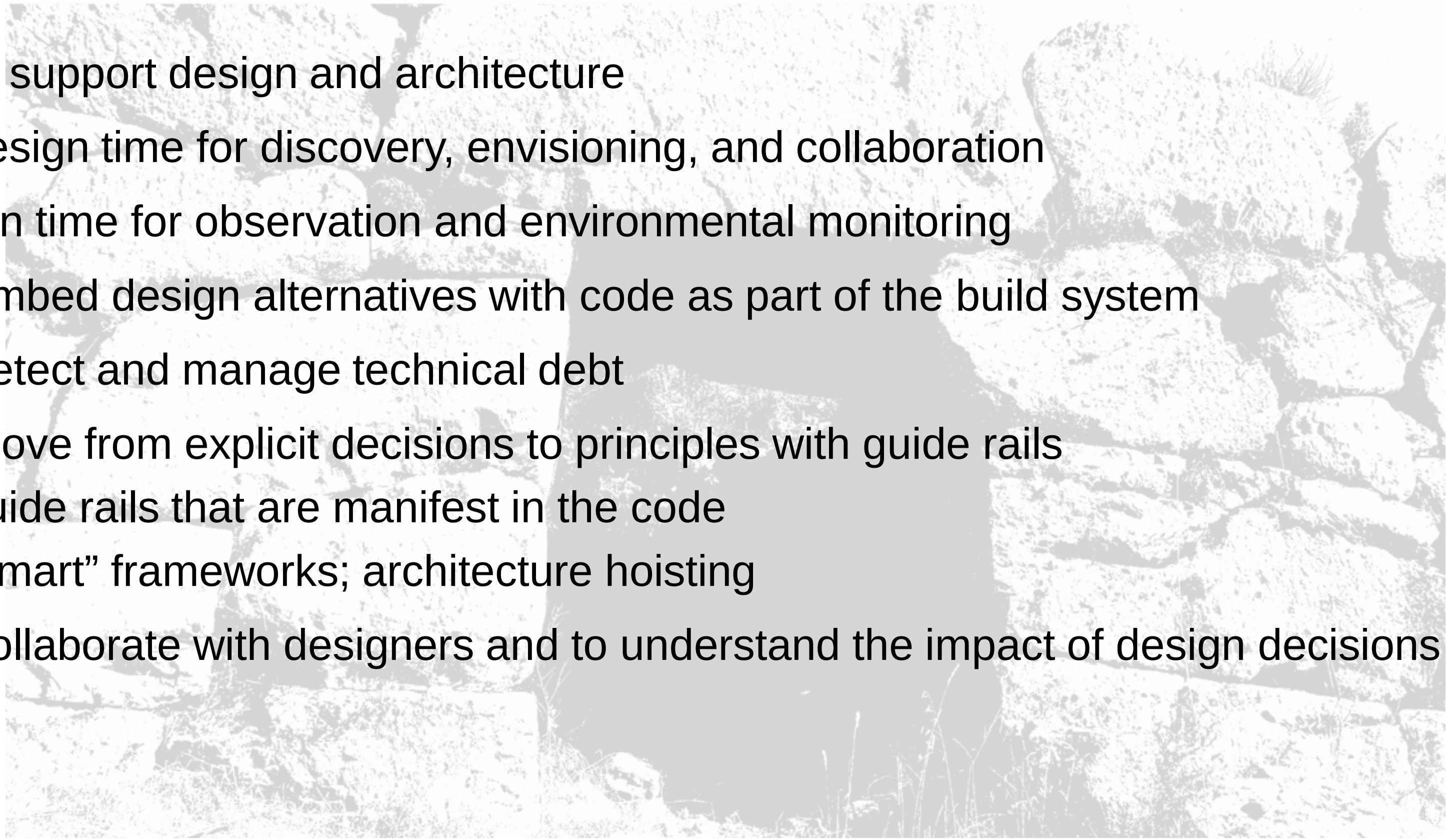
Data

- 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

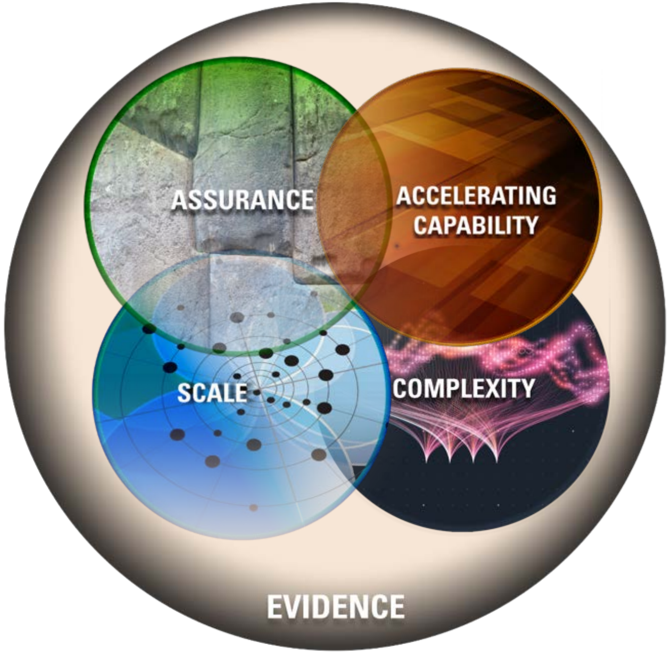


Tools to support design and architecture

- 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
- To move from explicit decisions to principles with guide rails
  - guide rails that 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