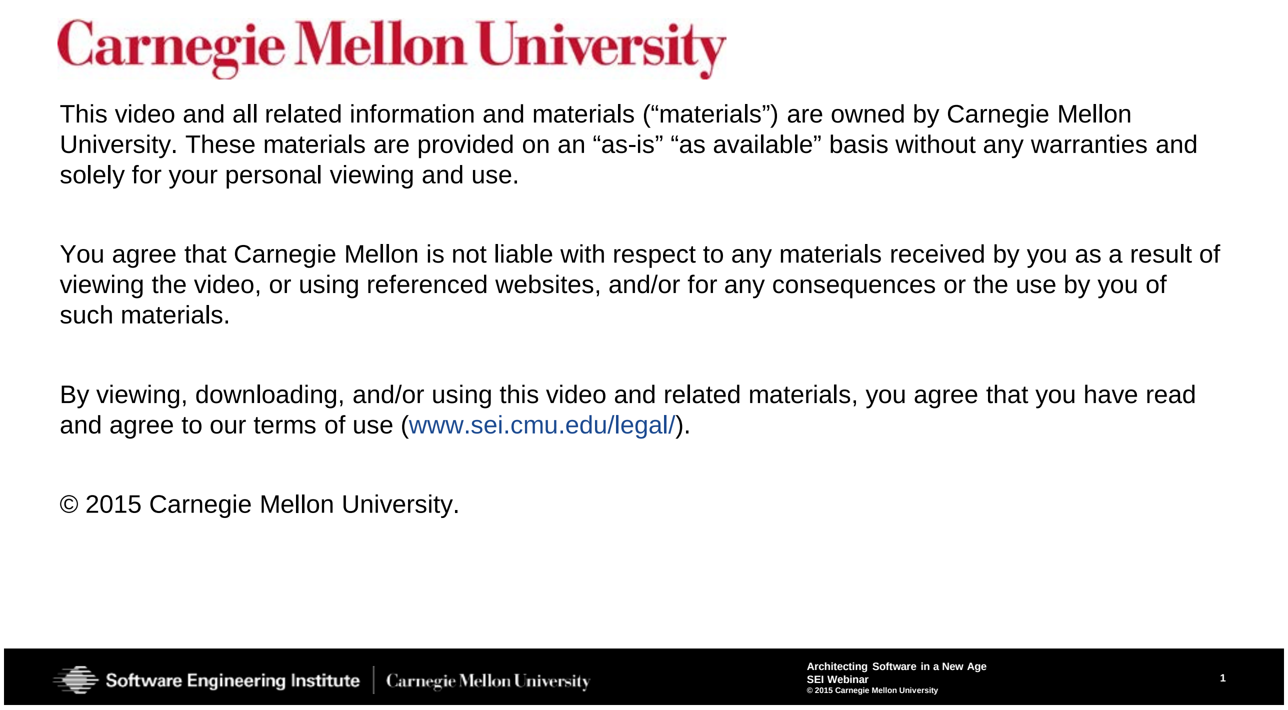
Trends and New Directions in Software Architecture

Table of Contents

Carnegie Mellon University Notice



Software Engineering Institute | Carnegie Mellon University

Architecting Software in a New Age
SEI Webinar
© 2015 Carnegie Mellon University

1

**001 Shane McGraw: And hello from the campus of Carnegie Mellon University in Pittsburgh, Pennsylvania. We welcome you to the Software Engineering Institute's webinar series. Our presentation today is Architecting Software in a New Age. Depending on your location, we wish you a good morning, a good afternoon, or a good evening. My name is Shane McGraw, your moderator for today, and I'd like to thank you for attending.

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

**002 For any questions you have pertaining to the presentations today, we will address all questions at the end of the second presentation. So you can log your questions at any time within the webcast or console, but we will address all questions at the end of the second presentation.

We're also going to ask a couple polling questions throughout the day. In fact, we're going to launch our first polling question for you to answer now, and what we'd like to know is: How did you hear about today's event? Let's take a couple seconds to answer that.

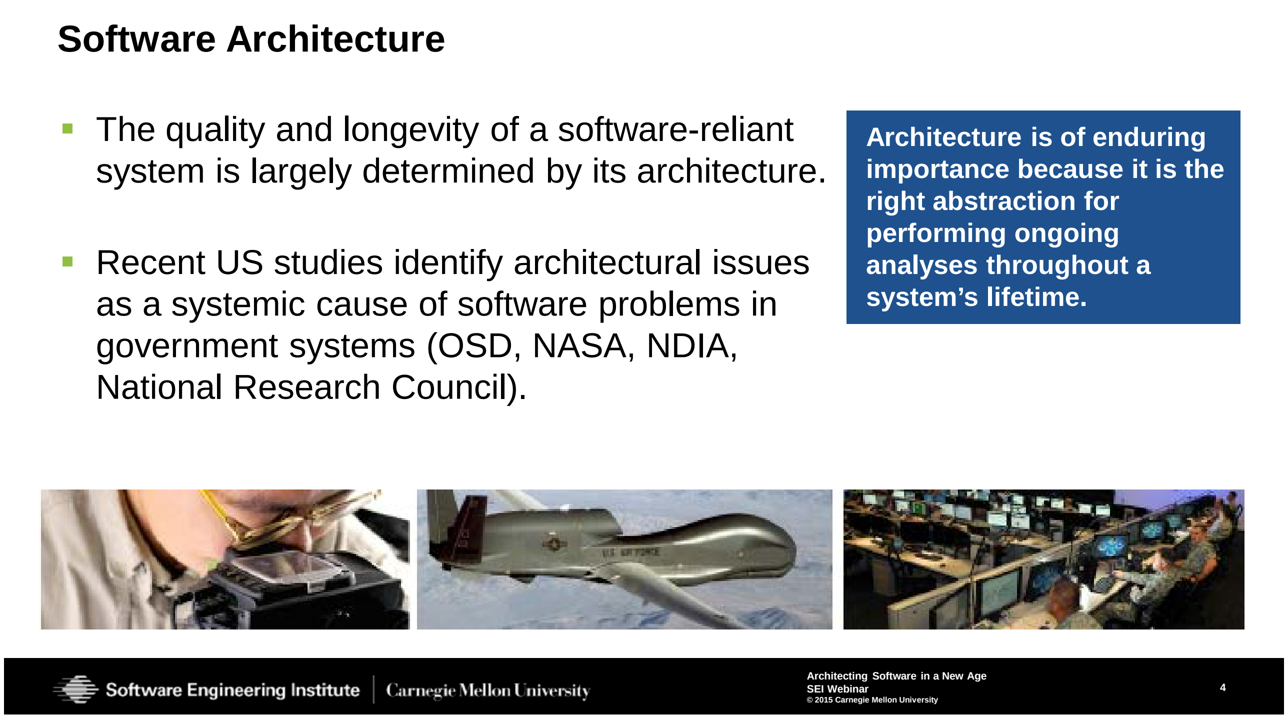
While you're doing that, I'd like to point out another three tabs that you're going to see on the console, and they are the Files tab, the

Twitter tab, and the Survey tab. The Files tab has a PDF copy of the presentation slides there now, along with other software architecture related conferences and training from the Software Engineering Institute. For those of you using Twitter, be sure to follow @saturn_news, and use the hashtag #seiswarch. Once again, it's @saturn_news, and #seiswarch, as in software architecture.

Now I'd like to introduce our first speaker for today, and the first talk is going to be Trends and New Directions in Software Architecture by Linda Northrup, and Linda will speak from one thirty to two fifteen.

Linda is chief scientist of the Software Solutions Division at the SEI, where the technical agenda compromises architecture-centric engineering, software development, and acquisition practices, measurement, software product lines, cyber physical systems, advanced mobile systems, and ultra-large-scale systems. Linda is coauthor of the book "Software Product Lines: Practices and Patterns," and led the research group on ultra-large-scale systems, or ULS, that resulted in the book "Ultra-Large-Scale Systems: The Software Challenge of the Future." Now I'd like to turn it over to Linda Northrup. Linda, all yours.

Linda Northrop: Thanks very much. I am absolutely delighted to be giving this webinar--



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 Engineering Institute | Carnegie Mellon University

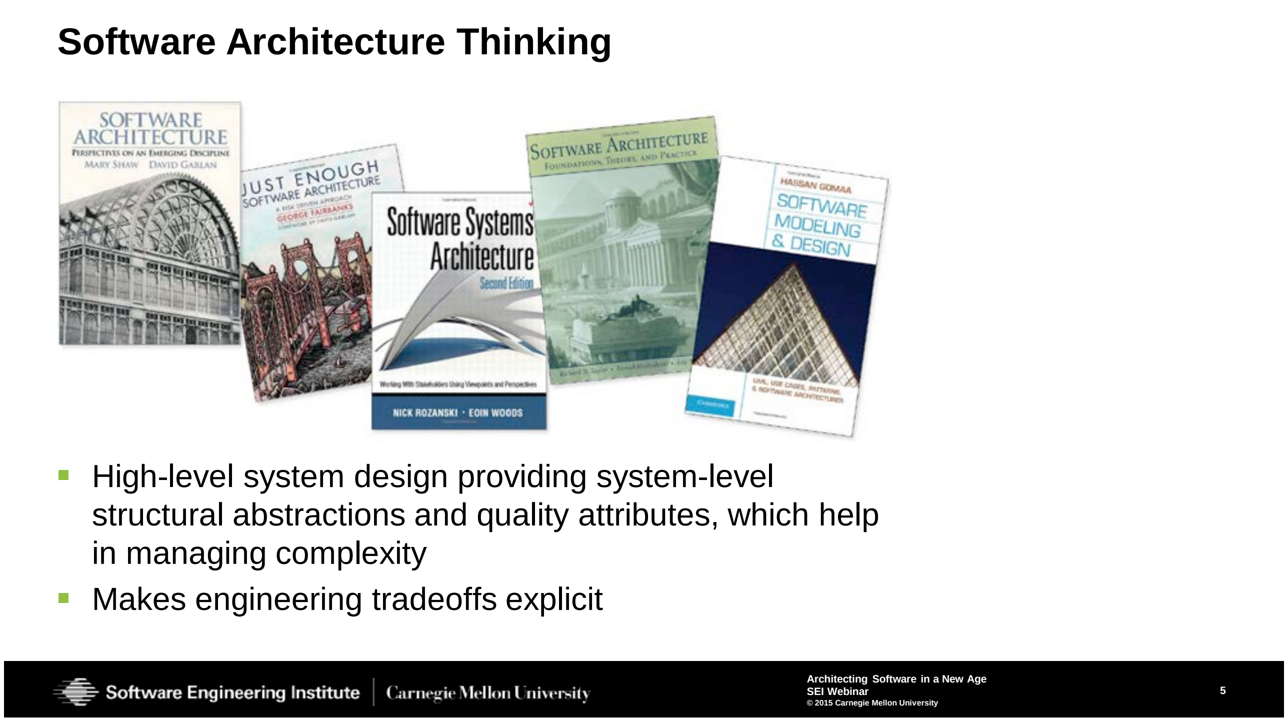
Architecting Software in a New Age
SEI Webinar
© 2015 Carnegie Mellon University

4

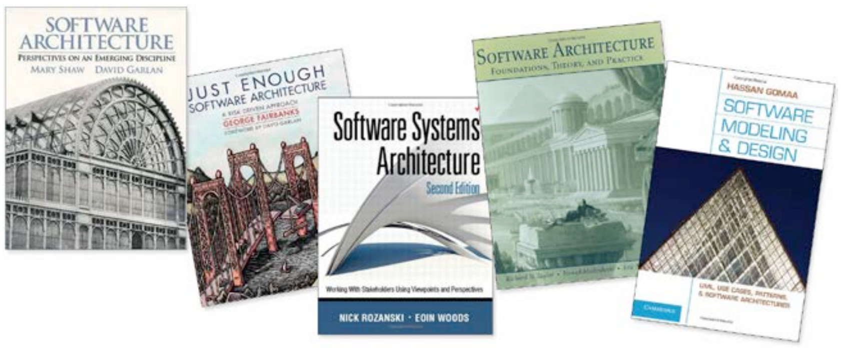
**004 --And what I'm going to be talking about is software architecture. Now, for those of you who don't know much about software architecture, you will understand Soft basics about software architecture, its importance, and why we believe it's critical to the quality and longevity of a software system. For those of you already savvy about software architecture, I hope that you will come away with some new perspectives about trends and challenges that we face in architecting today's systems, as well as some of the practices and current research. that we'll address.

So, basically, it has long been our premise that the quality and longevity of a software reliance system is largely determined by its architecture, and many studies have led us to that conclusion. There are many things we see in systems, like communication bottlenecks under certain loads and difficulty in integrating and testing and adding new features, and all of those actually are rooted in some architectural decisions that don't support the needs of the system. In fact, I will posit that architectural considerations are absolutely key to the quality of a software reliance system.

Software Architecture Thinking



Software Architecture Thinking


Software Architecture: Perspectives on an Emerging Discipline
Mary Shaw David Garlan
Just Enough Software Architecture
George Fairbanks
Software Systems Architecture Second Edition
Nick Rozanski · Eoin Woods
Software Architecture: Foundations, Theory, and Practice
Hassan Gomaa
Software Modeling & Design

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

Software Engineering Institute | Carnegie Mellon University

Architecting Software in a New Age
SEI Webinar
© 2015 Carnegie Mellon University

5

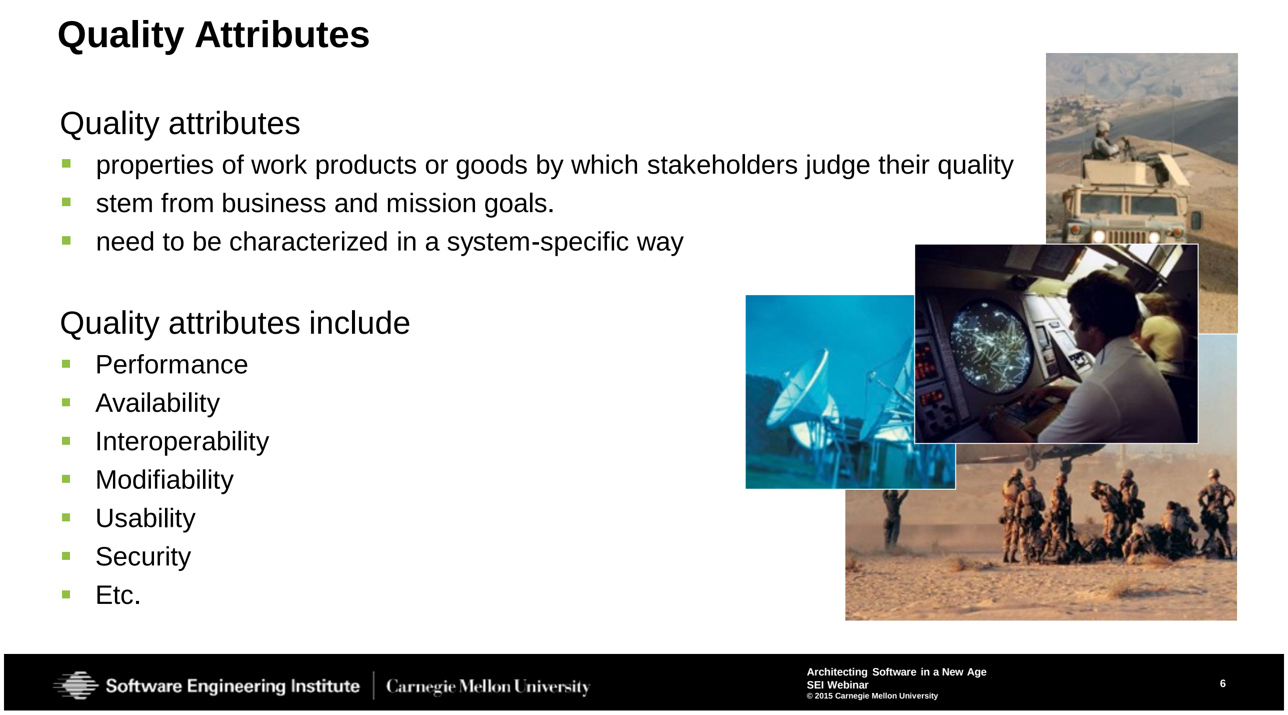
**005 Now, software architecture is not a new concept. In fact, people have been talking about software architecture since the '80s. The whole idea was introduced because systems were becoming much more complex and the behavior we were

expecting of those systems was much
more demanding, and so we needed
to reason about the system at a higher level
of abstraction. So over the years of
people thinking and talking about
software architecture, it's always
been about structure, it's always
been about abstraction and quality
attributes.

On our website, you would find about
150 definitions of software
architecture. The definition we use is
that the software architecture is the set
of structures needed to reason about
the system, which comprise
software elements, the relations
among them, and the properties of
both. But basically one might take
Martin Fowler's expression. He says,
"Architecture is basically the hard
stuff." It allows us to make
engineering tradeoffs. In fact, many
have said that the focus on software
architecture brought engineering
to software systems development
ngineering tradeoffs explicit.

Well, what do we trade off?

Quality Attributes



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.

Software Engineering Institute | Carnegie Mellon University

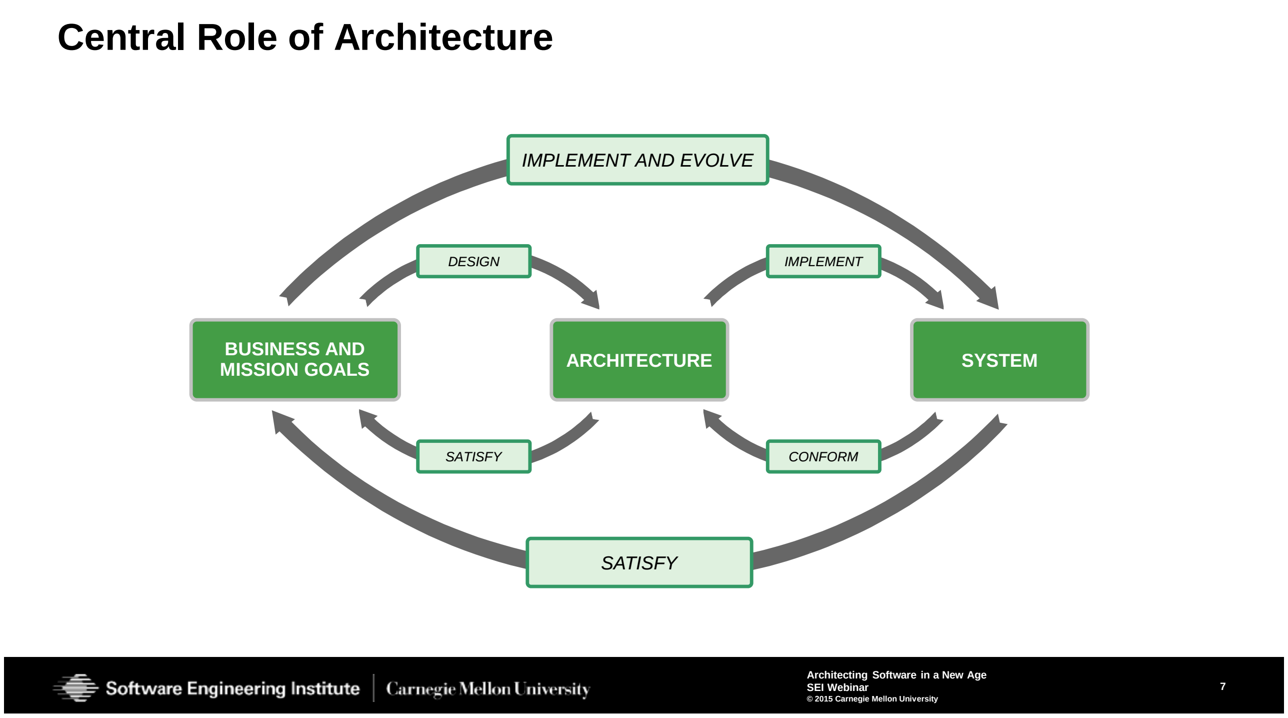
Architecting Software in a New Age
SEI Webinar
© 2015 Carnegie Mellon University

6

**006 We actually are trading off
the functionality-- of course we need
the functionality-- but what else?
What we call quality attributes.
Those are all those properties that
the system needs to have in order
to be assumed to be of high quality,
like performance and interoperability
and modifiability and the like, and the
problem is you can't have all of
these, and so you need to make
tradeoffs. It is the architecture that
actually allows us an abstraction to
make these tradeoffs. And oh, by
the way, these quality attributes need
to be characterized for particular
systems; they are not just arbitrary
"handles" that we select; they're
derived particularly from the business
goals, the mission goals of a system.
So if a business goal is to increase
market share, for example, then

you're going to need a system that's
scalable and that certainly has
demands on the architectural
decisions you make.

Central Role of Architecture



Central Role of Architecture

Software Engineering Institute | Carnegie Mellon University

Architecting Software in a New Age
SEI Webinar
© 2015 Carnegie Mellon University

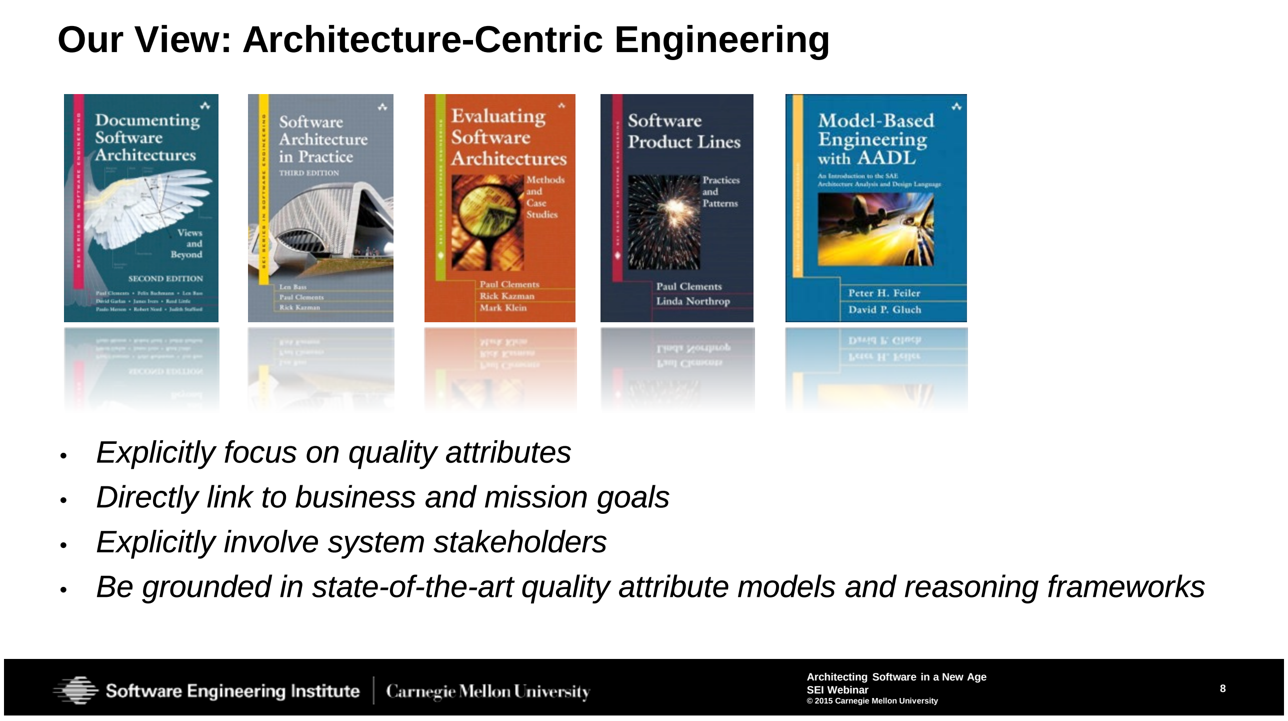
7

**007 So if we think about a
system, we have some business and
mission goals and we'd like to
implement it - to develop our software
to get the system that satisfies us.
But we really have no evidence, no
guarantee about how that's going to
occur, whether it's going to occur,
and what tradeoffs we've made. So
in fact the architecture is that
mechanism, that reasoning abstraction.

Now you might say, "Well, you know,
I use frameworks, I use open source.
I pick my technology stack. I don't
really have an architecture." Well,

you have an architecture, you just might not know the one you have, and you might not be using it to the advantage that you could in order to analyze the system and make appropriate tradeoffs.

Our View: Architecture -Centric Engineering


Documenting Software Architectures
Views and Beyond
SECOND EDITION
Software Architecture in Practice
THIRD EDITION
Len Bass
Paul Clements
Rick Kazman
Evaluating Software Architectures
Methods and Case Studies
Paul Clements
Rick Kazman
Mark Klein
Software Product Lines
Practices and Patterns
Paul Clements
Linda Northrop
Model-Based Engineering with AADL
An Introduction to the SAE Architecture Analysis and Design Language
Peter H. Feiler
David P. Gluch
Software Engineering Institute
Carnegie Mellon University

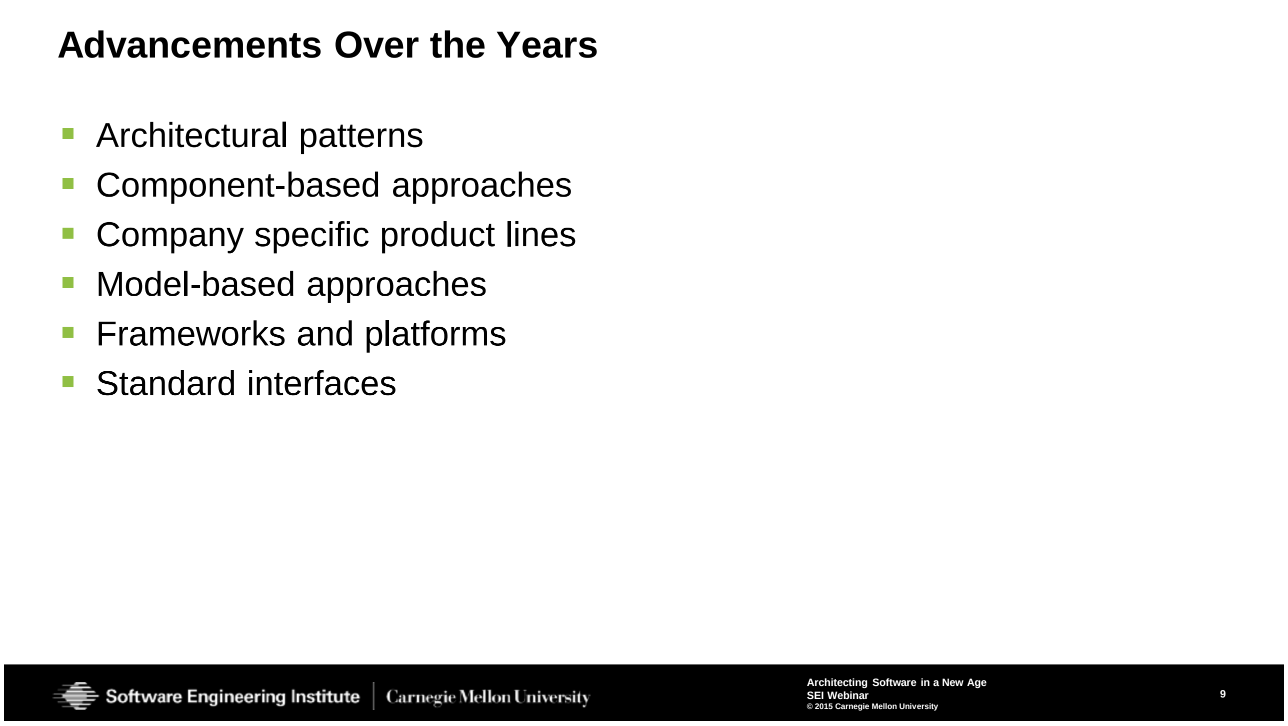
Our View: Architecture-Centric Engineering

- *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*

Architecting Software in a New Age
SEI Webinar
© 2015 Carnegie Mellon University
8

**008 We've been espousing our views on software architecture over the years. Our books all have this in common. They focus on quality attributes, they link to business and mission goals, they have this rooting in interaction with stakeholders -- whether those are supply chain partners, testers, customers, developers, or managers -- and most importantly, they're rooted in quality attribute models -- formal techniques, real-time scheduling techniques, reliability mechanisms, usability frameworks, and the like.

Advancements Over the Years



Advancements Over the Years

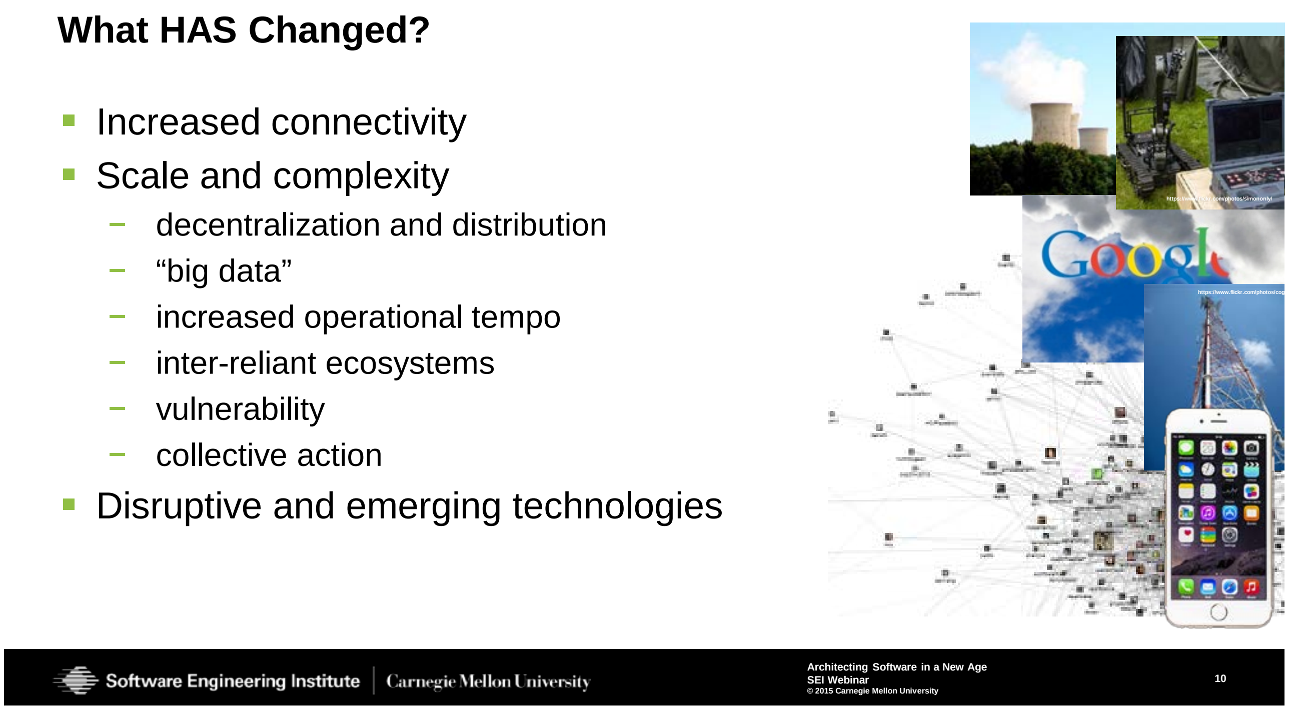
- Architectural patterns
- Component-based approaches
- Company specific product lines
- Model-based approaches
- Frameworks and platforms
- Standard interfaces

Software Engineering Institute | Carnegie Mellon University
Architecting Software in a New Age
SEI Webinar
© 2015 Carnegie Mellon University
9

**009 Over the years a lot has happened in software architecture: architectural patterns and styles that allow us a vocabulary for design and analysis; component-based approaches that take a containment or a container strategy with interfaces that make assumptions about quality attributes; company-specific product lines with architectures that allow us to manage the variation and at the same time capitalize on commonality; and model-based approaches where architectural models are used to generate code; most recently frameworks and platforms that form the basis of ecosystems where the

communication protocols are of
paramount importance. And all of this
has evolved into standard interfaces
that are used in architectures for
families of systems that need to
interoperate.

What HAS Changed?



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

Software Engineering Institute | Carnegie Mellon University

Architecting Software in a New Age
SEI Webinar
© 2015 Carnegie Mellon University

10

**010 But a lot has changed.
Beginning with the web in 1997, one can say
that everything has changed. Everything
has changed in terms of connectivity.
We are an infinitely connected world
where there are not only internet
connections but huge webs of
wireless connectivity and
autonomous devices, and all of this
has grown to actually planetary scale
and complexity that flies in the face
of old hierarchical ways of controlling
systems and developing systems.
We are awash in data. There is an accelerated

tempo: there's an appetite for speed that we see not only in the marketplace. We see it in system Development. We see it in government. And there are all kinds of ecosystems that rely on one another, so one cannot operate in a stovepipe.

Also, if you listened to the U.S. State of the Union message last night, you heard a tremendous focus on cyber defense. Everyone is concerned about vulnerability because we are connected and because we are so exposed and because software is so Prolific. and There is also this whole notion of collective action, where humans, through social media and wireless technology, join forces with computational elements and autonomous elements to form a society that is very new. Against this landscape there's a whole blush of disruptive and emerging technologies.

Technology Trends

Technology Trends

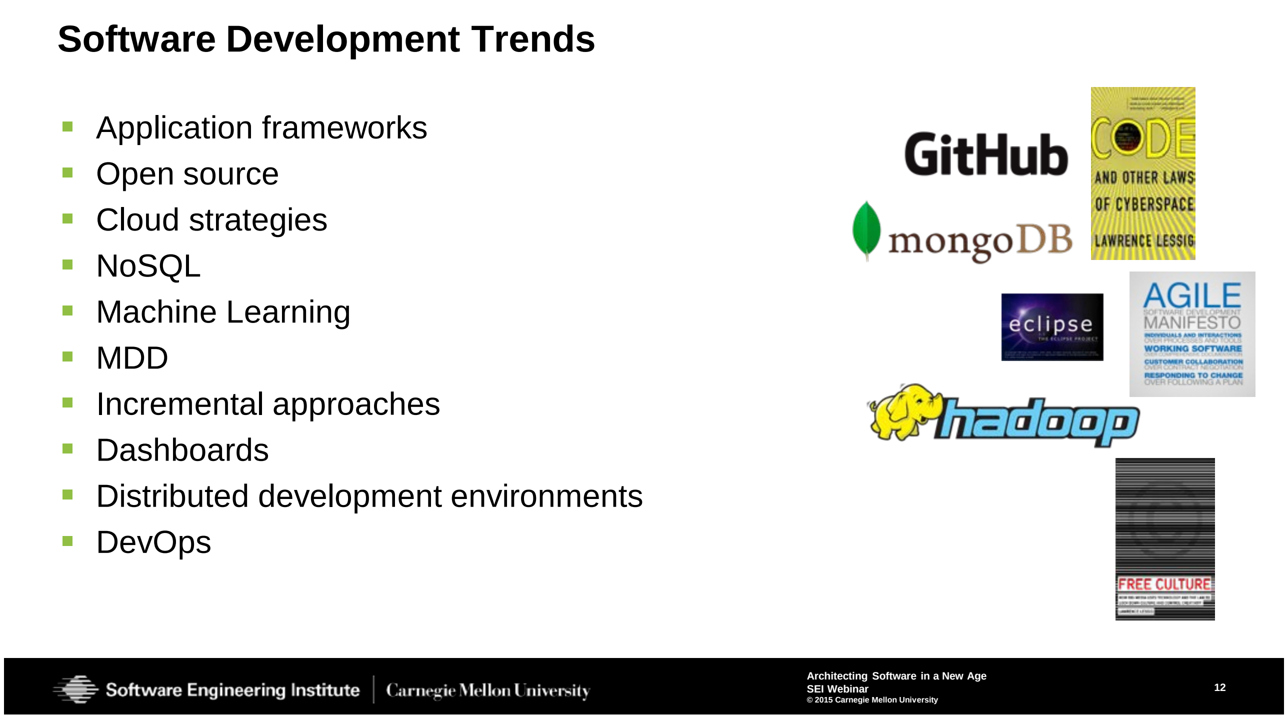
Software Engineering Institute | Carnegie Mellon University

Architecting Software in a New Age
SEI Webinar
© 2015 Carnegie Mellon University

11

**011 They're seductive, they're prolific, from Google Glass to social media to cloud computing to 3D printing.

Software Development Trends



Software Development Trends

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

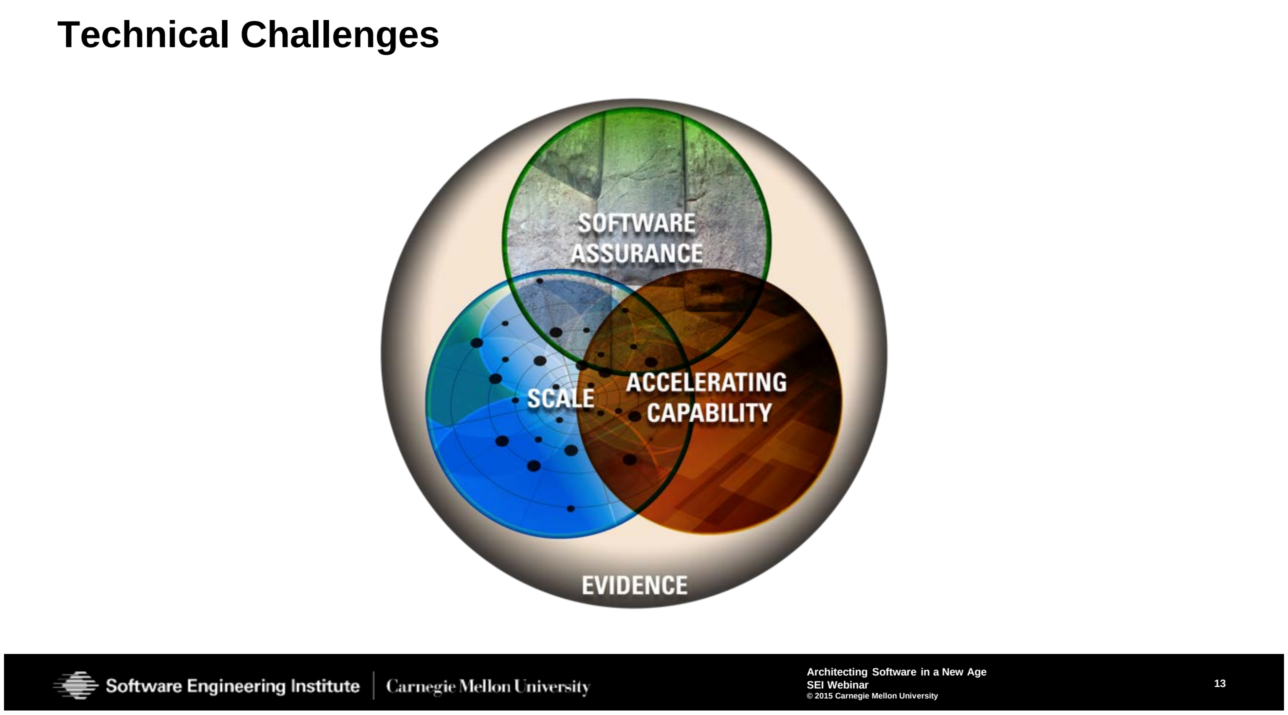
Software Engineering Institute | Carnegie Mellon University

Architecting Software in a New Age
SEI Webinar
© 2015 Carnegie Mellon University

12

**012 And a whole wash of
software development trends-- I'm
sure many of you are engaged in the
open source strategies, NoSQL,
machine learning, and all of the rest
that I have listed -- and so one might
focus on any of these, and we could
spend an afternoon talking about any
of these, any are worthy of discussion in the
Their relationship with software
architecture-- but I'm going to
ratchet it up to a higher level-a more
strategic view.

Technical Challenges



Technical Challenges

Software Engineering Institute | Carnegie Mellon University

Architecting Software in a New Age
SEI Webinar
© 2015 Carnegie Mellon University

13

**013 --Because I believe that the technical challenges can be boiled down to four.

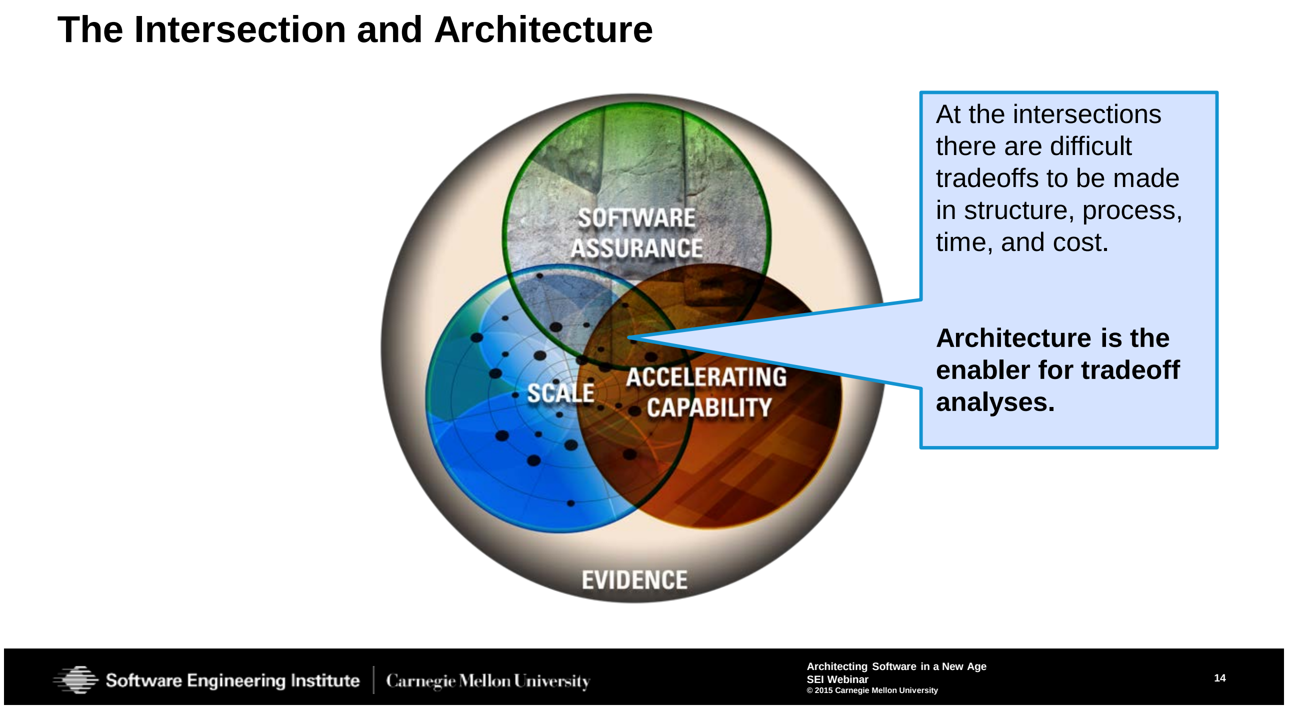
We think about accelerating capability. Everybody is talking about velocity, continuous integration, continuous deployment of many systems where we're getting thousands of releases in very short time periods, but also accelerated development and deployment of what I'd call deliberate systems, planned systems, ones that aren't released multiple hundred times a day; and then also the need to quickly incorporate innovations.
So, new innovations "happen" quickly. We don't want to wait a ten-year period to get those innovations into our code.

Likewise, we have a need for
software assurance. I've talked
about thinking about systems free of
vulnerabilities, but software
assurance means more than that. It
actually means that the system is
going to behave the way we intend it
to behave, to do what it's supposed
to do, and things that it's not
supposed to do don't happen. It
should cost what we expect it to cost
and it should be able to go live when
we time it to go live. Assurance is
about all of those things.

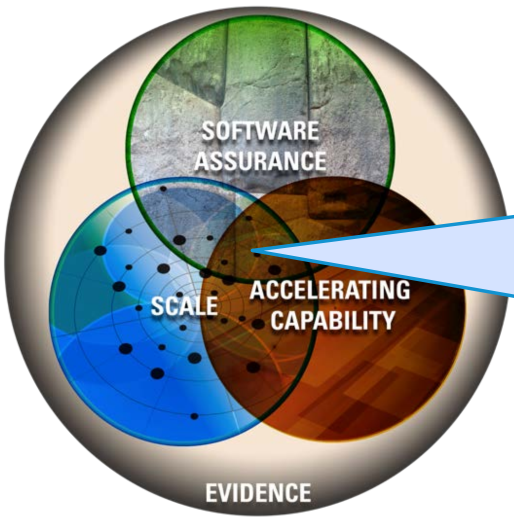
And then there's scale, scale in all
types of manifestations, whether it's
lines of code, number of processors,
number of users. We have
applications now that are very
commonplace, and have 500 million
to billions of users. This is very, very
Different scale - in data, in computational
elements, in the sheer number of people involved in the
system. This is a very different level
of scale than previous systems have
encountered.

And then we are challenged about evidence,
evidence that what you're doing is
going to work for delivering accelerated
capability, software assurance, and address
scale. Greg Wilson from Mozilla, in a
2012 blog, wrote that, "Evidence is
not the plural of anecdote." So we
need much more than a story that
says, "It happened here." We need
scientific evidence, we need proofs,
we need simulations, we need
situations where we have statistically
sound samples that say certain
techniques will work to achieve your
objectives.

The Intersection and Architecture



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.

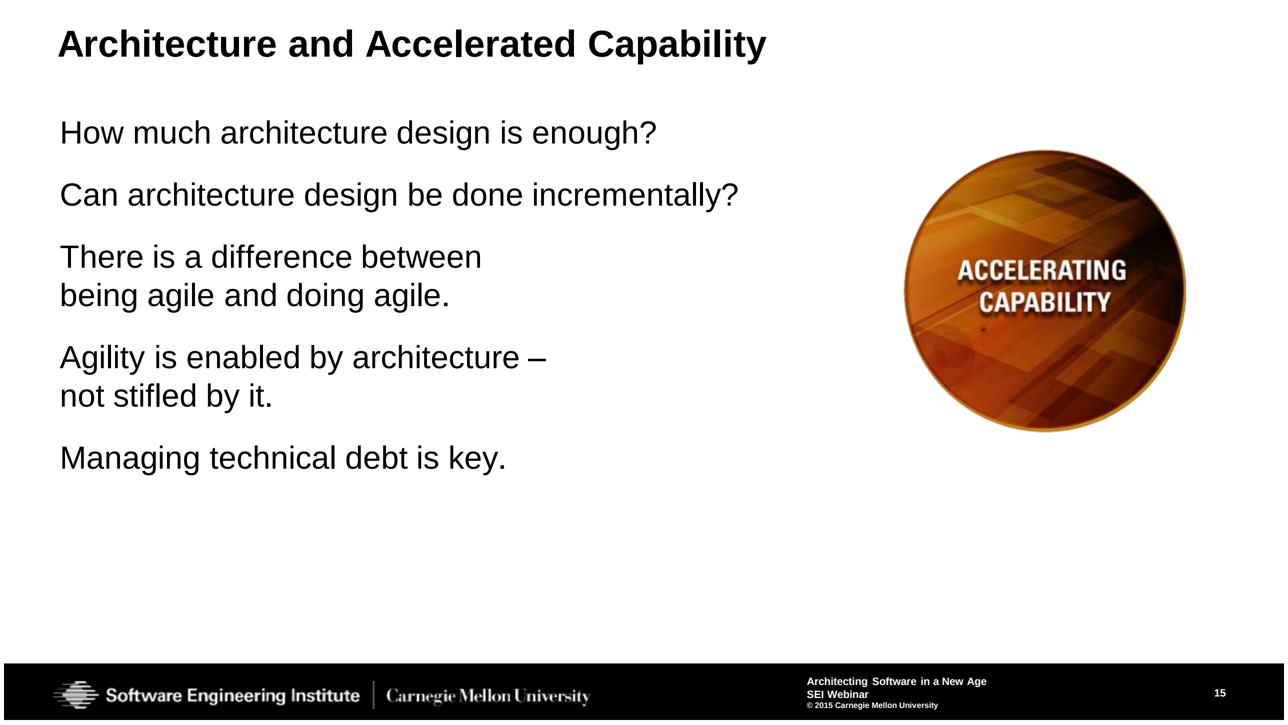
Software Engineering Institute | Carnegie Mellon University

Architecting Software in a New Age
SEI Webinar
© 2015 Carnegie Mellon University

14

**014 Now here's the rub: Each one of these is a challenge, but the intersection-- systems that are fast, at scale, and are assured-- that's the real challenge. And you can't have it all, so there are tradeoffs in these intersections-- tradeoffs in the structure of the systems, the process, the time, and the cost. And, as I mentioned, architecture is that mechanism for making tradeoff analyses, and so I would argue that architecture as the enabler for tradeoff analyses, very important in our brave new world. where we have these challenges.

Architecture and Accelerated Capability



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.

ACCELERATING CAPABILITY

Software Engineering Institute | Carnegie Mellon University

Architecting Software in a New Age
SEI Webinar
© 2015 Carnegie Mellon University

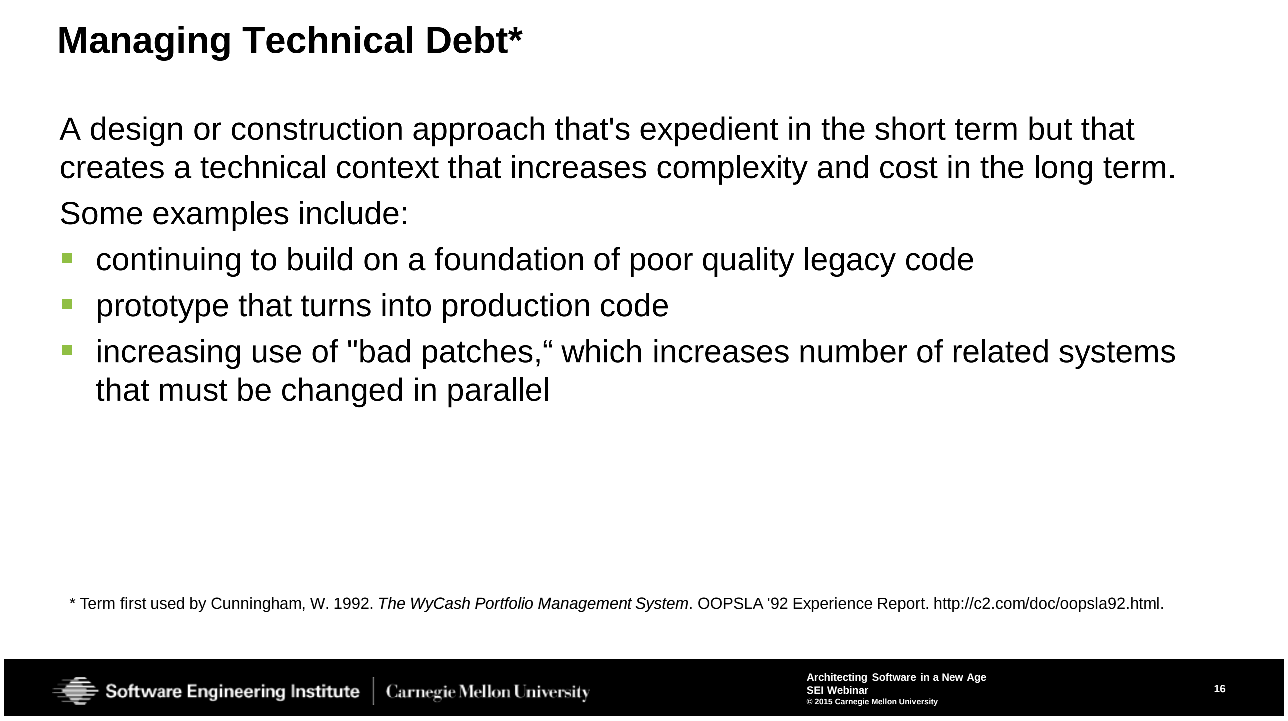
15

**015 So what I'm going to do for
the remainder of this overview is take
a look at each one of these challenges
and the relationship of architecture
to that particular challenge.

So when we think about accelerating
capability-- everybody does want
high velocity, everybody wants to be
agile-- agile with respect to balancing
structure and flexibility, agile with
respect to being able to cause
change and respond to change. We
want to be agile in our software
practice, we want to be agile in our
businesses, because we live in a
dynamic world. Many people
have sort of cast architecture as a big
document-a big document-driven
approach-- and they think about believe ,
"Well, if we focus on the architecture,
there is no way we can be agile."

Jim Highsmith, who is viewed to be one of the founding agilistas, if you will, gave a wonderful keynote at a conference called SATURN in 2010, in which he took on this topic of architecture and agile, and he said there's a difference between being agile and doing agile, and he said agility is enabled by architecture, not stifled by it. But in the process of being agile, in the process of taking an incremental approach to architecture, we have to be careful not to accrue what I'll refer to, and many people refer to, as technical debt.

Managing Technical Debt*



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.

Software Engineering Institute | Carnegie Mellon University

Architecting Software in a New Age
SEI Webinar
© 2015 Carnegie Mellon University

16

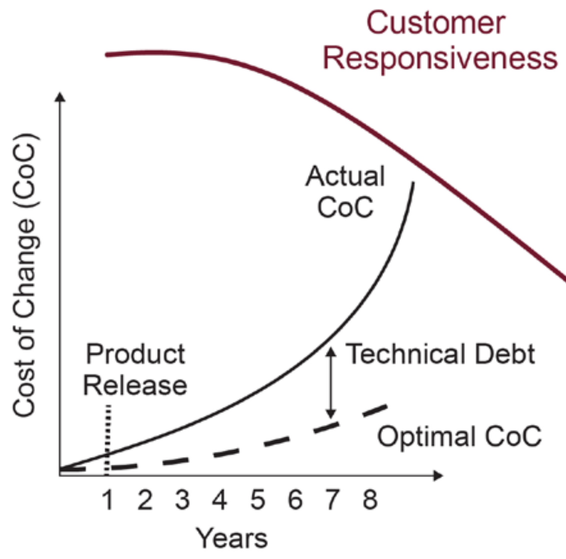
**016 Often when we are fielding a system, we make some decisions, whether they are architectural or code

decisions, to take shortcuts, because
we need to release. We can't afford
the cost of delay. And those
decisions result in increased
complexity in the system, more
difficulty in changing the system later
on, and possibly some quality issues.

Ward Cunningham in 1992
gave this situation a name. He used
the metaphor "technical debt" because
he was trying to justify the need to
refactor a system to some
nontechnical product management
stakeholders. And so one can think
about debt decisions, and we all make these
decisions. One's mother is ill, and
you need to go see your mom, but
you don't have the money for the
plane ticket. So you charge the
money for the plane ticket. Now, as
a mom, I would argue you need to
make that trip. Charge it. But it
goes on your credit card. Now, if at
the end of the month, you don't
manage that charge, and instead you accrue
other charges because you need to
make other decisions and pay for
other sorts of trips or luxury items.
Eventually you get to the point where
your credit card is maxed out, and it
isn't easy for you to respond to
emergencies; it's not easy for you to
purchase that plane ticket. You have
to somehow manage your debt, and
so it is with software.

Technical Debt Impact

Technical Debt Impact



From:

Jim Highsmith

2010

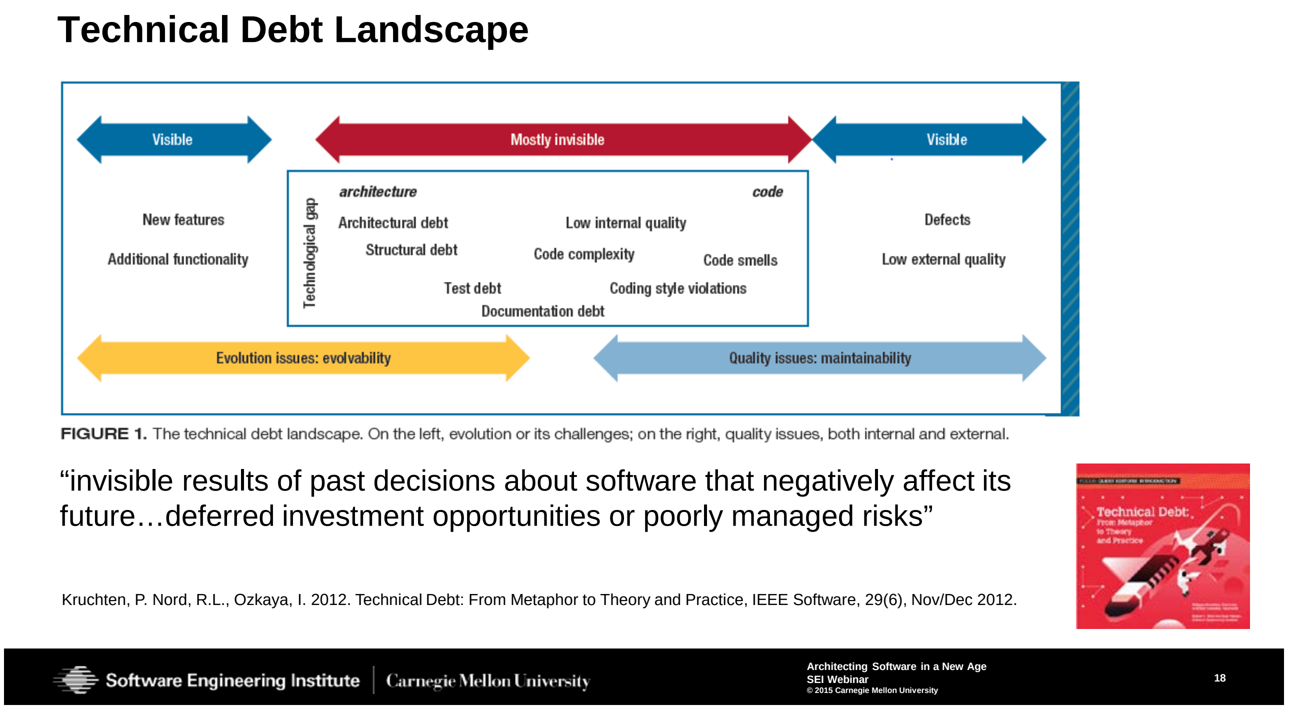
Software Engineering Institute | Carnegie Mellon University

Architecting Software in a New Age
SEI Webinar
© 2015 Carnegie Mellon University

17

**017 In fact, if the technical debt increases, your ability to change the system decreases, and actually the customer responsiveness, that is, your ability to respond to customers, decreases, and this is problematic.

Technical Debt Landscape



Technical Debt Landscape

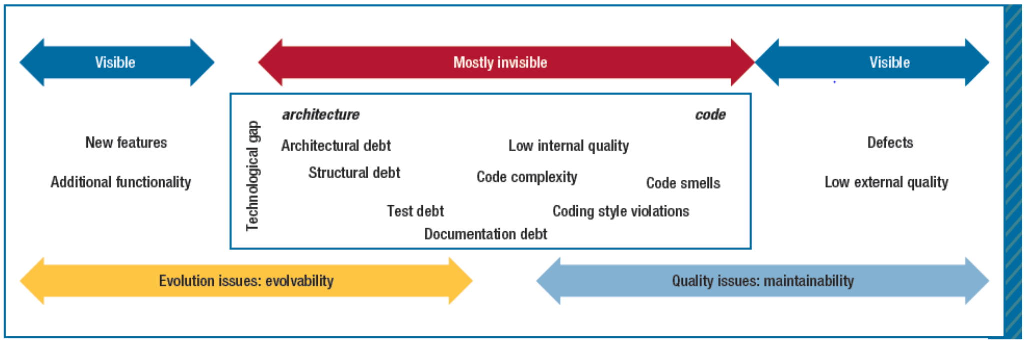


FIGURE 1. The technical debt landscape. On the left, evolution or its challenges; on the right, quality issues, both internal and external.

"invisible results of past decisions about software that negatively affect its future…deferred investment opportunities or poorly managed risks"

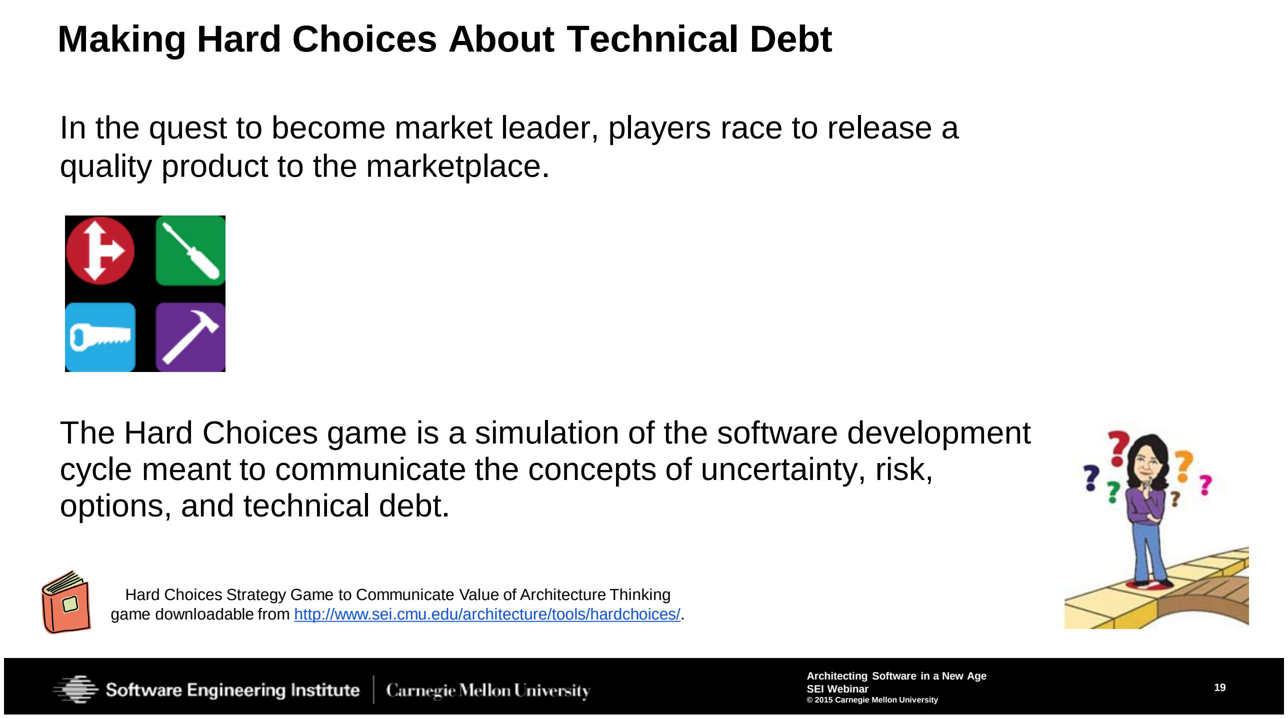
Kruchten, P. Nord, R.L., Ozkaya, I. 2012. Technical Debt: From Metaphor to Theory and Practice, IEEE Software, 29(6), Nov/Dec 2012.

Software Engineering Institute | Carnegie Mellon University

Architecting Software in a New Age
SEI Webinar
© 2015 Carnegie Mellon University

18

**018 So you don't actually see technical debt, you only see evidence of it, and the evidence of it becomes visible when you're trying to add new functionality or new features, or when some defects or low quality becomes apparent. And again, this debt can be architecturally rooted, it can be code rooted. But the whole idea is you need to have a prudent plan to understand the risk you're taking on, to avoid accumulating excessive technical risk, and to actually manage the technical risk. We all need to take some at some time, we all need to accrue some technical debt, because you can't afford the cost of delay, but being prudent about it is what's important.



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/.

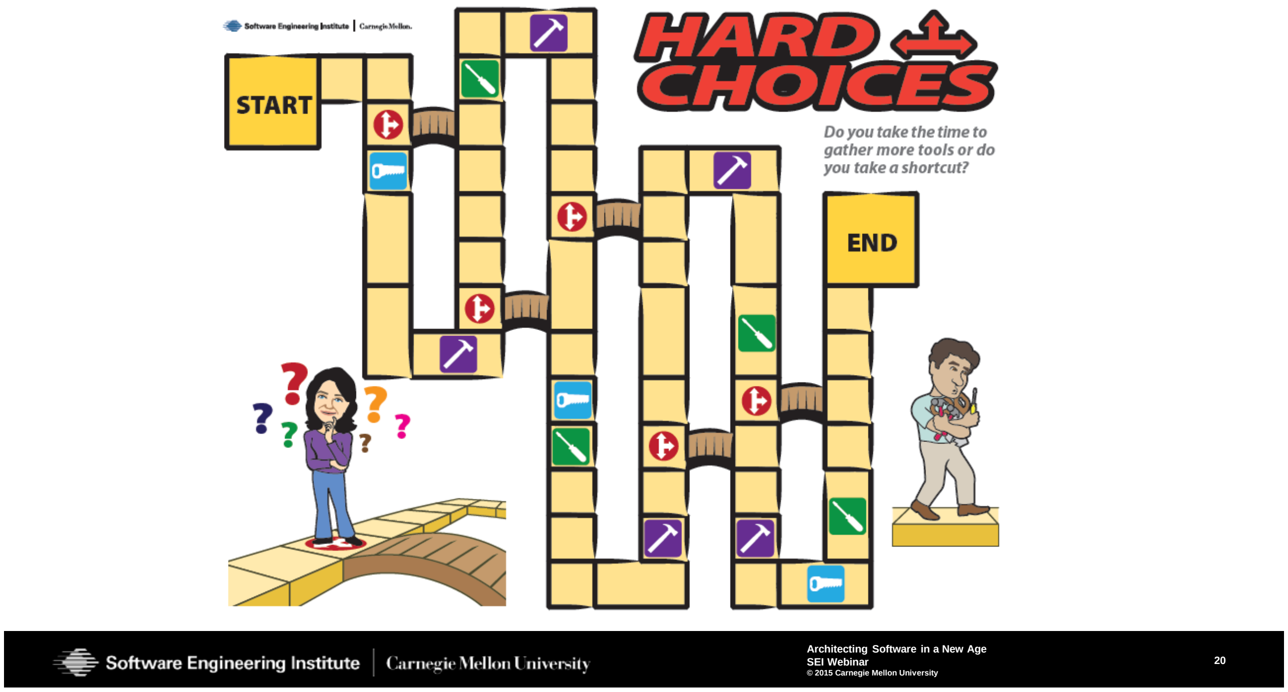
Software Engineering Institute | Carnegie Mellon University

Architecting Software in a New Age
SEI Webinar
© 2015 Carnegie Mellon University

19

**019 So people in the agile space like games, and we at the SEI made a little game in which you can simulate the software development lifecycle and you can understand the concepts of uncertainty, risk, options, and technical debt by playing this game. You can download it from our website--

HARD CHOICES

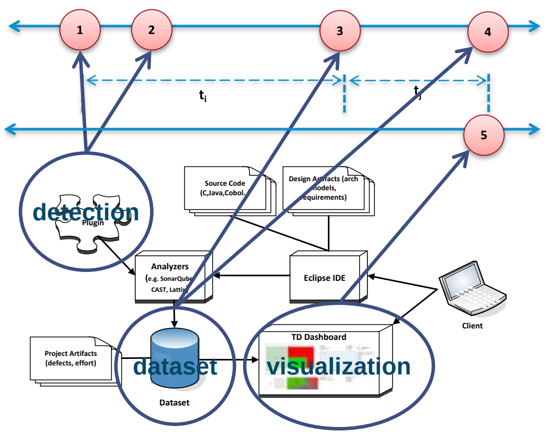

Software Engineering Institute | Carnegie Mellon.
HARD CHOICES
Do you take the time to gather more tools or do you take a shortcut?
START
END
Software Engineering Institute | Carnegie Mellon University
Architecting Software in a New Age
SEI Webinar
© 2015 Carnegie Mellon University
20

**020 --And you can use it in a classroom or in a work setting.

Our Current Research

Our Current Research

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


1
2
3
4
5
t_i
t_j
Source Code (C,Java,Cobol..)
Design Artifacts (arch models, requirements)
detection
Plugin
Analyzers (e.g. SonarQube, CAST, Lattix)
Eclipse IDE
Client
TD Dashboard
Project Artifacts (defects, effort)
dataset
Dataset
visualization

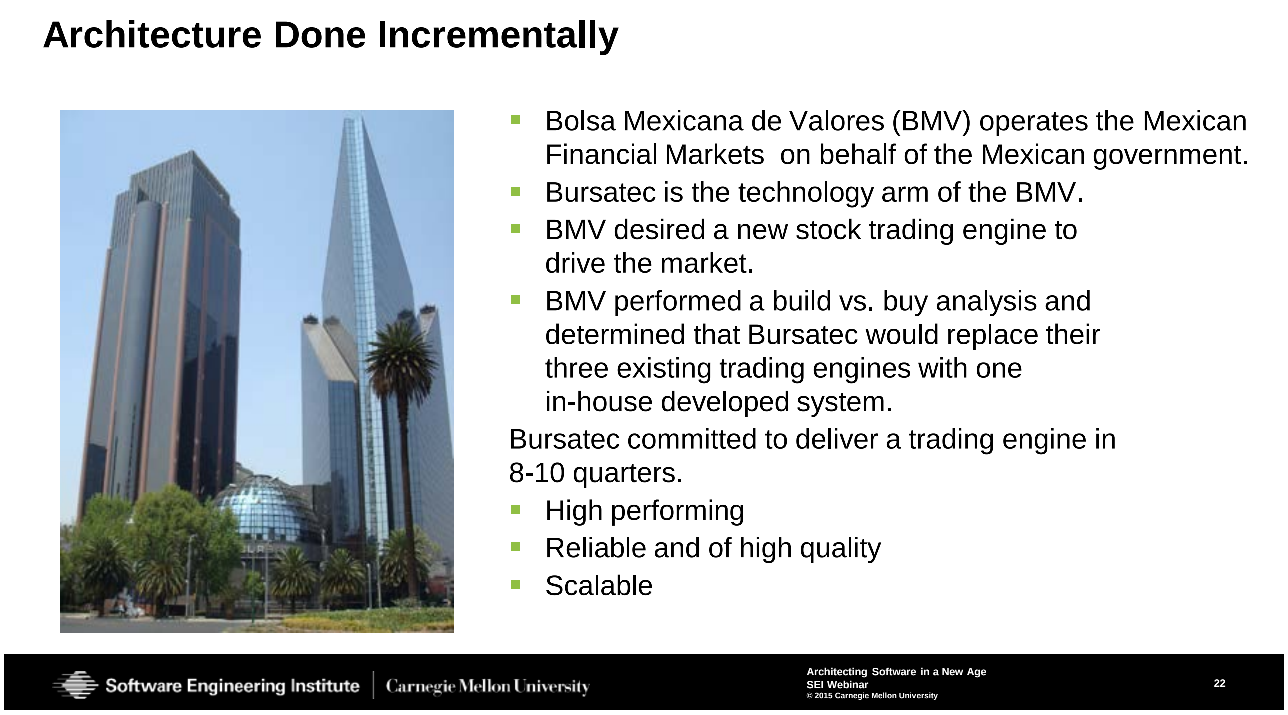
1. time technical debt is incurred
2. time technical debt is recognized
3. time to plan and re-architect
4. time until debt is actually paid-off
5. continuous monitoring

Software Engineering Institute | Carnegie Mellon University

Architecting Software in a New Age
SEI Webinar
© 2015 Carnegie Mellon University

21

**021 Toward a more serious end, we are doing research. We're building a workbench. We're using architectural abstractions, field studies, some conceptual correlation modeling to build a workbench that would allow organizations to see the technical debt they have and to manage the technical debt over time. There is a very large community organized around technical debt right now--people who are building toolsets, people who are building dashboards--and it is a concept that is very important for anyone who takes an incremental approach to system development and needs to understand ramifications of architectural and code decisions that they make in order to avoid the cost of delay.



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

Software Engineering Institute | Carnegie Mellon University

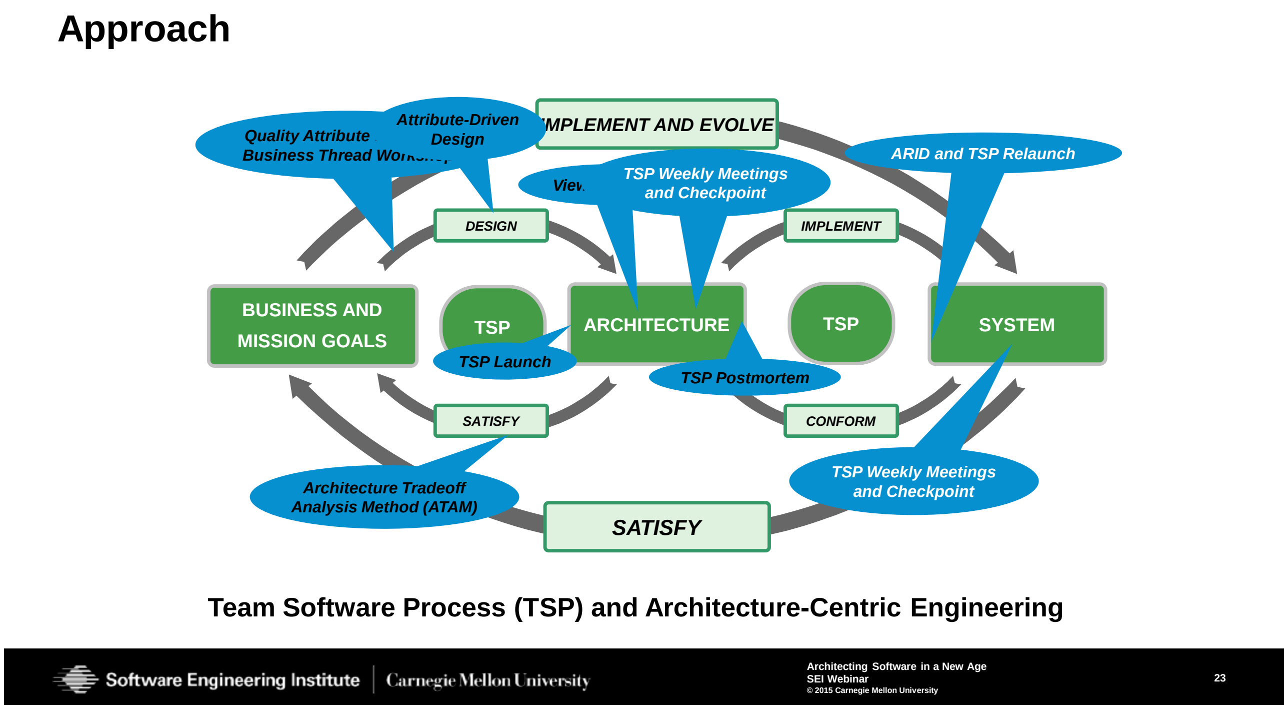
Architecting Software in a New Age
SEI Webinar
© 2015 Carnegie Mellon University

22

**022 Let me give you a little bit of an example. You might say, "Well, can you do incremental development with a focus on architecture? Can you do architect incrementally?" Let me tell you a very quick story, and this story is about the Mexican Stock Exchange.

The Mexican Stock Exchange approached the SEI. MSE -- actually, Bursatec, which is the technology arm of the Mexican Stock Exchange-- was tasked with building a new trading engine, because the decision was made not to buy an off-the-shelf trading engine but to replace the three that they had, and Bursatec committed to deliver this trading engine in a little more than two years. They asserted that it would be high-performing. They wanted it to be a higher-performing-- in other words, faster-- than the previous engines they had, but they also wanted it to be faster than their competitors, faster than NASDAQ, the London Stock Exchange. Such a system has to be highly reliable, as you would understand, and of high quality, and it's got to be scalable because there are peaks and ebbs in stock trading and the system has got to be able to withstand high loads and trading volume.

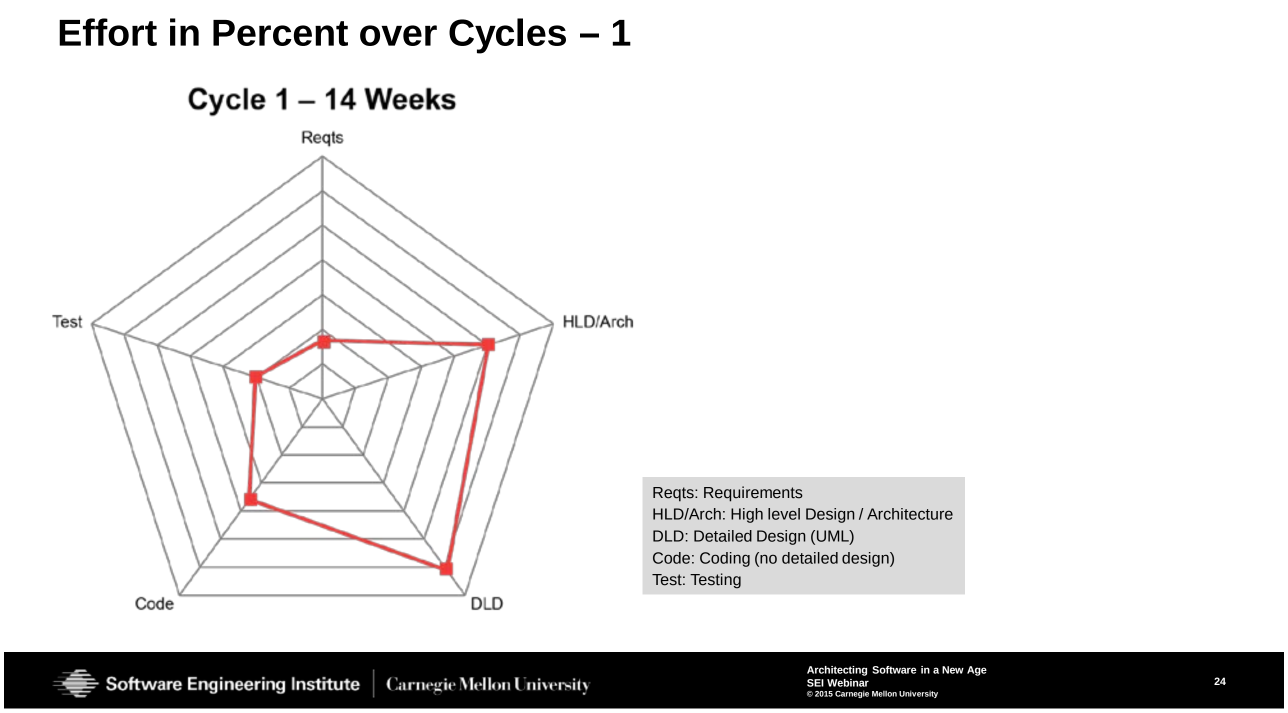
Approach


Approach
Quality Attribute
Business Thread Work
Attribute-Driven Design
MPLEMENT AND EVOLVE
ARID and TSP Relaunch
View
TSP Weekly Meetings and Checkpoint
DESIGN
IMPLEMENT
BUSINESS AND MISSION GOALS
TSP
ARCHITECTURE
TSP
SYSTEM
TSP Launch
TSP Postmortem
SATISFY
CONFORM
Architecture Tradeoff Analysis Method (ATAM)
TSP Weekly Meetings and Checkpoint
SATISFY
Team Software Process (TSP) and Architecture-Centric Engineering
Software Engineering Institute | Carnegie Mellon University
Architecting Software in a New Age
SEI Webinar
© 2015 Carnegie Mellon University
23

**023 The SEI's role was coach. We coached the Bursatec team and we worked with the principles that we espouse, namely our architecture-centric engineering approaches, which you can read

about, and also the Team Software
Process. This is a familiar
diagram that I showed you earlier
about the role of architecture, and
we wickered this with the various SEI
architectural techniques. We used
TSP as the scaffolding for team
management, project management,
and measurement.

Effort in Percent over Cycles – 1



Effort in Percent over Cycles – 1

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

Software Engineering Institute | Carnegie Mellon University
Architecting Software in a New Age
SEI Webinar
© 2015 Carnegie Mellon University
24

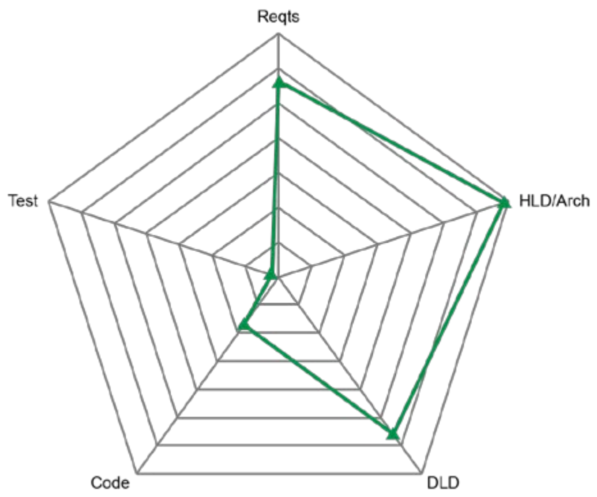
**024 I want to show you,
very quickly, some effort charts, so
that you see that architecture was
actually spread across the six cycles
of development that were used. In
the beginning we did architectural
design and we did some detailed
Design with UML, and we did some prototyping.
We needed to understand what was
possible, and so the prototyping was
important. We did very little

requirements solicitation because we started out basically replacing the engine systems -- stock-trading engine systems -- that actually existed.

Effort in Percent over Cycles – 2

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

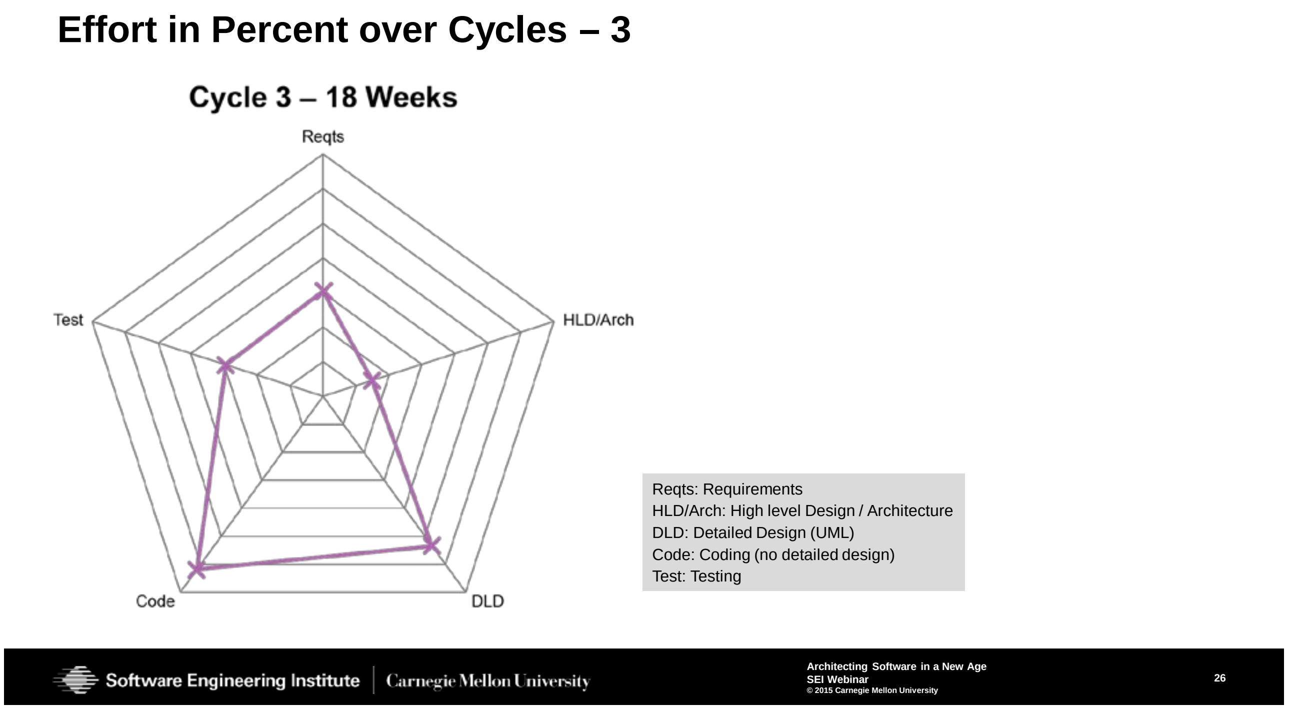
Software Engineering Institute | Carnegie Mellon University

Architecting Software in a New Age
SEI Webinar
© 2015 Carnegie Mellon University

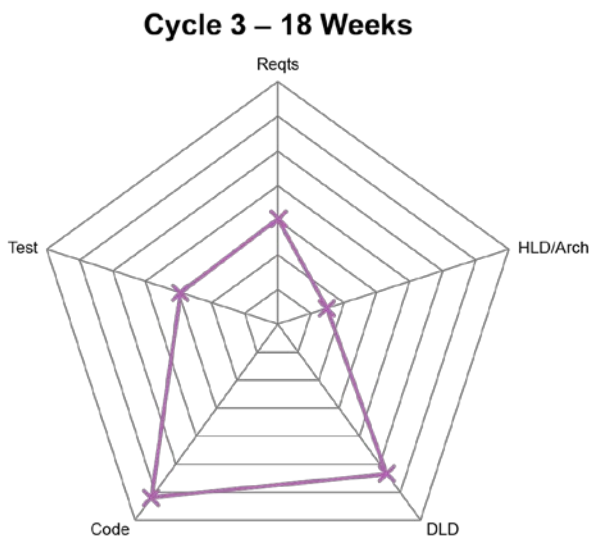
25

**025 In the second cycle there was much more of an architecture emphasis, much more requirements. We did a lot of coding in UML diagrams, and we implemented a skeleton of the system so that we could test out the communication between the various components in the architectural design.

Effort in Percent over Cycles – 3



Effort in Percent over Cycles – 3



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

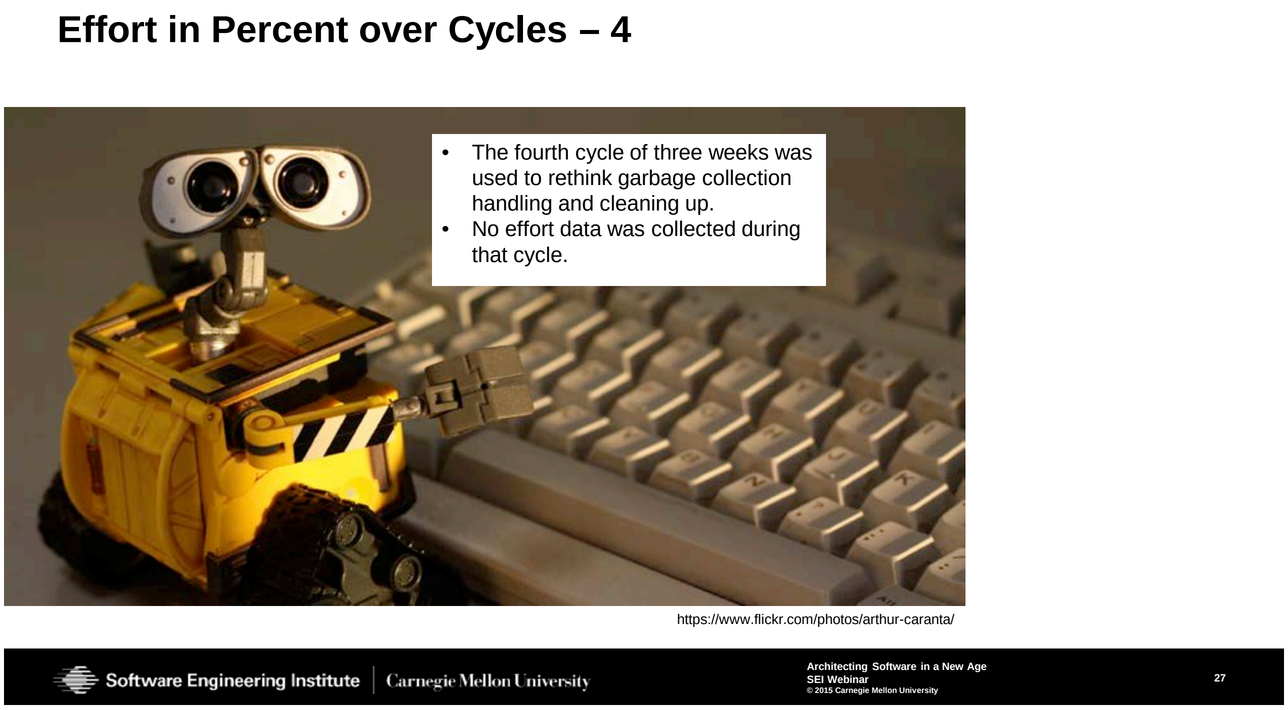
Software Engineering Institute | Carnegie Mellon University

Architecting Software in a New Age
SEI Webinar
© 2015 Carnegie Mellon University

26

**026 In the third cycle, we actually stubbed out the system and did a fair amount of performance testing going end-to-end to see what the throughput would be.

Effort in Percent over Cycles – 4


Effort in Percent over Cycles – 4

- 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

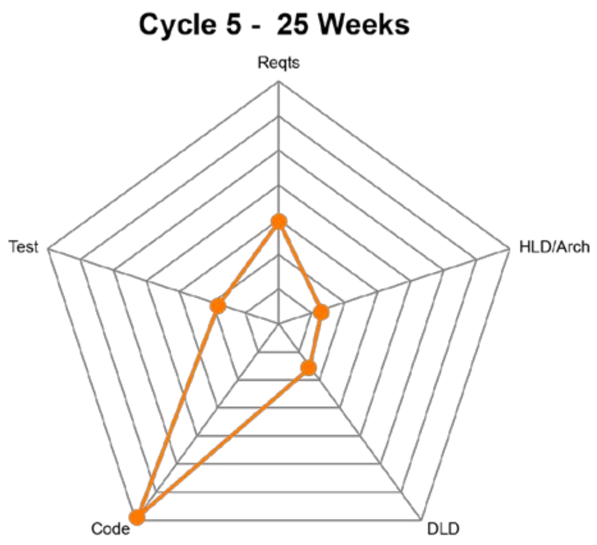
Architecting Software in a New Age
SEI Webinar
© 2015 Carnegie Mellon University

27

**027 In the fourth cycle, we didn't actually keep effort metrics, but we used this to rethink garbage collection and collect cleaned up some things that needed to be done.

Effort in Percent over Cycles – 5

Effort in Percent over Cycles – 5

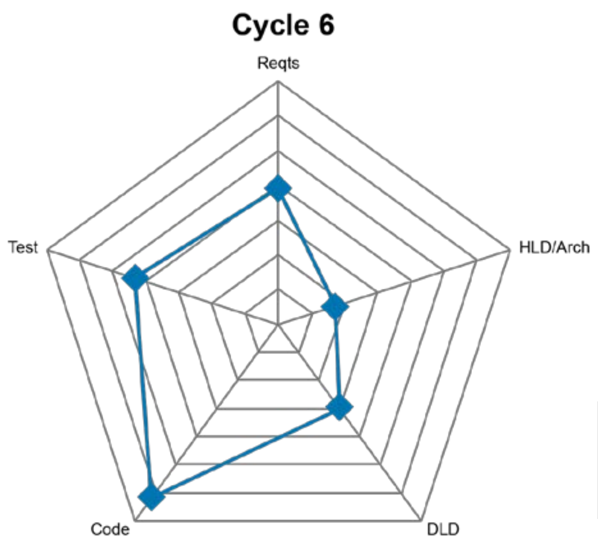


Reqts: Requirements
HLD/Arch: High level Design / Architecture
DLD: Detailed Design (UML)
Code: Coding (no detailed design)
Test: Testing

**028 In the fifth cycle, we actually had a day-trading system-- full functionality for day trading. We developed the testing framework, test cases, and we begin to admit the new requirements for the things that were to enhance this stock-trading engine.

Effort in Percent over Cycles – 6

Effort in Percent over Cycles – 6



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

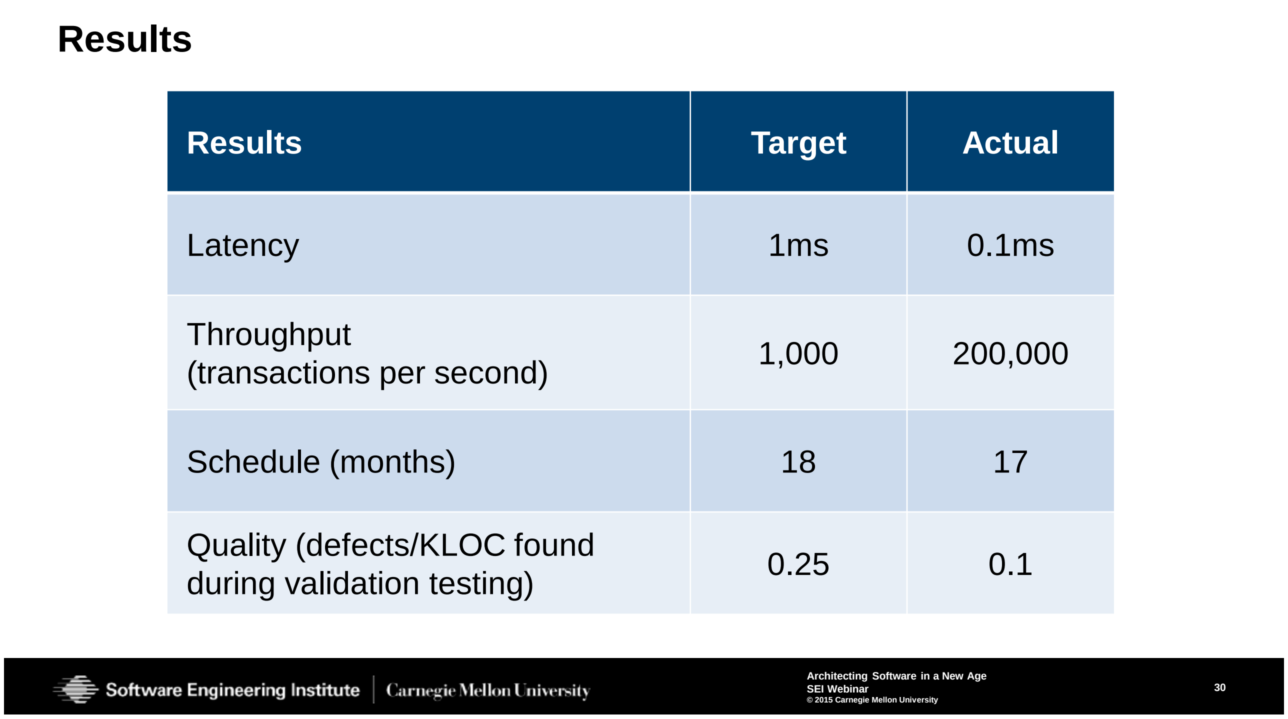
Software Engineering Institute | Carnegie Mellon University

Architecting Software in a New Age
SEI Webinar
© 2015 Carnegie Mellon University

29

**029 And finally, in cycle six, we delivered the complete system, which was not only the day-trading but the maintenance at night, the startup, the actual maintenance and logging that was done. There was still some architecture, still some requirements. If you add up all the effort overall, only 15 percent was spent on testing, which is very unusual for this type of system. development.

Results



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

Software Engineering Institute | Carnegie Mellon University

Architecting Software in a New Age
SEI Webinar
© 2015 Carnegie Mellon University

30

**030 Now, let me give you the punchline. Was Bursatec able to deliver on what they promised? Yes, actually they were able to deliver ahead of time, with hiring no increased staff, and I would add that many of the people we coached were not seasoned architects by any stretch of the imagination. What's really impressive is that the performance was 300 times faster than the actual performance of their earlier stock-trading engine, faster than NASDAQ and faster than the London Stock Exchange.

What I have here before you are numbers that are publicly available, but I know that the situation is actually even better than what's publicly available.

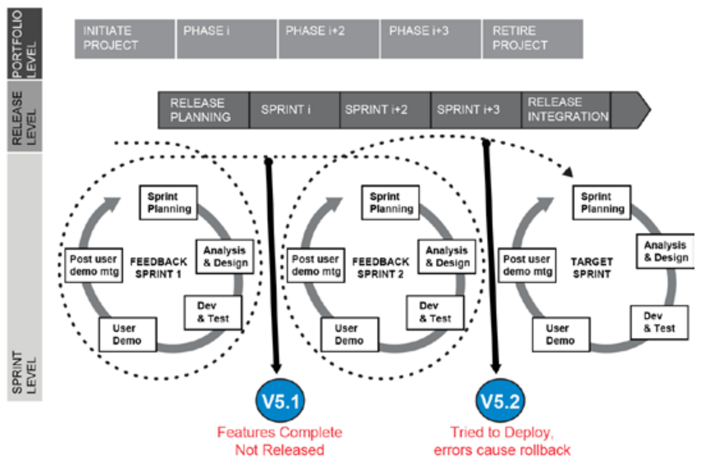
And the quality-- 0.1 bugs per KLOC, which is very unusual. In this type of system we would see 0.5 to 1. Since this system has been launched, there has only been one software problem, and it was easily remedied.

So in fact, the architecture was developed incrementally, the quality of the system was delivered as expected-- in fact exceeded expectations-- it was delivered on time, at cost, and we did it incrementally and used the architecture to perform the analysis that was necessary in order to deliver on the quality attribute agenda.

Deployment Challenges

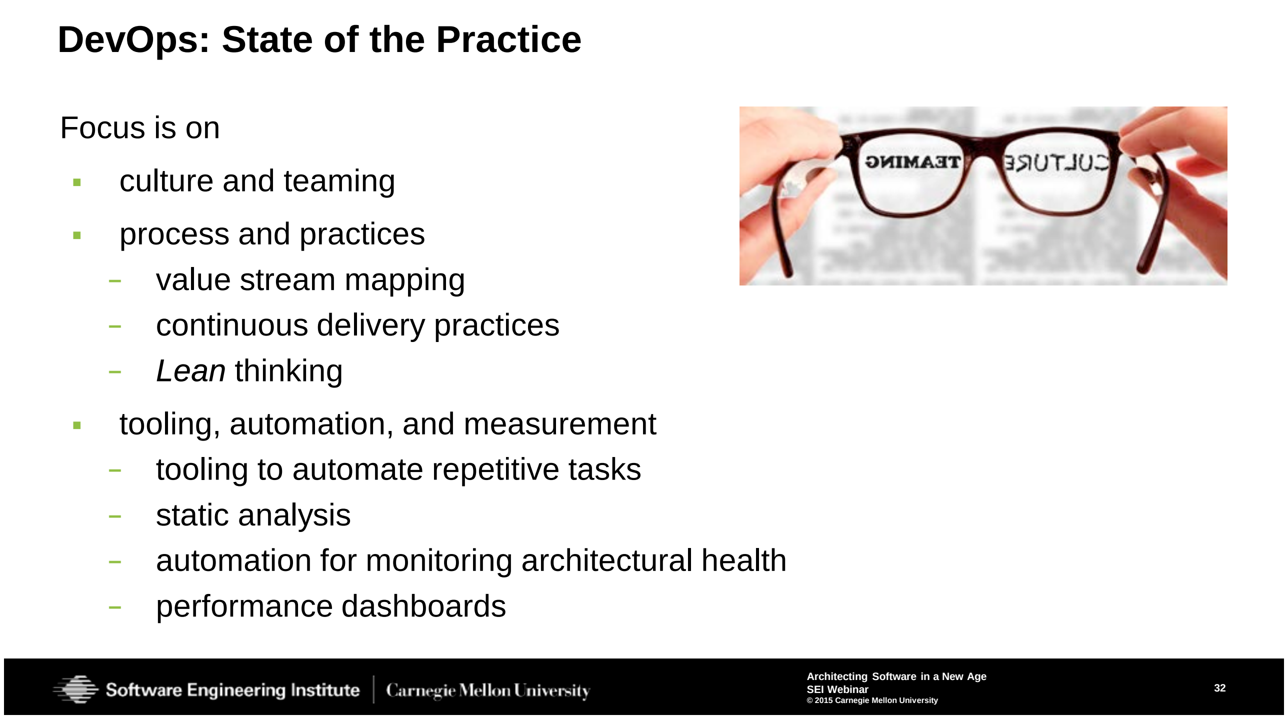
Deployment Challenges

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



**031 Now let me switch topics a little bit. Most people are able to handle incremental development, and a lot of people are incorporating architecture techniques in their incremental development. If you haven't, I hope I've given you some insights into how to do that. But let me switch to DevOps, because the focus now is on deployment. Even though people can handle incremental development, there are problems with deployment, and the mantra these days is velocity, continuous integration, continuous deployment-- and so we need to understand how to pick up where our agile development left off.

DevOps : State of the Practice



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

Software Engineering Institute | Carnegie Mellon University

Architecting Software in a New Age
SEI Webinar
© 2015 Carnegie Mellon University

32

**032 The state of the practice in DevOps is focusing largely on culture and teaming. There are a lot of processes that are used to monitor, status checks, tooling, dashboards--

a tremendous amount of tooling to understand the architectural health, to understand the runtime performance, the operational performance of the system, and all of this is working quite well, and there are a number of organizations who have really gotten on the DevOps bandwagon and are doing well with it.

Architecture and DevOps

Architecture and DevOps

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

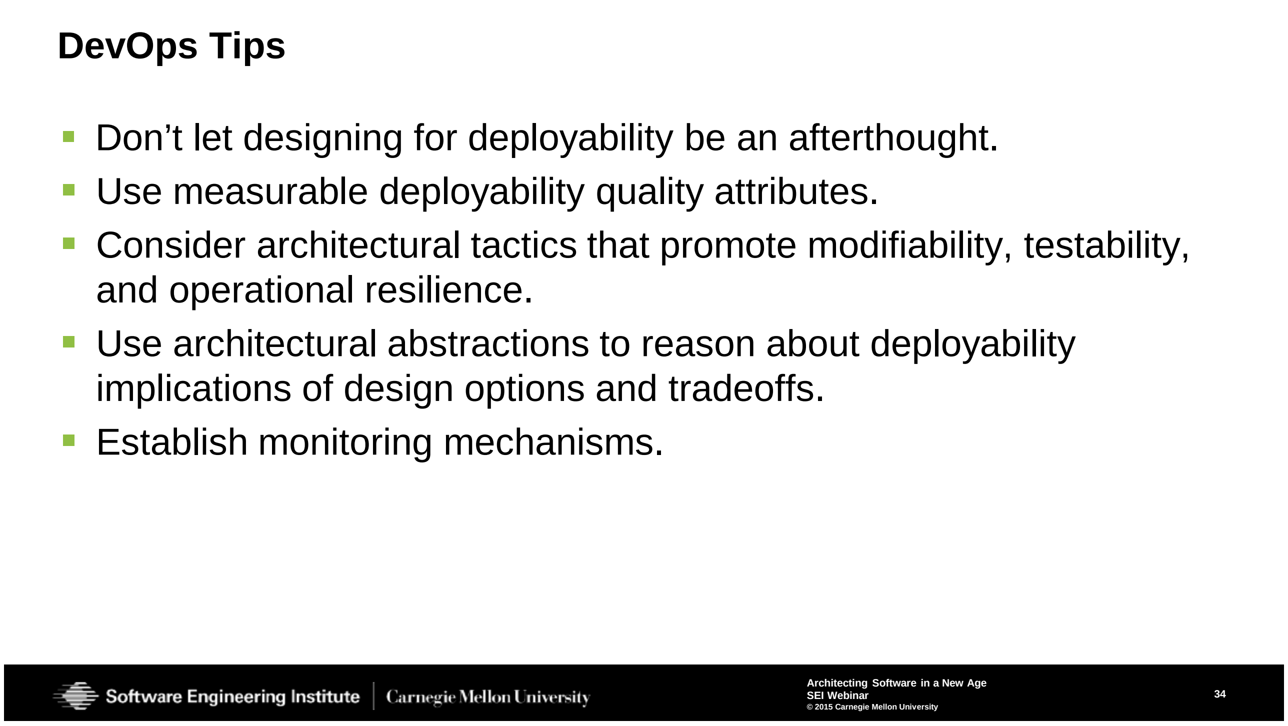
Software Engineering Institute | Carnegie Mellon University

Architecting Software in a New Age
SEI Webinar
© 2015 Carnegie Mellon University

33

**033 And yet you can still be blindsided. You can be blindsided by frameworks that you've chosen or tech stacks that you've chosen for your system that don't allow you the deployability that you need for your continuous integration and continuous deployment, and many of those decisions are not a matter of refactoring, but would require that you would actually have to replace hardware or would require substantial and topological changes in the software. So one of the things you need to focus on, if you are interested in a successful DevOps strategy, is to think about it early on-

DevOps Tips



DevOps Tips

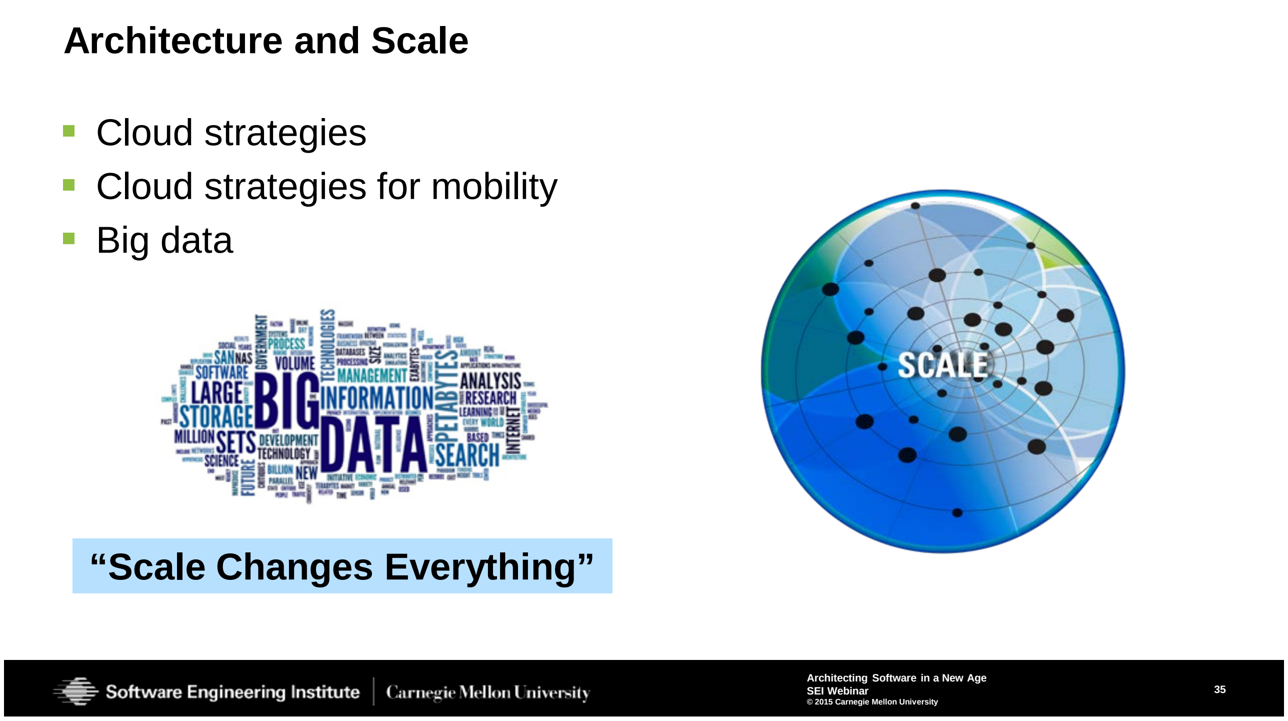
- Don't let designing for deployability be an afterthought.
- Use measurable deployability quality attributes.
- Consider architectural tactics that promote modifiability, testability, and operational resilience.
- Use architectural abstractions to reason about deployability implications of design options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

Software Engineering Institute | Carnegie Mellon University
Architecting Software in a New Age
SEI Webinar
© 2015 Carnegie Mellon University
34

**034 --To think about deployability at the architecture stage, to understand that in order to have a system that will meet your deployability expectations, you are going to have to think about what your deployability scenarios are and pick architectural tactics and make architectural decisions that will actually support testability, and deployability. You need to think about wiring into your system monitoring mechanisms so that during runtime you can actually monitor the health of the system.

These are architectural decisions, these are architectural strategies. It's too late to think about these once you have already developed the system and move into the operation phase. So you have to blend the development and operation, and architecture has got to be part of the conversation.

Architecture and Scale



Architecture and Scale

- Cloud strategies
- Cloud strategies for mobility
- Big data

BIG DATA
SCALE

"Scale Changes Everything"

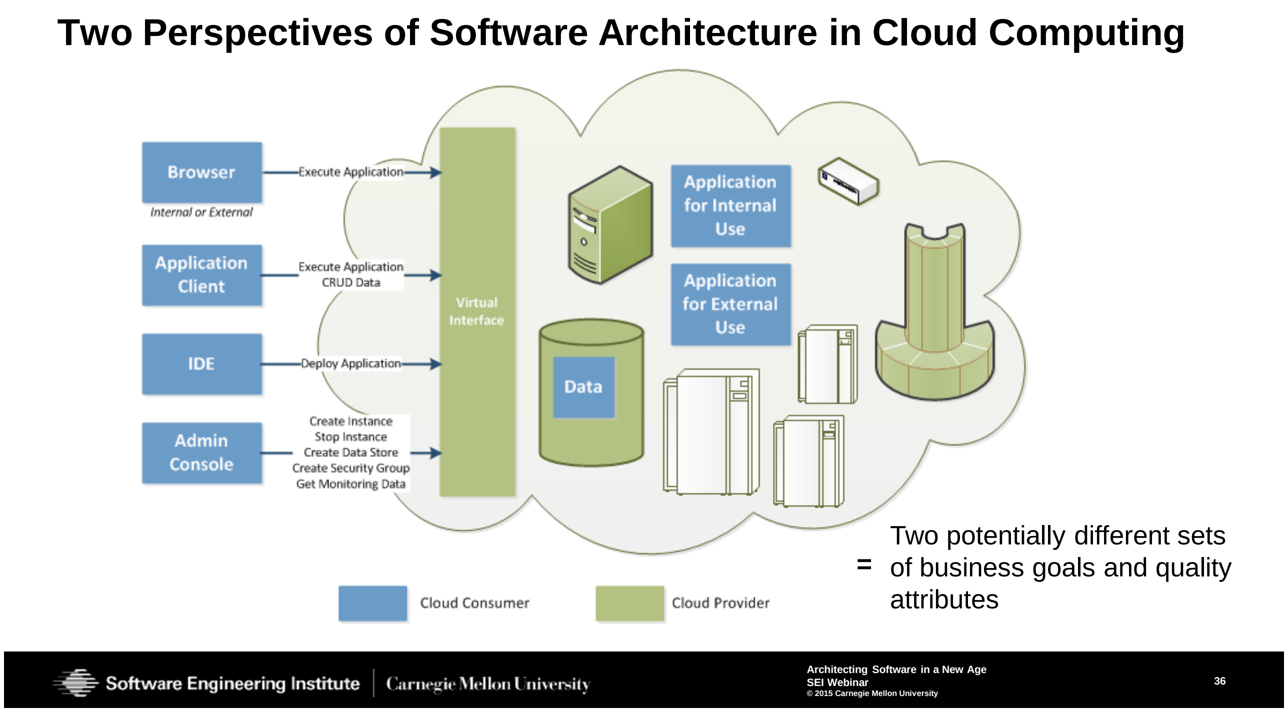
Software Engineering Institute | Carnegie Mellon University

Architecting Software in a New Age
SEI Webinar
© 2015 Carnegie Mellon University

35

**035 Now let's move to the second challenge, which is scale. I've given lots of invited talks about scale, and in fact one of the titles that I've used over and over is "Scale Changes Everything," which is not a hyperbole, because in reality it does. When you have systems of the scale that we're talking about-- this sort of planetary scale involving humans and autonomous entities and computational devices-- we see that the situation requires distribution-- distribution of development, distribution of data, distribution of evolution. We see heterogeneous software and hardware. We see unprecedented connections, unprecedented use of systems. There are commonly documented challenges, and I could talk about scale for a long time, and have. But I'm going to focus on three issues related to scale, and in particular related to architecture and scale: cloud strategies, cloud strategies for mobility, and big data.

Two Perspectives of Software Architecture in Cloud Computing

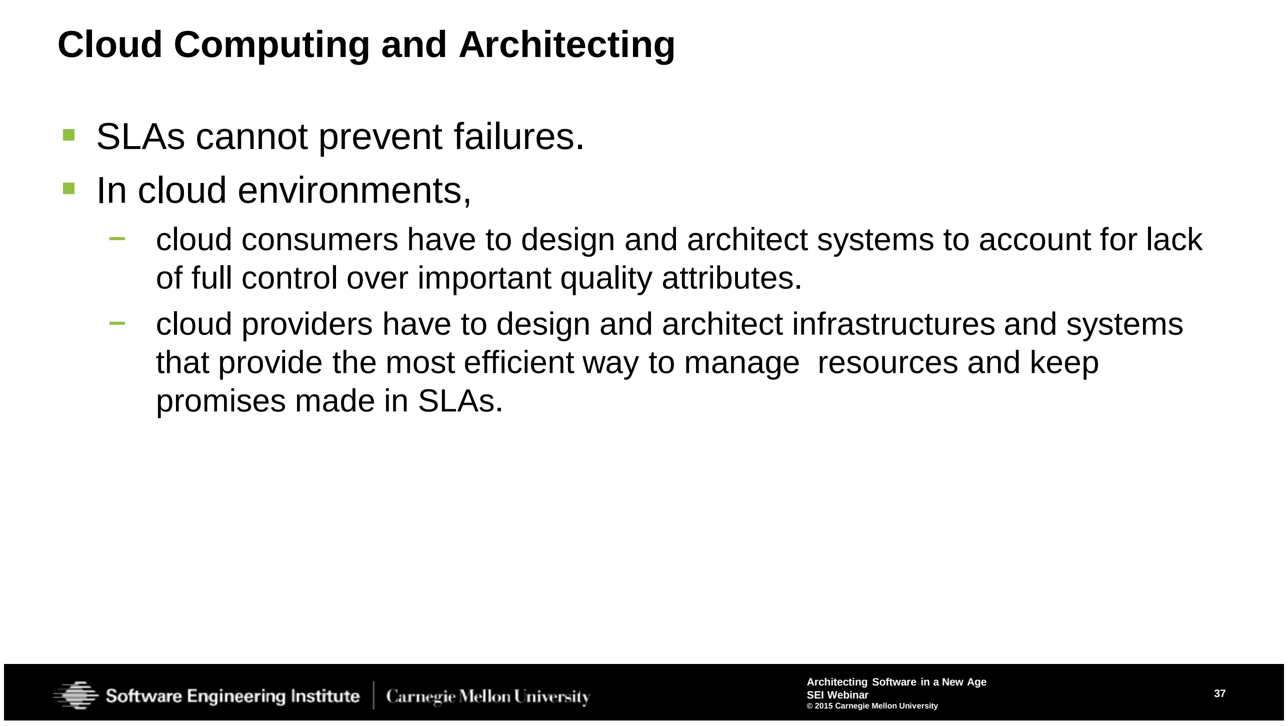


Two Perspectives of Software Architecture in Cloud Computing

= Two potentially different sets of business goals and quality attributes

**036 So when we think about cloud computing-- and almost everyone is using cloud computing because we have these warehouses of computational capability that we can tap into through web services--we have to understand that the cloud provider and the cloud user have potentially very different business goals, and because the quality attributes are driven from the business goals, potentially different quality attributes.

Cloud Computing and Architecting



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.

Software Engineering Institute | Carnegie Mellon University

Architecting Software in a New Age
SEI Webinar
© 2015 Carnegie Mellon University

37

**037 One might argue that, "Okay, but we have service-level agreements so that all is copacetic between the cloud provider and the cloud consumer." However, the service-level agreements are a minimum, and they can't prevent failures. So what we have to do as a cloud consumer is architect our

systems knowing that we don't have
full control over many of the
important quality attributes.

Specifically, cloud providers are going
to optimize on reliability-- they want
to provide consistent computational
power to the cloud user. They want
to provide a level of acceptable
performance, which is usually
articulated in the SLA. At the same
time, their tradeoffs have to do with
energy efficiency. They're paying for
electricity and cooling of these
massive server farms, and this is a
nontrivial issue that factors directly
into their business goals.

So you're the cloud
consumer, and you like the reliability
you're getting and the performance,
but now you say, "But I'm really
concerned about security. I want to
make sure that nobody hacks into
what I have on the cloud." Whoa.
That's a tradeoff. And you need to
think, when you're architecting your
system, how you're going to
compensate for what the cloud
provider is not going to provide to
you - what is not necessarily
articulated in the SLA. You need
to be smart about cloud computing.

Ian Gorton, who is one of our
colleagues, has said that, "Cloud
computing allows us to fail cheaper
and faster than we were able to
before." So, very important to think carefully
about architecture when you're using
cloud strategies.

Mobile Device Trends

Mobile Device Trends

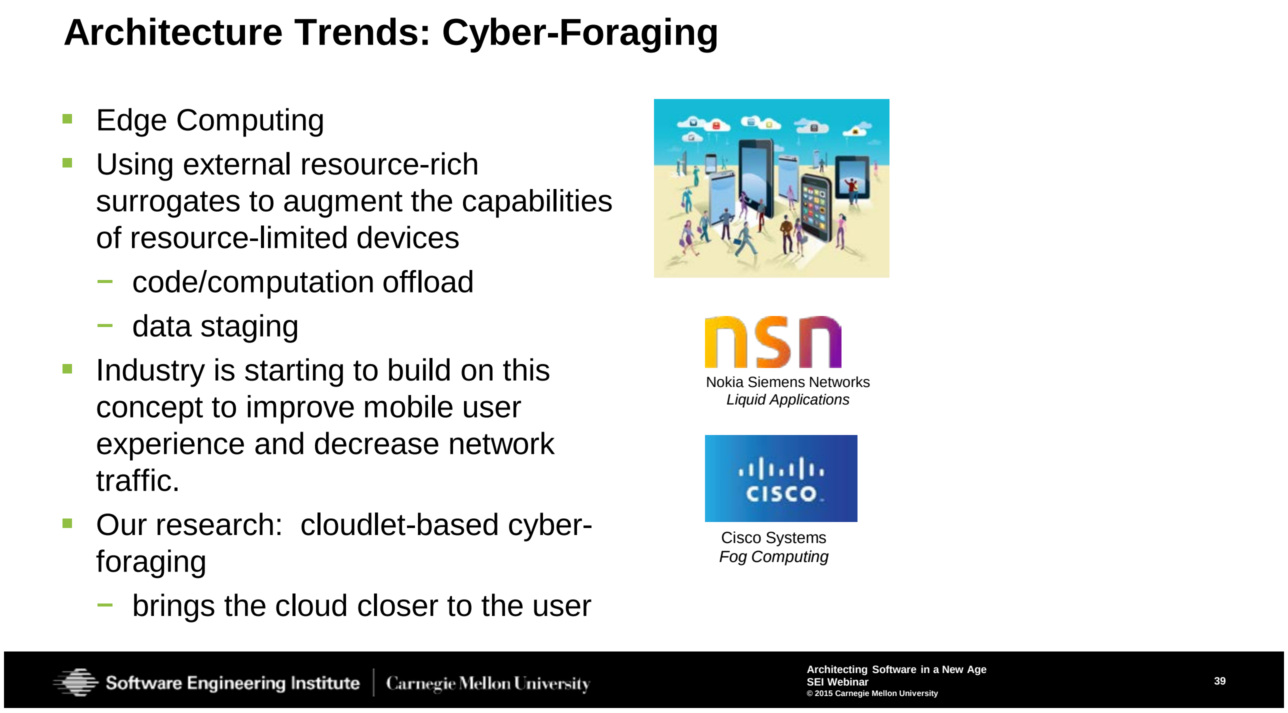
Software Engineering Institute | Carnegie Mellon University

Architecting Software in a New Age
SEI Webinar
© 2015 Carnegie Mellon University

38

**038 The world has also moved to mobile devices. There is a huge mobile device trend, and we're now using our smartphones and our tablets as ways to connect with the internet, as ways to connect with social media, as ways to control our appliances and devices. We have come to expect a level of performance, because we're used to laptops. So we think our tablets and our mobile phones are going to have the same sort of capability as our laptops. But the reality is: they are limited in size, they are limited in battery power, and there is some variance in the latency between your mobile device and the cloud from which you are gathering your enterprise data. And if you're using this mobile device for something that is critically important, like triaging some sort of health situation in an emergency event, you want a little more reliability than what you can typically get with mobile devices today.

Architecture Trends: Cyber-Foraging



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

Nokia Siemens Networks
Liquid Applications

Cisco Systems
Fog Computing

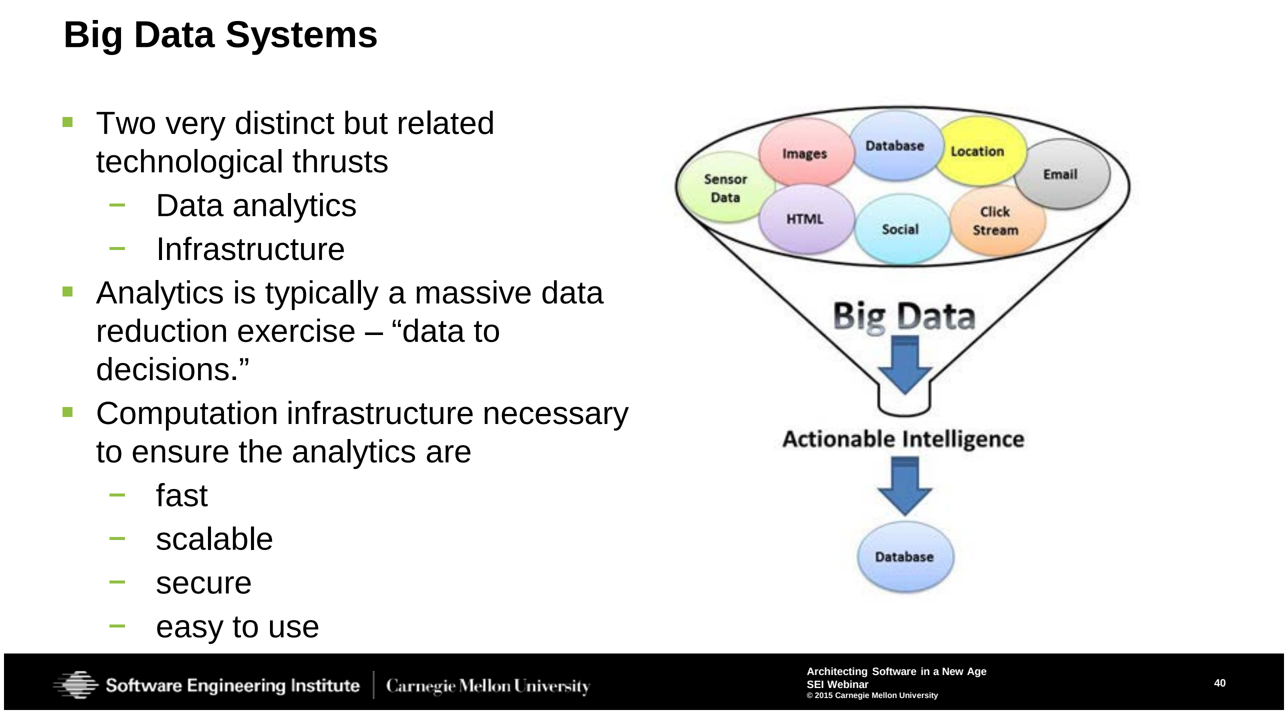
Software Engineering Institute | Carnegie Mellon University
Architecting Software in a New Age
SEI Webinar
© 2015 Carnegie Mellon University
39

**039 So, what are some of the architecture trends? Well, architecture trends today are in what many people are calling cyber-foraging. One term is edge computing, where we push the computation, the data, the analysis to the very edge of the network, to the people who are using these mobile devices. This sort of computing we find is necessary for early responders, for soldiers in war situations, where they have a mobile device, a handheld or a tablet. And they need to do some analysis, they need computational power, and they need data staging, and they need a way

to get data from the cloud.
So we appeal to some surrogates,
maybe some nearly laptops, that will
allow us to augment the capabilities
of the mobile device so we can
offload some of the computation,
or do some of the data staging
there so the data coming from the
cloud can in transit be hosted in this
surrogate -- say, in the laptop -- and then
move to the mobile device.

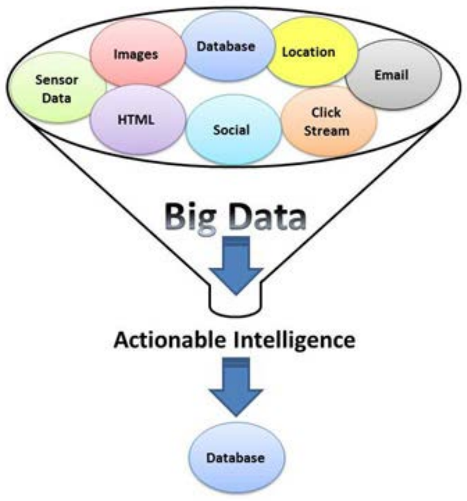
Industry is starting to build on this
concept to improve mobile user
experience and decrease network
traffic. Our experience and our
research is in cloudlets. Cloudlets are
Discoverable, virtual machine-based forward-
deployed servers. They're located a
single hop away from the mobile
device. What we've been able to
do is allow the mobile device to
operate in disconnected mode. Where,
for example in a war situation when the
mobile device totally disconnects
from the network and from the
enterprise, but the cloudlet is able to
be provisioned so that it provides the
continuity that's needed in these
situations that are high stress and
high criticality.

Big Data Systems



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



Software Engineering Institute | Carnegie Mellon University

Architecting Software in a New Age
SEI Webinar
© 2015 Carnegie Mellon University

40

**040 One other topic related to scale is big data. I would be really remiss if I didn't talk about big data because everybody's talking about big data. But when we talk about big data there are tradeoffs, from the network on down. And again-- you guessed it-- architecture is a very good abstraction for us to reason about those tradeoffs.

Now, a little bit about big data. There are really two distinct but related technology thrusts. There is the data analytics, how you want to analyze the data. The data is usually Heterogeneous -- it comes from lots of different sources, and is big volume but low information content -- and what we want to be able to do is analyze so that it's high information content and low volume. So people

use a combination of machine
learning and static analysis.
There are many algorithms that
people are employing for data
analytics.

The flipside is the infrastructure. You need infrastructure to house this big data, you need infrastructure to actually perform the analysis and the computation; and that infrastructure has got to ensure that the analytics are fast, they're scalable, they're secure, and they're easy to use. So basically you've got a big filtering problem.

Big Data – State of the Practice " The problem is not solved"

Big Data – State of the Practice "The problem is not solved"

Building scalable, assured big data systems is hard.

HealthCare.gov
Take health care into your own hands
NETFLIX
amazon
Google
NASDAQ

Building scalable, assured big data systems is expensive.

Google
facebook
amazon

Software Engineering Institute | Carnegie Mellon University

Architecting Software in a New Age
SEI Webinar
© 2015 Carnegie Mellon University

41

**041 The state of the practice
is that the big data problem is not
solved. We know that companies like
Amazon, NASDAQ, Google,

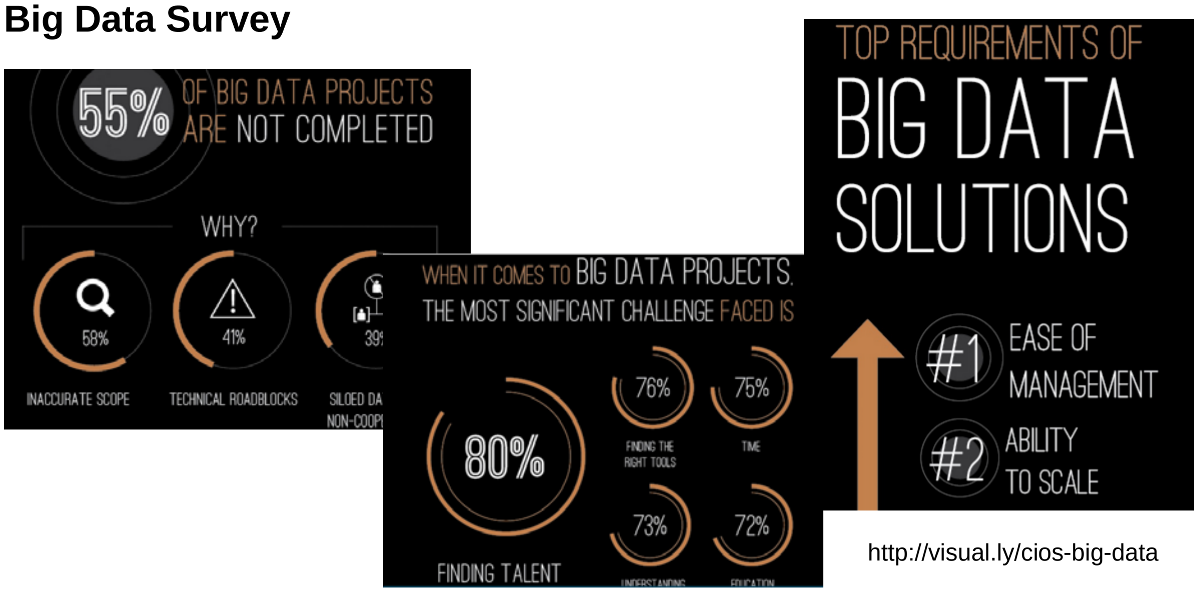
Facebook, and Netflix are way
ahead of most of the rest of us. If
you work for those organizations,
you're in an enviable position. But
these organizations have also been at
it for about a decade, and they have
pumped billions of dollars
against this problem. Most other
organizations have not been able to
enjoy that kind of a lead time and
that kind of a pocketbook.

But even so, these big organizations
have had some problems -- some
problems that many of us have
Experienced -- like the Christmas Eve
2012 Netflix outage; Amazon's
August 19, 2013 45 minutes of
downtime that resulted in five million
dollars loss in revenue; Google's
homepage offline for five minutes on
the 16th of August in 2013; and I
think most of us have already heard
about NASDAQ's issues with Facebook's IPO
in June of 2012.

So, lots of problems. In fact, there
was a study--

Big Data Survey

Big Data Survey



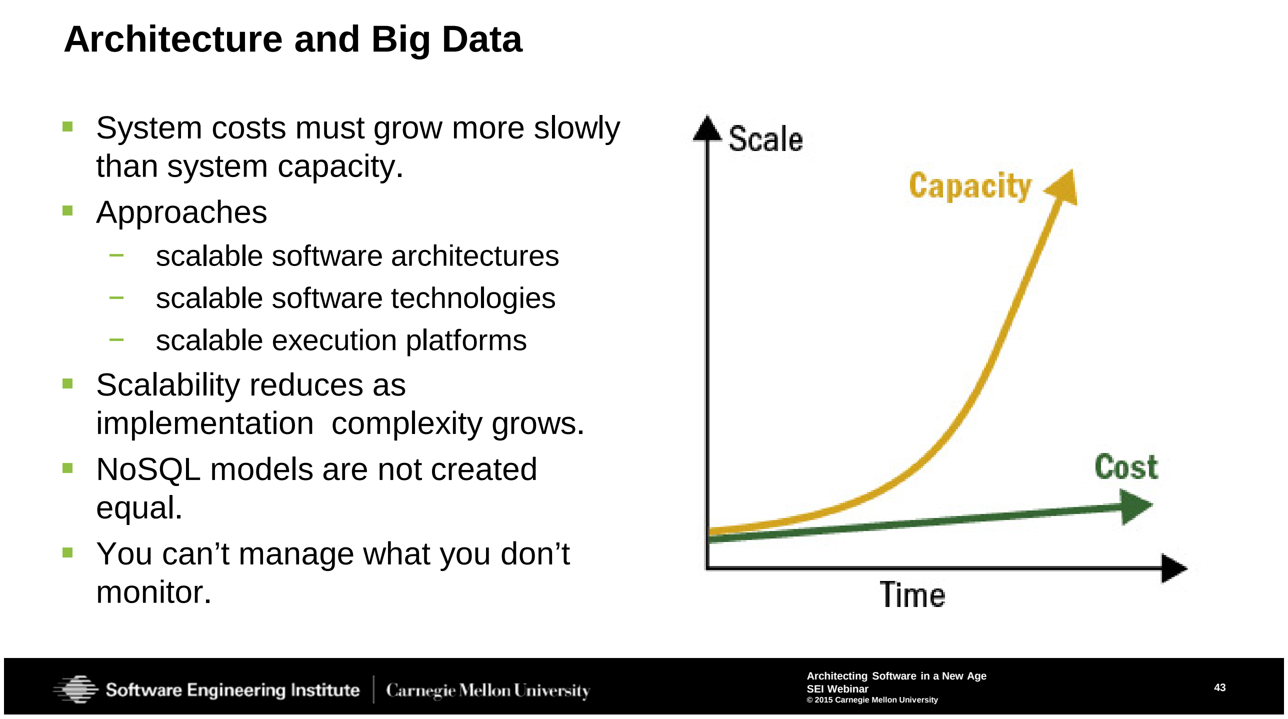
http://visual.ly/cios-big-data

Software Engineering Institute | Carnegie Mellon University

Architecting Software in a New Age
SEI Webinar
© 2015 Carnegie Mellon University

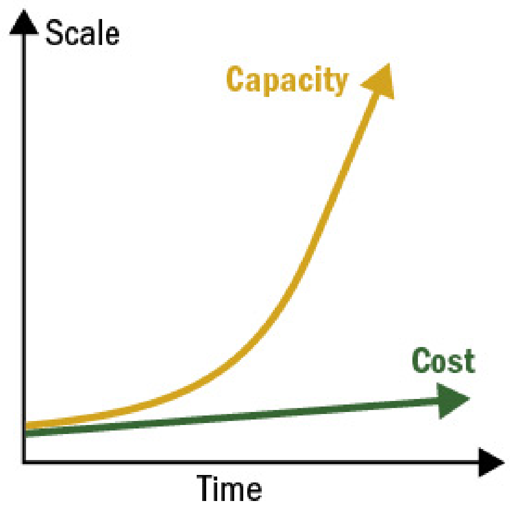
42

**042 --by Infosys that 64 percent of companies admit to having big data initiatives but only 55-- actually 55 percent of them have no strategy for doing so. So we know there are some big challenges here.



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.



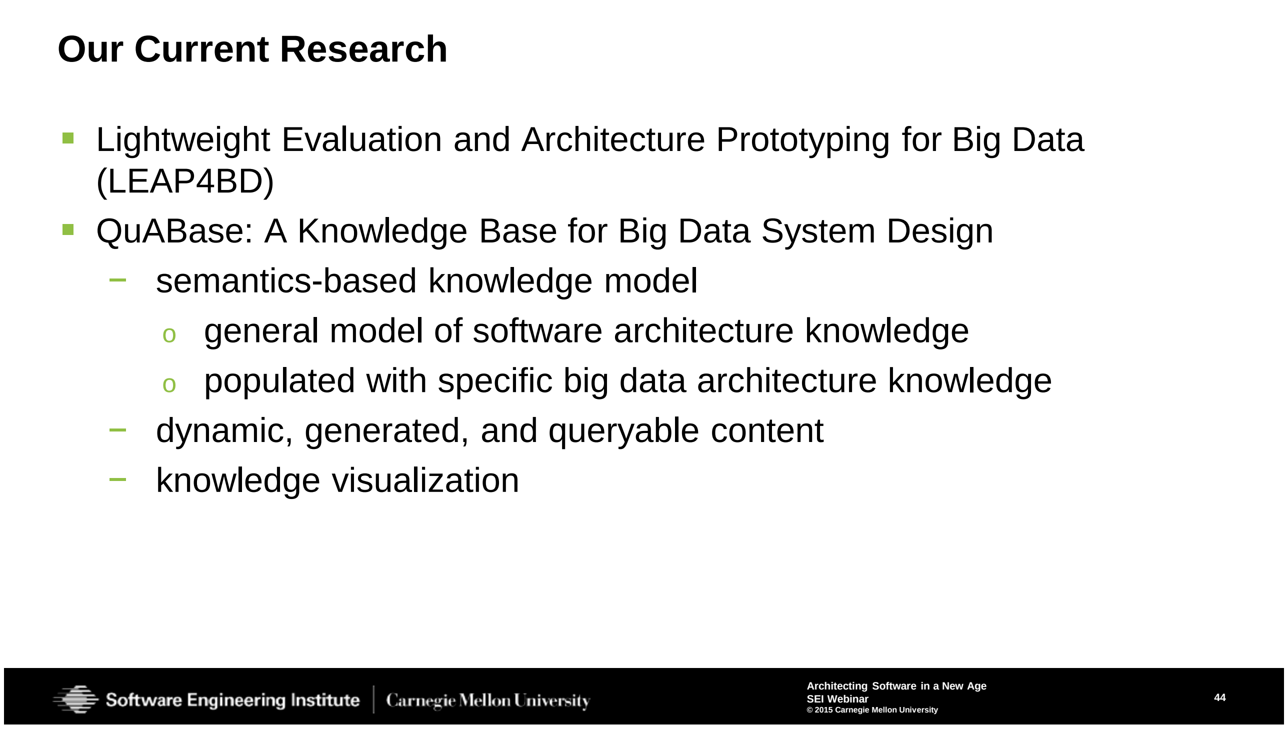
Software Engineering Institute | Carnegie Mellon University

Architecting Software in a New Age
SEI Webinar
© 2015 Carnegie Mellon University

43

**043 When you're thinking about architecture and big data, the tradeoff here is between capacity and cost. We want capacity, but most organizations can't afford the capacity that they actually need. They want tremendous scalability, but as the systems become more complex, as the analysis becomes more complex, there are tradeoffs with scalability. Most people use NoSQL data bases, but those are not all created equal. They have different data models, different query models, different consistency. The other thing is, you need to be able to monitor during runtime what's happening with these big data systems, otherwise you can't manage them. All of these are really big challenges.

Our Current Research



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

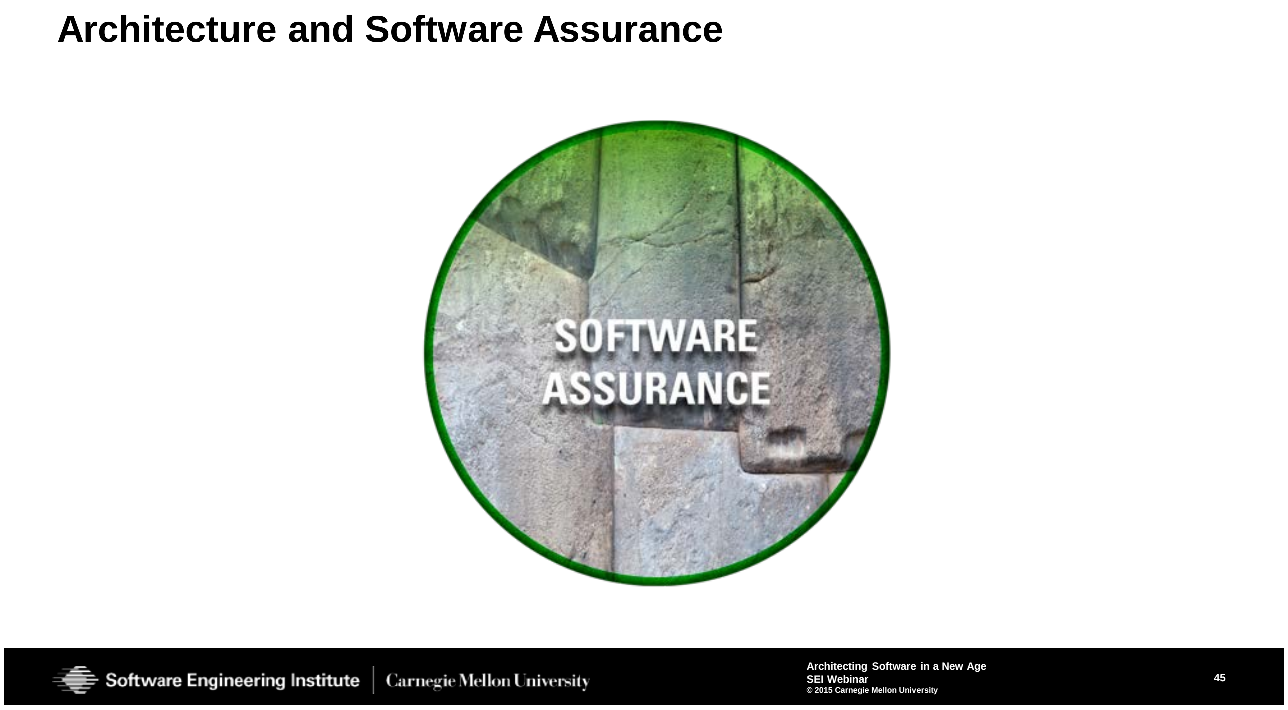
Software Engineering Institute | Carnegie Mellon University

Architecting Software in a New Age
SEI Webinar
© 2015 Carnegie Mellon University

44

**044 We are doing some research in this area. I have some URLs at the end of the talk to direct you to. One of them is called LEAP4BD, where we're actually providing a risk reduction decision support system that allows people to input their quality attribute requirements and a spectrum of the NoSQL technologies that they're looking at, and then provides you some support for what the best choices are. We're actually building this into a knowledge base that continues to grow in knowledge, using machine learning techniques. We call this "QuABase."

Architecture and Software Assurance



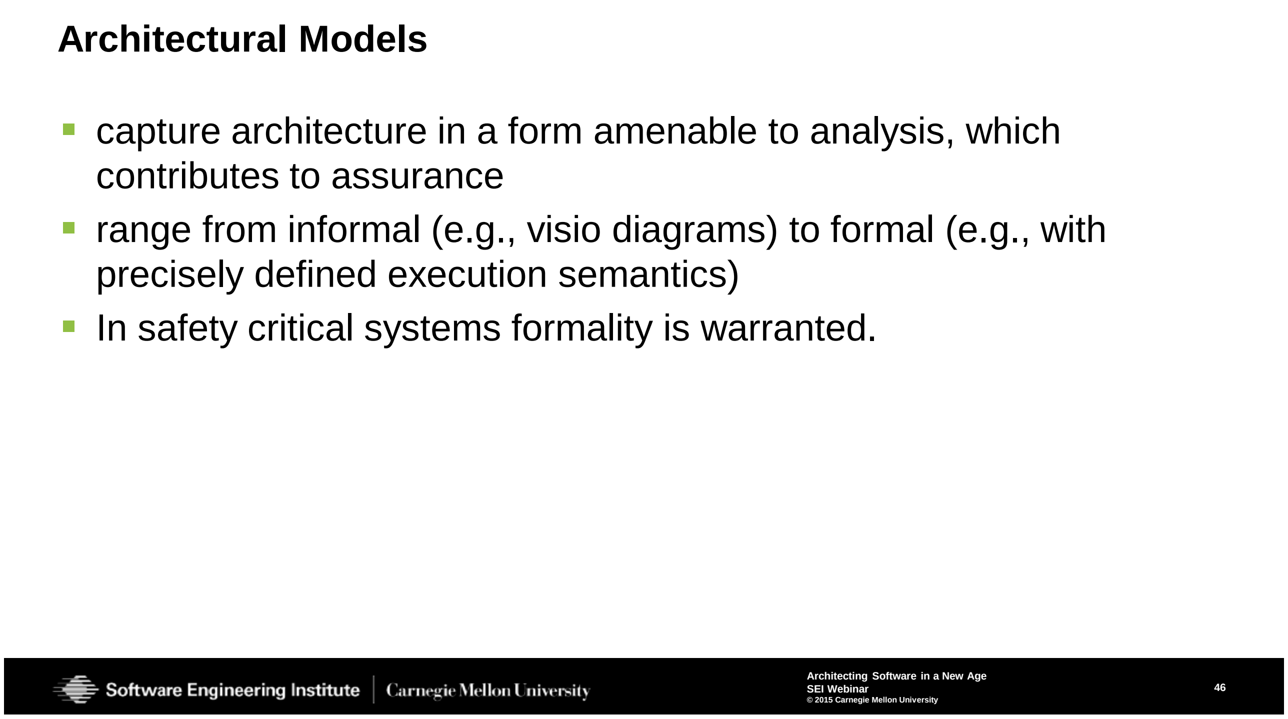
Architecture and Software Assurance

SOFTWARE
ASSURANCE
Software Engineering Institute
Carnegie Mellon University
Architecting Software in a New Age
SEI Webinar
© 2015 Carnegie Mellon University
45

**045 So, everything I've talked about actually is about software assurance, because we're using the architecture to in fact provide assurance that we're going to get the right behavior -- to perform the engineering tradeoffs between the various qualities that are most important to us. And this gives us a level of assurance, provides us evidence that we can count on the System. wWe can count on the system as it's built, as it's deployed, and during runtime.

Rick, in his next talk, is going to focus more on security, which is often interpreted as software assurance. I admit to a broader definition. But I wanted to end with a little bit of a focus on software assurance--

Architectural Models



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.

Software Engineering Institute | Carnegie Mellon University

Architecting Software in a New Age
SEI Webinar
© 2015 Carnegie Mellon University

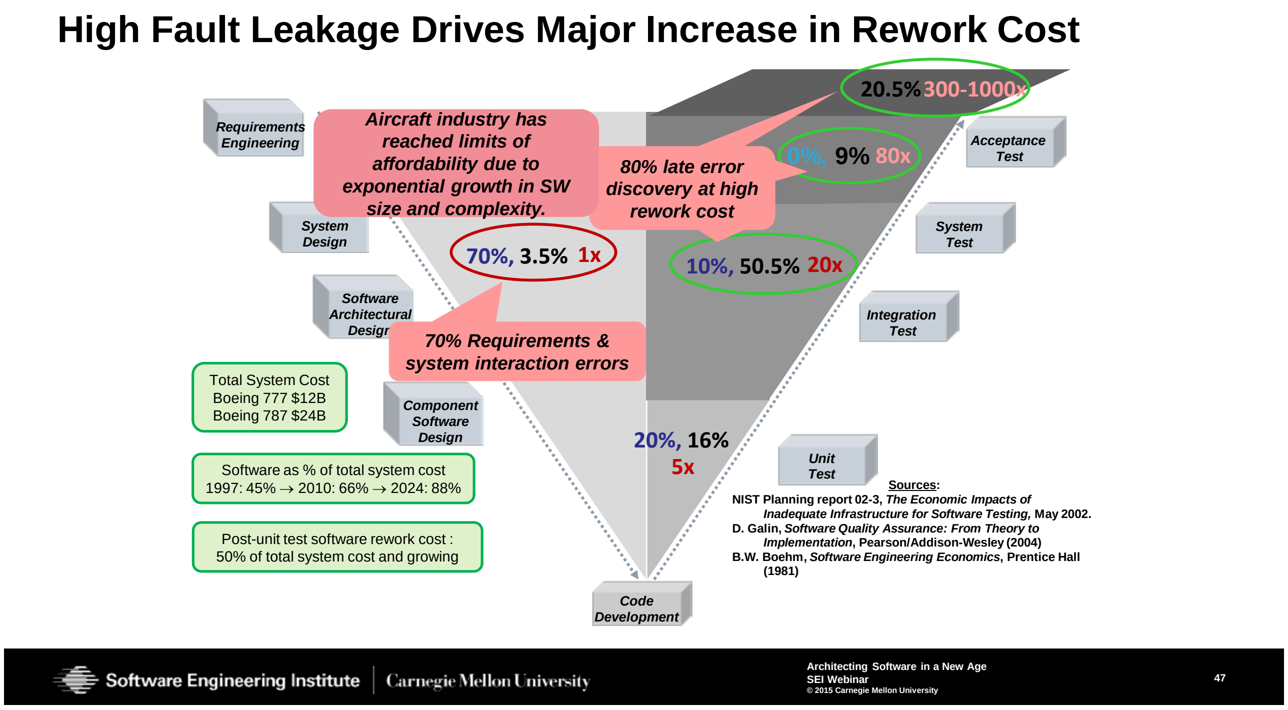
46

**046 --Because I haven't yet said much about how architectures are depicted. What do we use?

Well basically, you need enough detail in your depiction to do the analysis that you're trying to perform, and that depiction can be informal, from Visio diagrams, to formal using formal architecture languages that have precisely defined execution semantics.

When we're talking about safety-critical systems -- when we're talking about systems that are internal to the engines of our automobiles or in avionics -- we need more. Informal models are not sufficient.

High Fault Leakage Drives Major Increase in Rework Cost



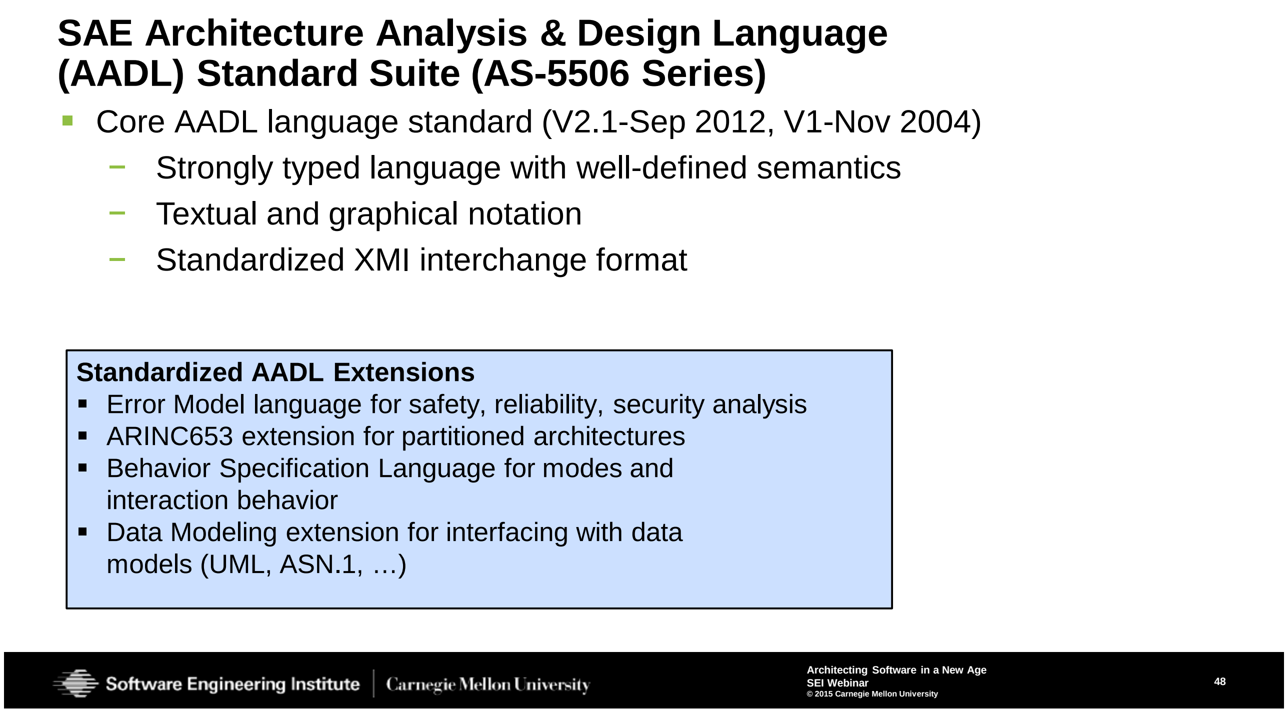
High Fault Leakage Drives Major Increase in Rework Cost

Requirements Engineering
System Design
Software Architectural Design
Component Software Design
Code Development
Unit Test
Integration Test
System Test
Acceptance Test
Aircraft industry has reached limits of affordability due to exponential growth in SW size and complexity.
80% late error discovery at high rework cost
70% Requirements & system interaction errors
20.5% 300-1000x
0%, 9% 80x
10%, 50.5% 20x
70%, 3.5% 1x
20%, 16% 5x
Total System Cost Boeing 777 $12B Boeing 787 $24B
Software as % of total system cost 1997: 45% → 2010: 66% → 2024: 88%
Post-unit test software rework cost : 50% of total system cost and growing
Software Engineering Institute | Carnegie Mellon University
Architecting Software in a New Age
SEI Webinar
© 2015 Carnegie Mellon University
47

Sources:
NIST Planning report 02-3, *The Economic Impacts of Inadequate Infrastructure for Software Testing*, May 2002.
D. Galin, *Software Quality Assurance: From Theory to Implementation*, Pearson/Addison-Wesley (2004)
B.W. Boehm, *Software Engineering Economics*, Prentice Hall (1981)

**047 And so we need the formality because we understand, from lots of studies-- this is a chart from one of them-- that in these sorts of systems-- this is from the avionics industry-- that the cost of air vehicles is increasing dramatically, and the amount of software in those air vehicles is also increasing dramatically. And what we see is faults that leak through the system and the tremendous task of testing and integrating the system-- almost prohibitive. And so what we need to think about is how we can use the architecture early on and do some formal reasoning so that we actually can eliminate some of those faults and preclude them from leaking through the lifecycle of the system.

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



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, …)

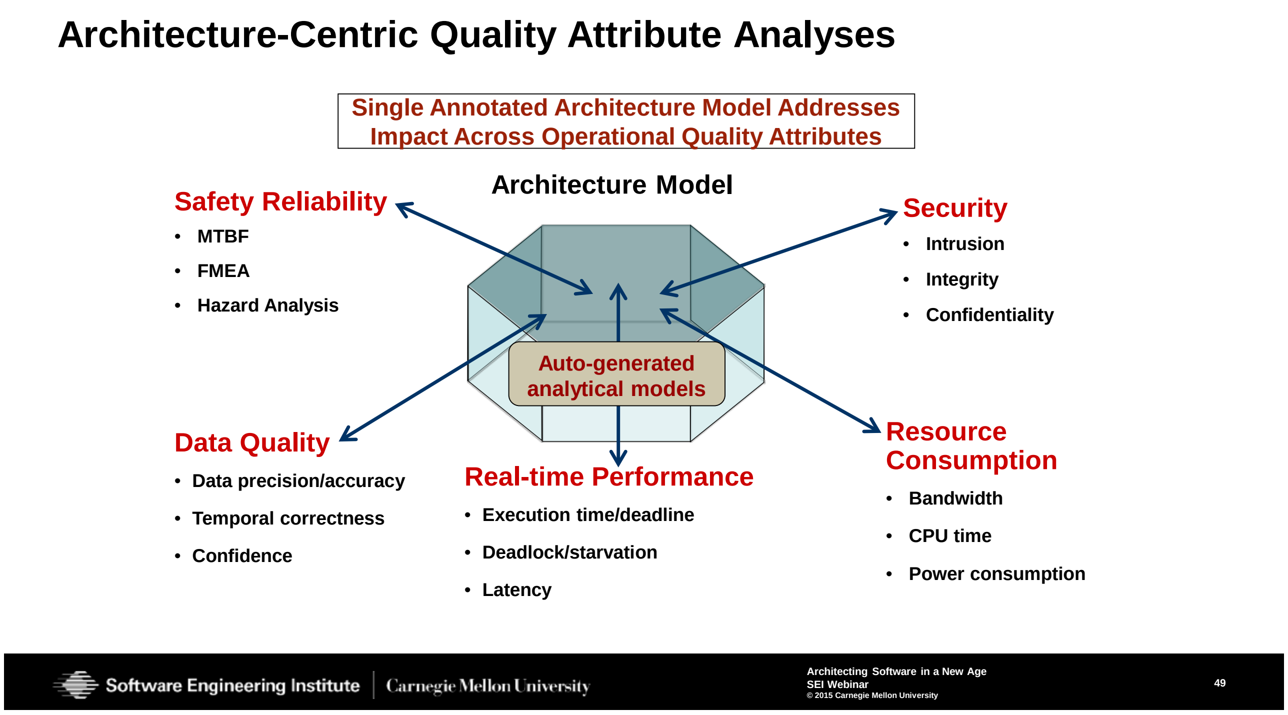
Software Engineering Institute | Carnegie Mellon University

Architecting Software in a New Age
SEI Webinar
© 2015 Carnegie Mellon University

48

**048 One effort is a language
called the Architecture Analysis and
Design Language, or AADL, which the
SEI has been involved in developing, and this
language provides grist for doing that
sort of formal reasoning.

Architecture-Centric Quality Attribute Analyses



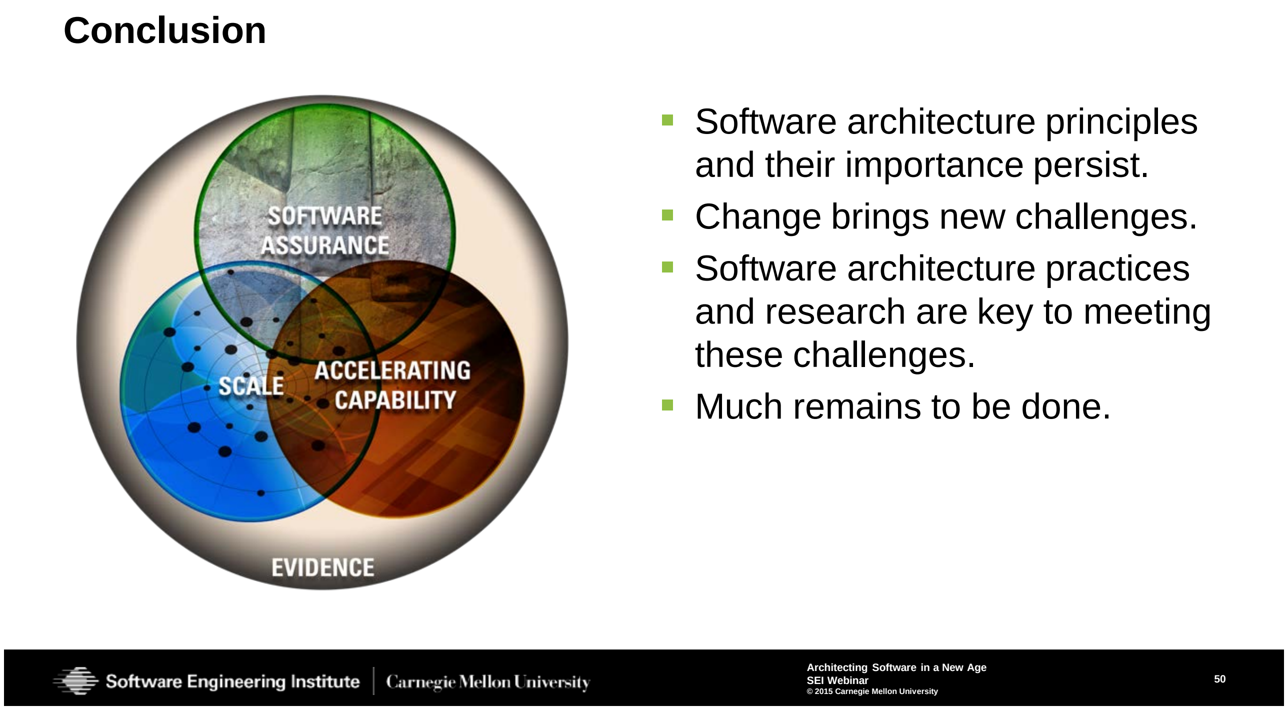
Architecture-Centric Quality Attribute Analyses

Single Annotated Architecture Model Addresses Impact Across Operational Quality Attributes

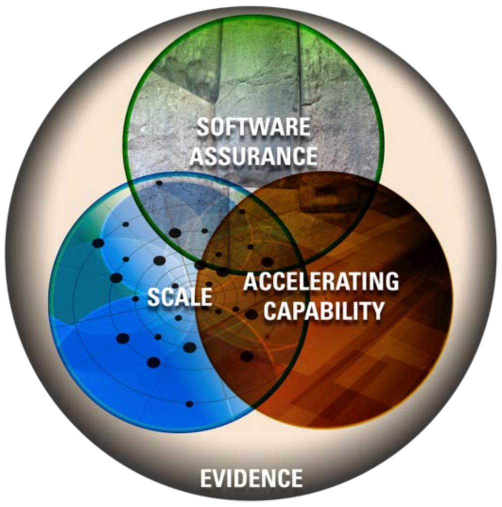
**049 In fact, provide a semantic model that allows us to express quality attributes in a formal way, and to be able to reason about all of those quality attributes using formal mechanisms, and make the appropriate tradeoffs. We, with this technique and this language, have been able to perform what's being called virtual integration, so that before the system is developed we can actually make tradeoffs using these architectural models, virtually integrate the system, and identify lots of problems that would only be detected downstream in integration and test previously.

So these are This is the kind of technique that can be used to provide the level of software assurance that you need in safety-critical systems, and they're based on formal architectural modeling.

Conclusion



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.

Software Engineering Institute | Carnegie Mellon University

Architecting Software in a New Age
SEI Webinar
© 2015 Carnegie Mellon University

50

**050 So let me conclude by saying that if you thought that software architecture was an old idea, something that started in the '80s and maybe was best left to the early 2000s, before our brave new world of social media and cloud computing and mobile computing and all of the rest, I hope I've convinced you that the principles of software architecture and their importance persist. The challenges are different, but the need to be able to do ongoing analysis -- to do tradeoff analysis -- is still very key. And the demands on our systems are much higher than systems of the past, and in fact the systems themselves are much more prolific,

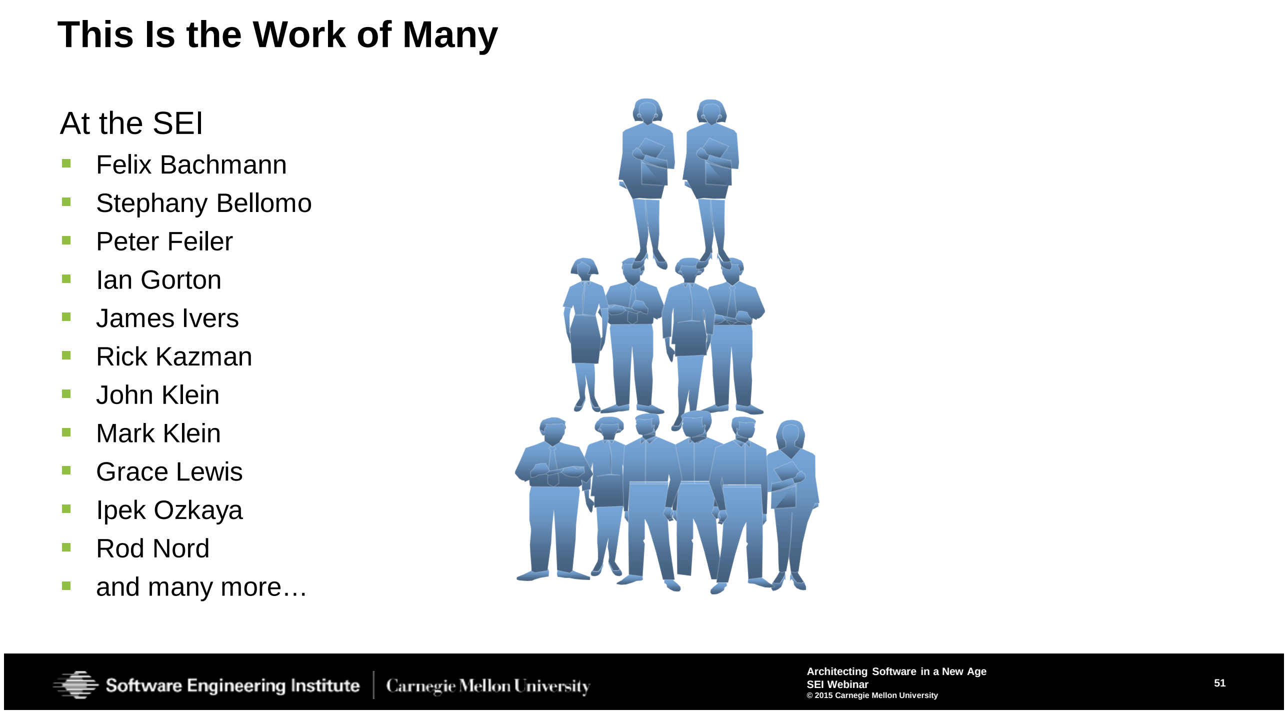
and much more important to life as
we know it.

So there's a lot to be done. What I
see in the future are much more fluid
architectures, much more tool
support, adaptive architectures,
architectures with lots of runtime
monitoring built into them so they
are capable of internal monitorability.

And I will stop there. I hope this has
provided you sort of a whirlwind
perspective of not only where we've
been and how important architecture is,
but some of the important challenges
and the relationships of architecture to
those challenges. Thanks very
much.

Shane McGraw: Linda, thank you.
That was a terrific talk, and Linda's
going to stick around, folks, for the
Q&A after Rick's talk, so she'll be
here for any questions that came
through during that part of the
presentation, we'll address in a few
minutes.

This Is the Work of Many



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…

Software Engineering Institute | Carnegie Mellon University

Architecting Software in a New Age
SEI Webinar
© 2015 Carnegie Mellon University

51

**051 Just one quick housekeeping item to address.

Approaching Security from an "Architecture First" Perspective

Software Engineering Institute, Carnegie Mellon University

Rick Kazman - University of Hawaii
Jungwoo Ryoo - Penn State University
Humberto Cervantes -
Universidad Autonoma Metropolitana-Iztapalapa

Software Engineering Institute | Carnegie Mellon University

**056 A number of questions came in about recording and the availability of the slides. The event is being archived. The login will be the same that you used today. It should be available at some point tomorrow. An email will go out letting you know when the archive is up, and it's the same login process as today.