现代软件工程趋势

Carnegie Mellon University
Software Engineering Institute

[DISTRIBUTION STATEMENT A] Approved for public release and unlimited distribution

Modern Trends through an Architecture Lens

Linda Northrop

Software Engineering Institute
Carnegie Mellon University
Pittsburgh, PA 15213

社会对软件的依赖

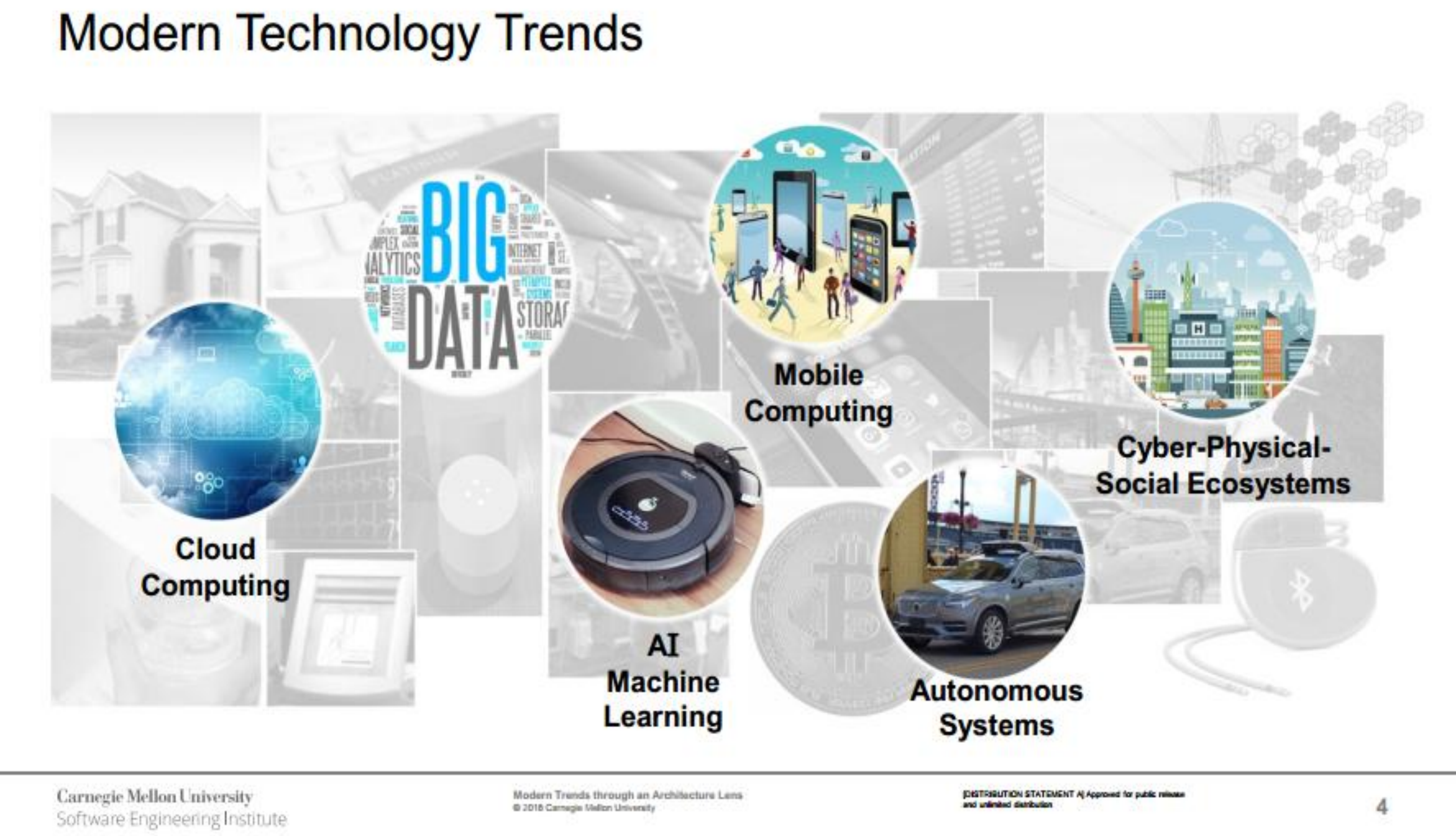
Society’s Dependence on Software

Carnegie Mellon University
Software Engineering Institute

Modern Trends through an Architecture Lens
© 2018 Carnegie Mellon University

[DISTRIBUTION STATEMENT A] Approved for public release and unlimited distribution

现代技术趋势



Modern Technology Trends

Cloud Computing

Big Data

AI Machine Learning

Mobile Computing

Autonomous Systems

Cyber-Physical-Social Ecosystems

软件开发的趋势

Software Development Trends

Clustrix

Jenkins

mongoDB

Agile approaches

DevOps

Scripting languages

Dashboards

Application frameworks

Distributed development environments

Open source libraries

Containers

Microservices

NoSQL

docker

GitHub

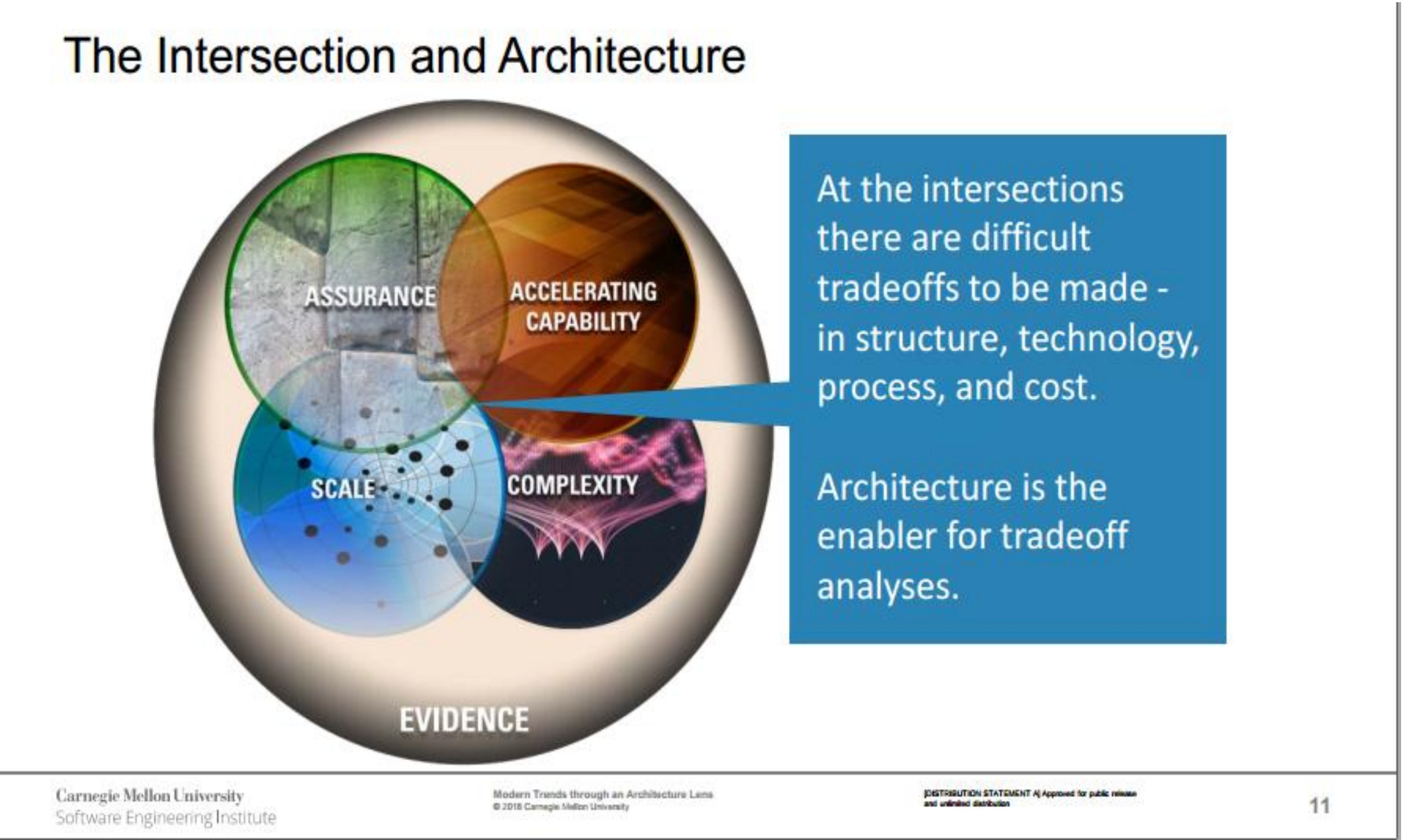
hadoop

snowflake

ANGULAR by Google

APACHE Spark™

软件工程的挑战

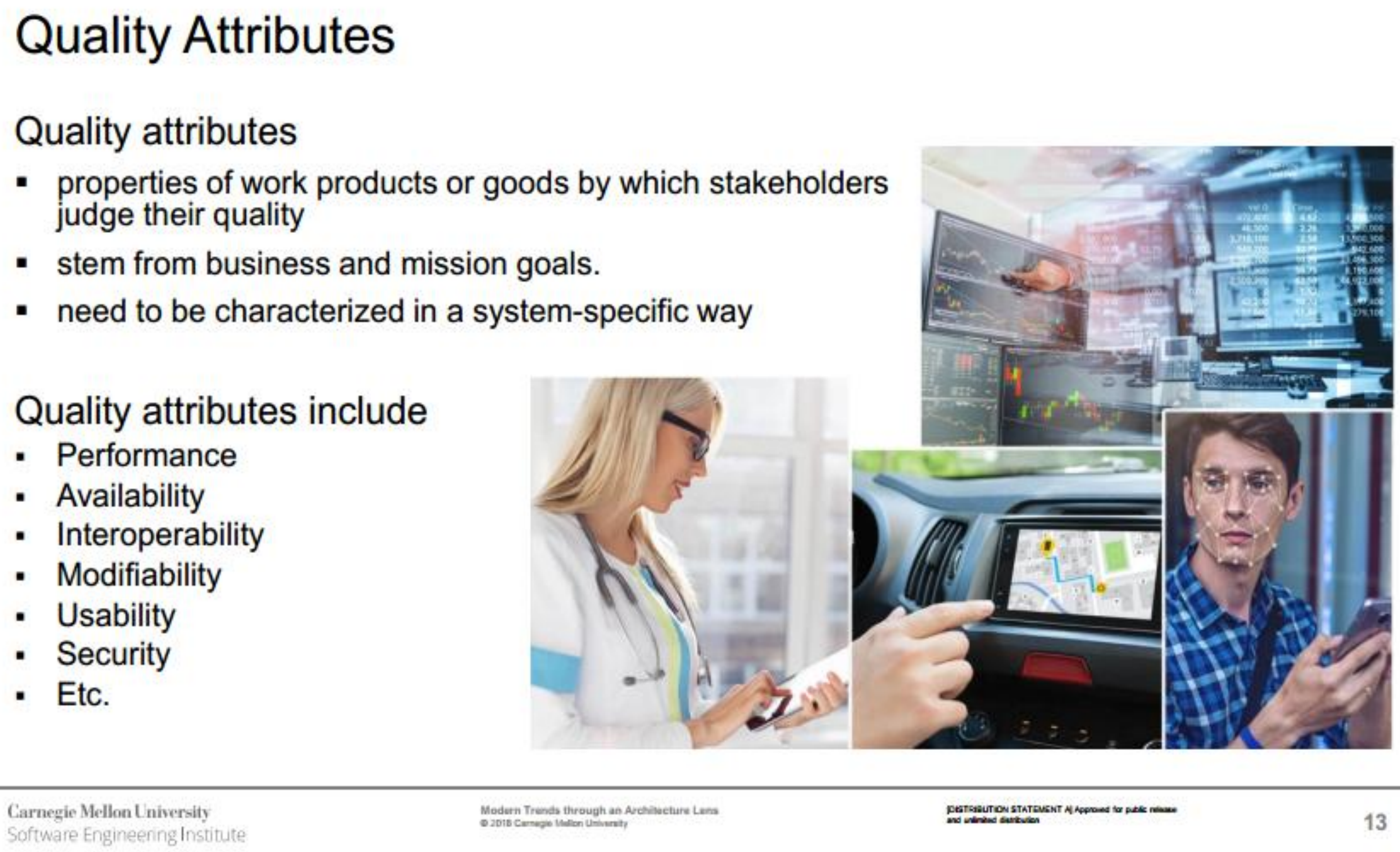


The Intersection and Architecture

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

Architecture is the enabler for tradeoff analyses.

软件的质量属性



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.

软件架构的中心作用

Mobile Computing
BIG DATA
Cloud Computing
Software Architecture
SCALE
COMPLEXITY
AI Machine Learning
Cyber-Physical Social Ecosystems
Autonomous Systems

Carnegie Mellon University
Software Engineering Institute

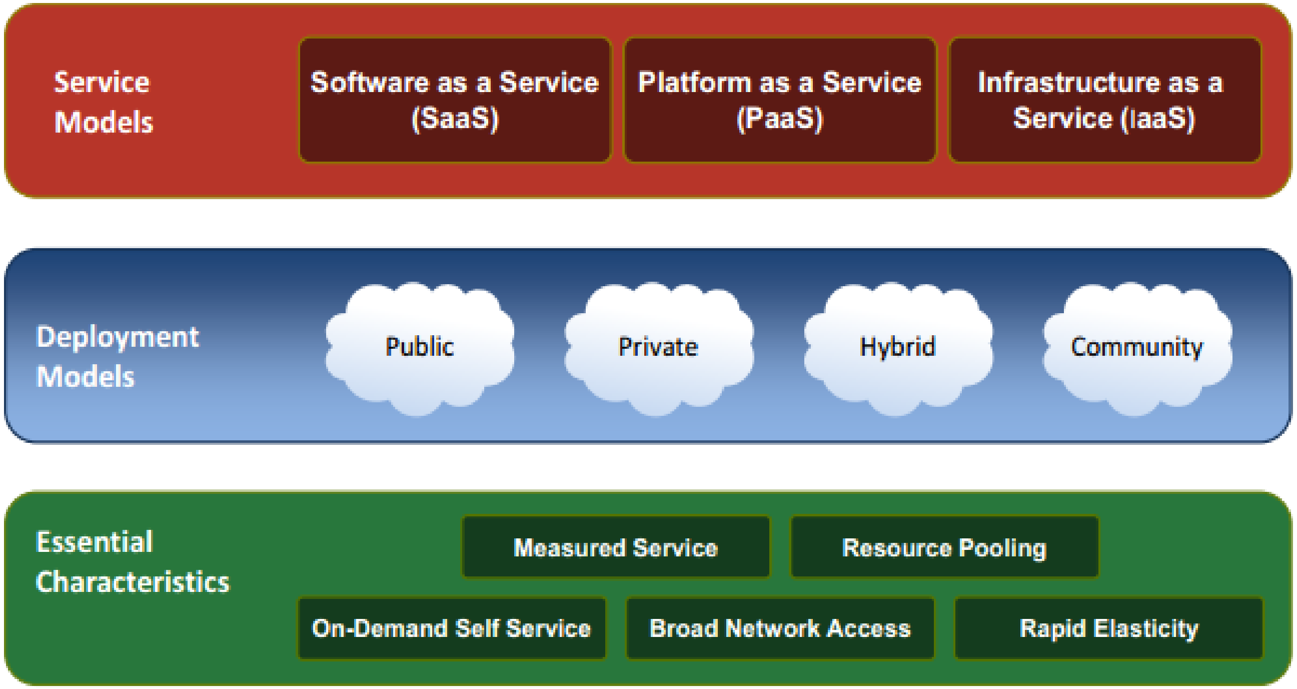
Modern Trends through an Architecture Lens
© 2018 Carnegie Mellon University

DISTRIBUTION STATEMENT A] Approved for public release and unlimited distribution

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

云计算模型和基础特性

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

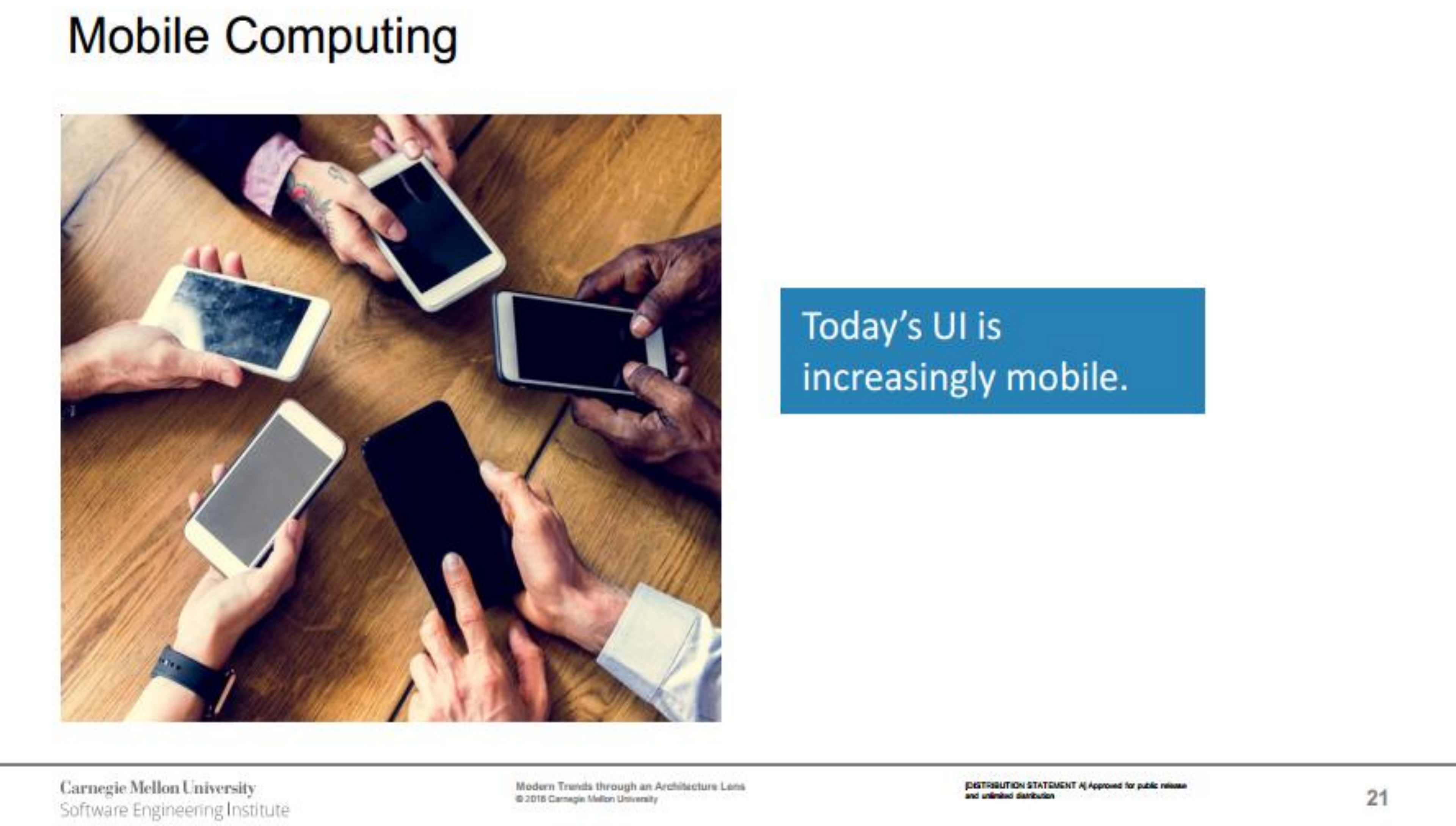
On-Demand Self Service

Broad Network Access

Rapid Elasticity

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

移动计算

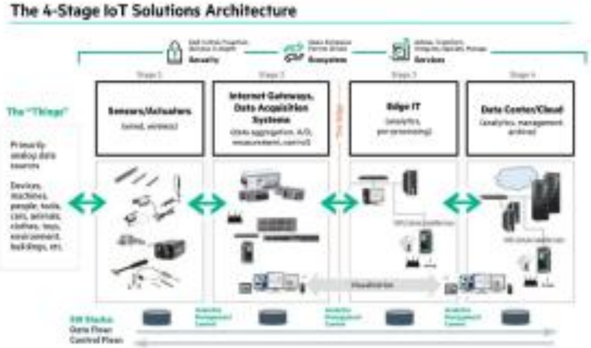


Mobile Computing

Today's UI is increasingly mobile.

软件架构趋势：边缘计算和雾计算

Related Architecture Trends: Edge and Fog Computing


The 4-Stage IoT Solutions Architecture
Security
Ecosystem
Services
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)
Edge IT
(analytics, pre-processing)
Data Center/Cloud
(analytics, management, archive)
Visualization
SW Stacks
Data Flow
Control Flow

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

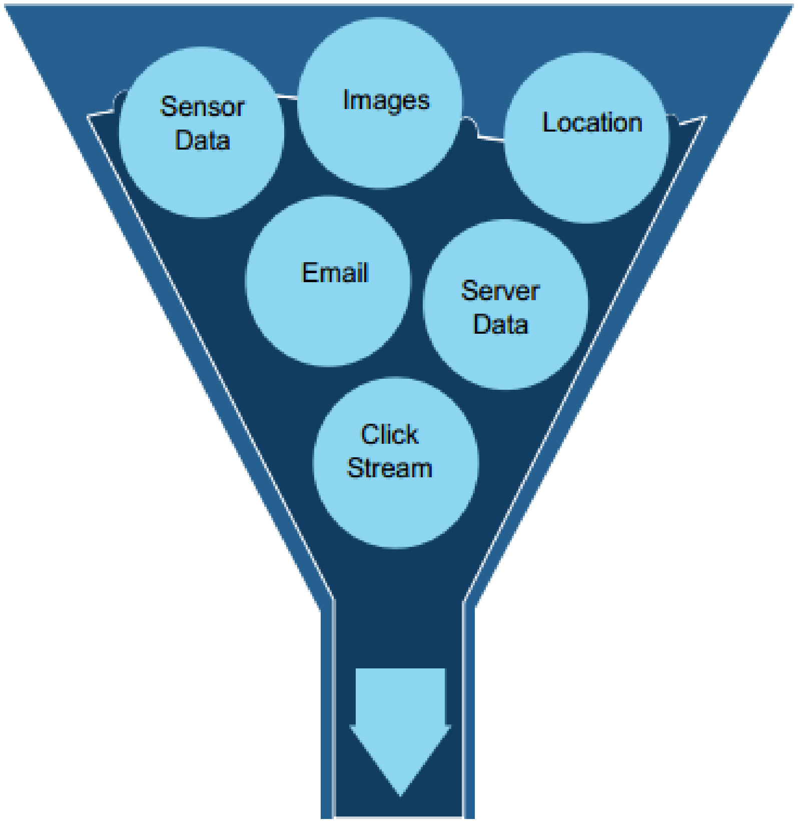
Architecture challenges

- 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

大数据系统

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

大数据实践状态：从先驱者到多元化工业

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

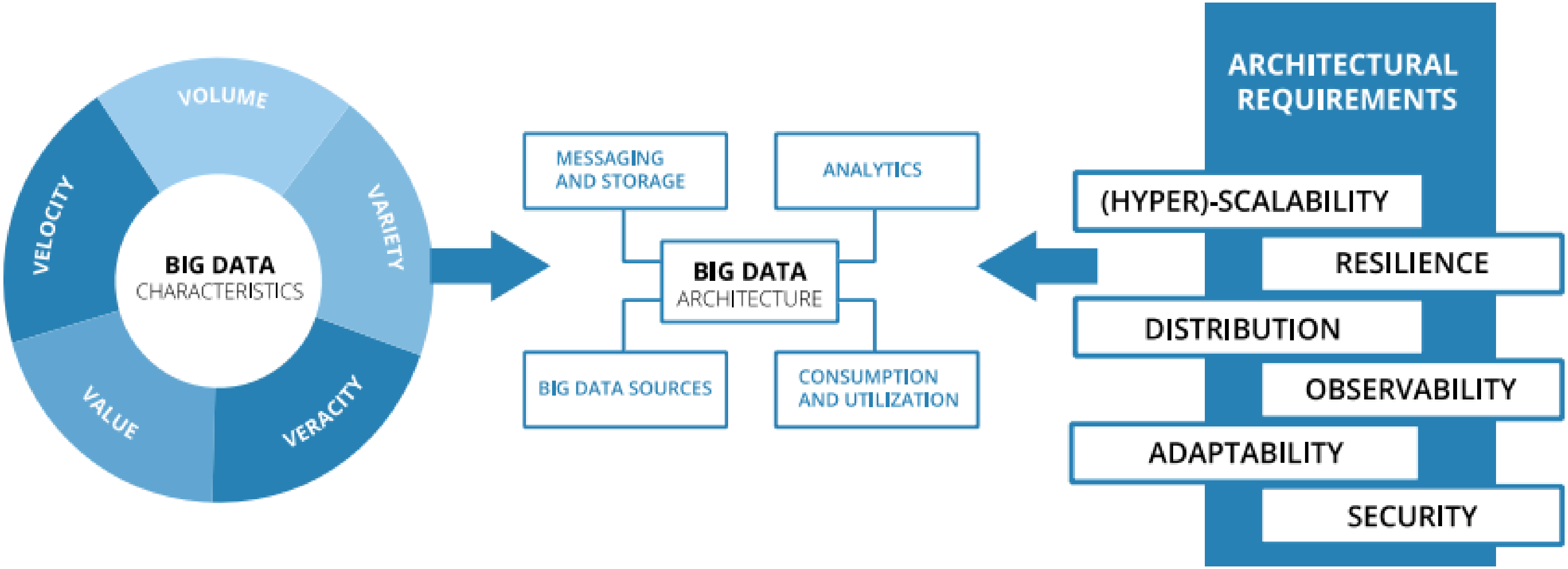
Google
amazon
facebook
NETFLIX
Microsoft

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

Data is a business asset.

大数据的软件架构

Big Data Software Architecture



BIG DATA CHARACTERISTICS

VOLUME

VARIETY

VERACITY

VALUE

VELOCITY

MESSAGING AND STORAGE

ANALYTICS

BIG DATA ARCHITECTURE

BIG DATA SOURCES

CONSUMPTION AND UTILIZATION

ARCHITECTURAL REQUIREMENTS

(HYPER)-SCALABILITY

RESILIENCE

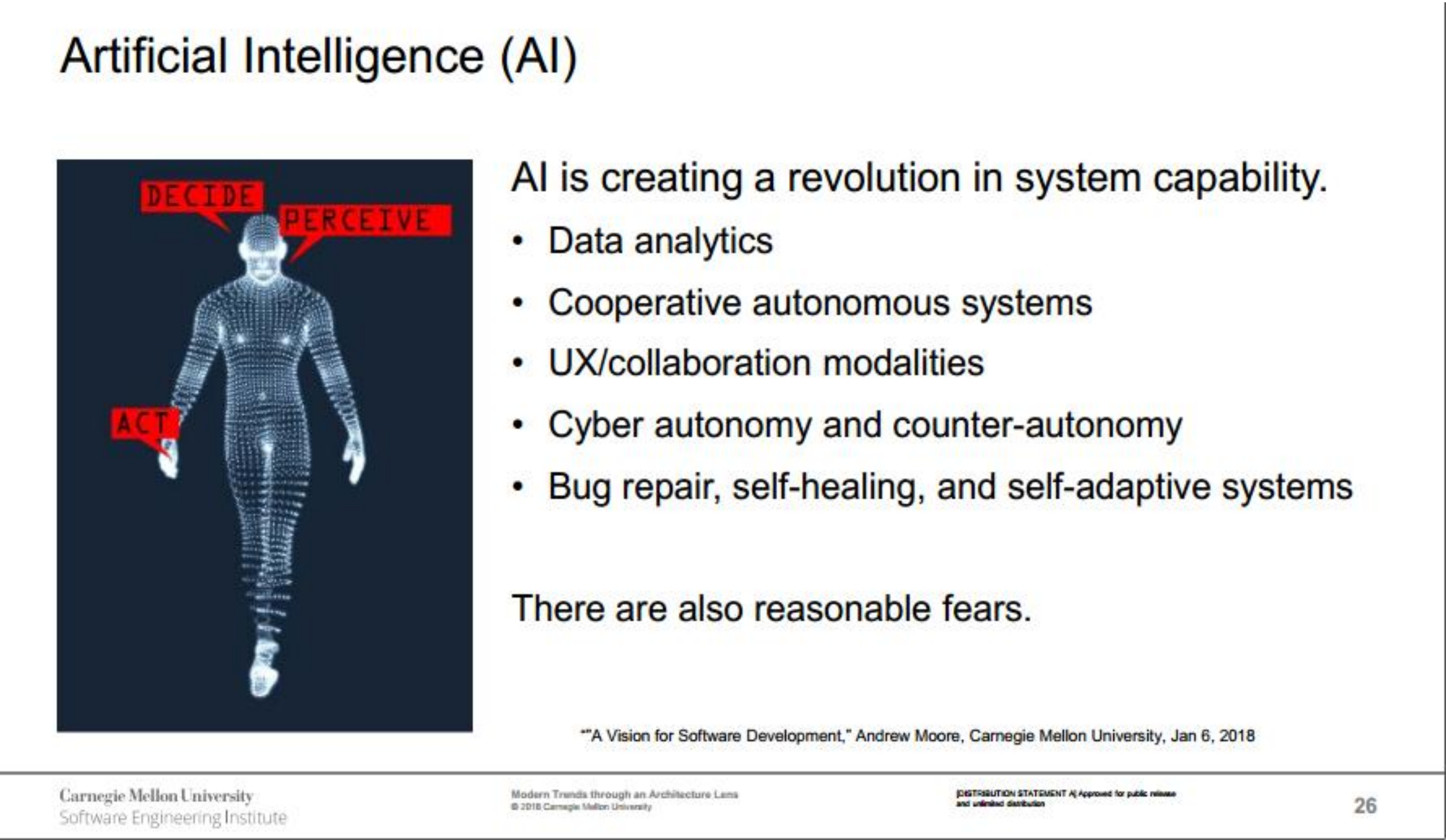
DISTRIBUTION

OBSERVABILITY

ADAPTABILITY

SECURITY

人工智能



Artificial Intelligence (AI)

DECIDE
PERCEIVE
ACT

AI is creating a revolution in system capability.

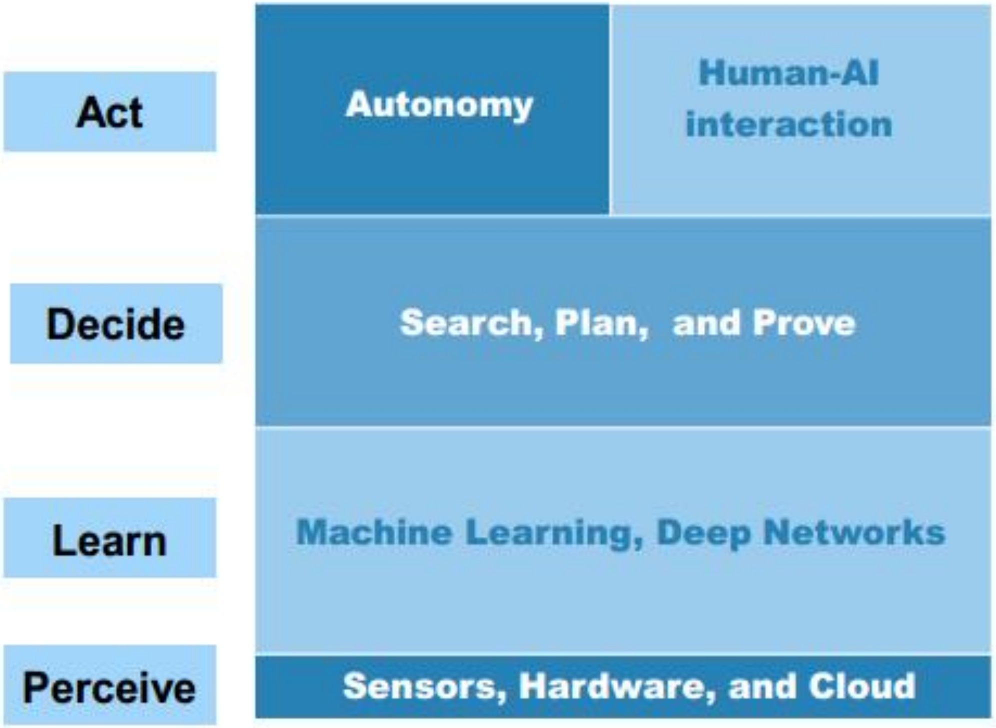
- Data analytics
- Cooperative autonomous systems
- UX/collaboration modalities
- Cyber autonomy and counter-autonomy
- Bug repair, self-healing, and self-adaptive systems

There are also reasonable fears.

*"A Vision for Software Development," Andrew Moore, Carnegie Mellon University, Jan 6, 2018

人工智能栈

AI Stack



Act

Autonomy

Human-AI interaction

行动

Decide

Search, Plan, and Prove

决策

Learn

Machine Learning, Deep Networks

学习

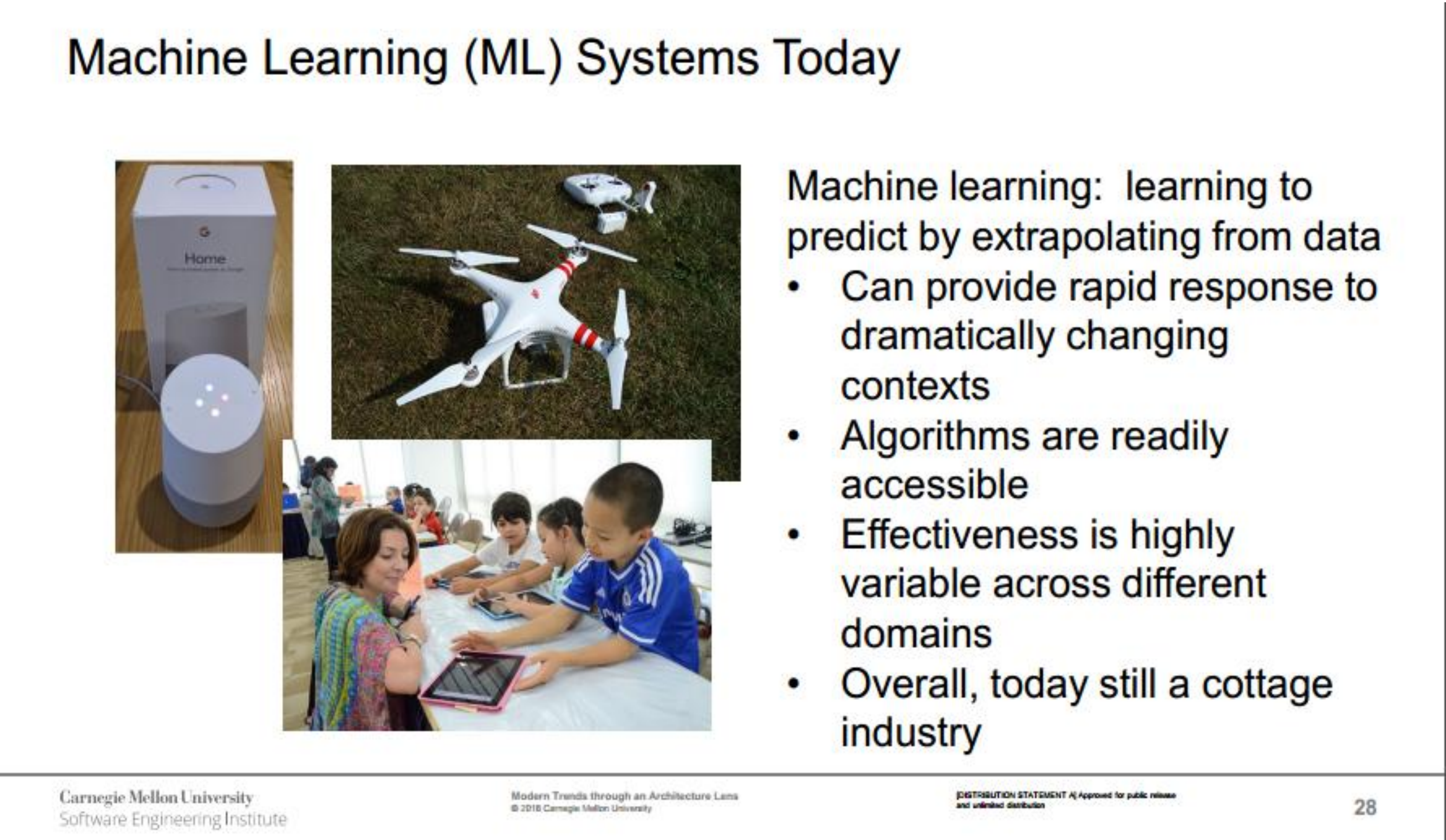
Perceive

Sensors, Hardware, and Cloud

感知

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

机器学习系统现状

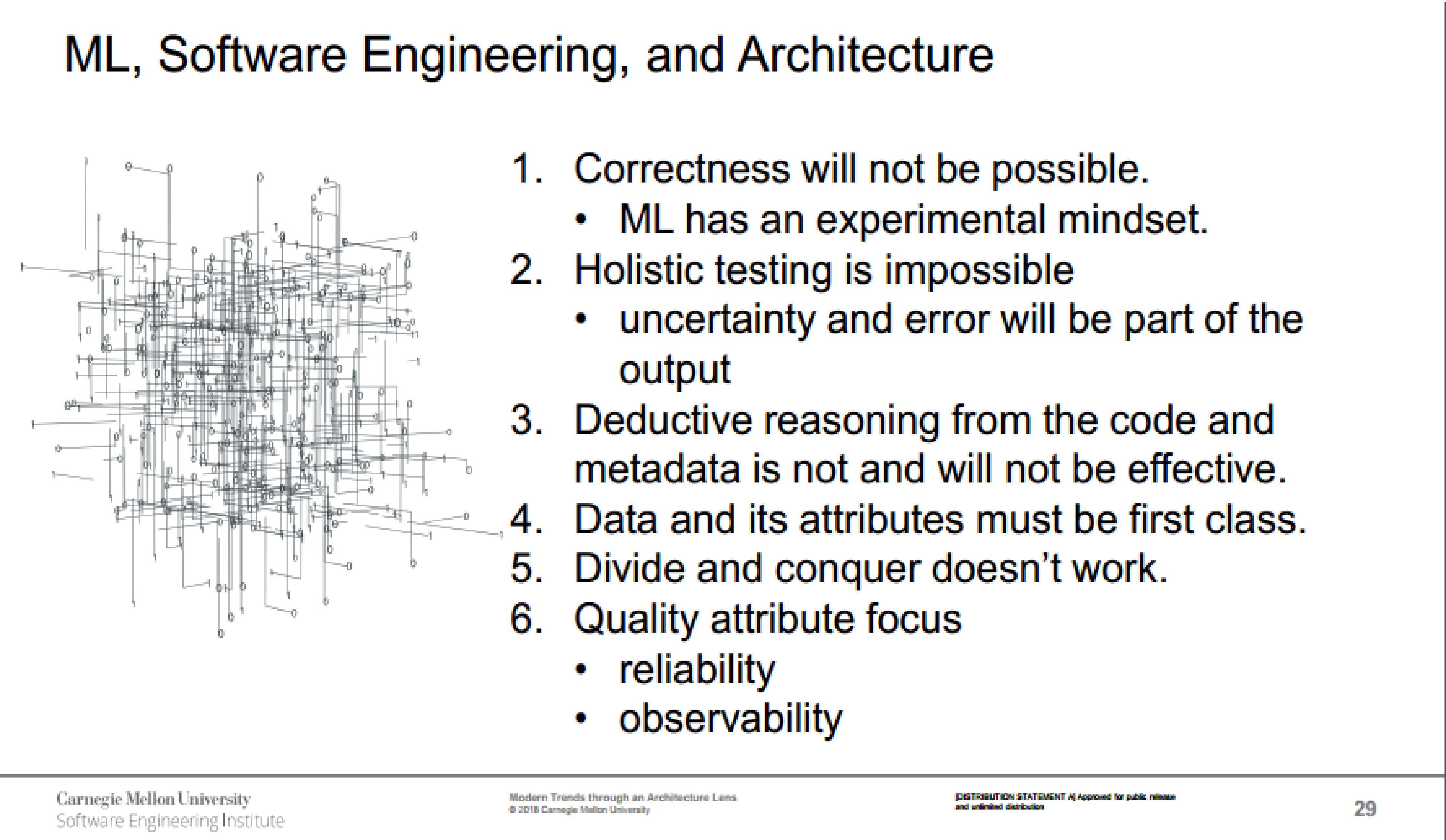


Machine Learning (ML) Systems Today

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

机器学习、软件工程、软件架构



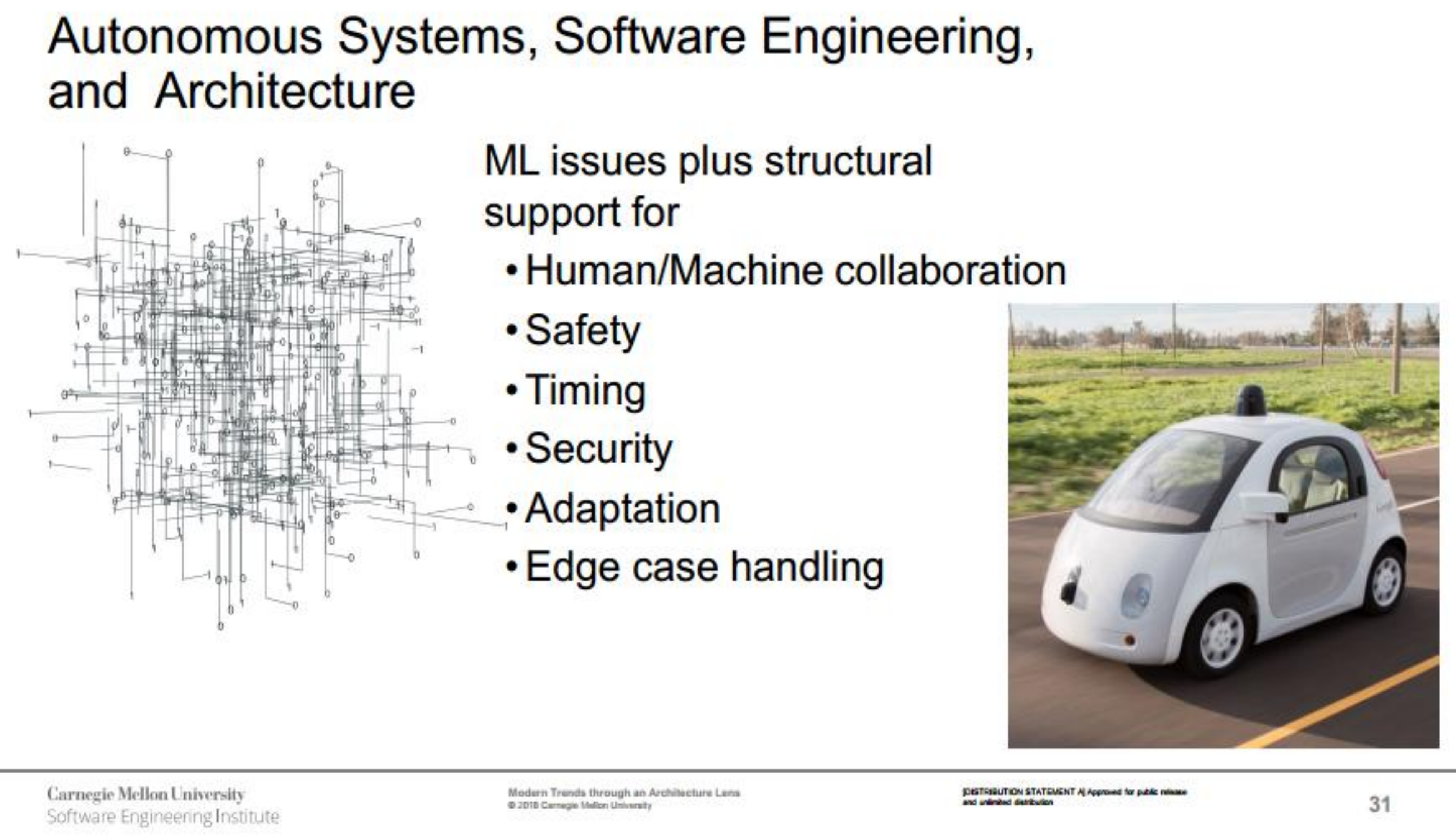
ML, Software Engineering, and Architecture

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

自动系统

Autonomous Systems

自动系统、软件工程、软件架构



Autonomous Systems, Software Engineering, and Architecture

ML issues plus structural support for

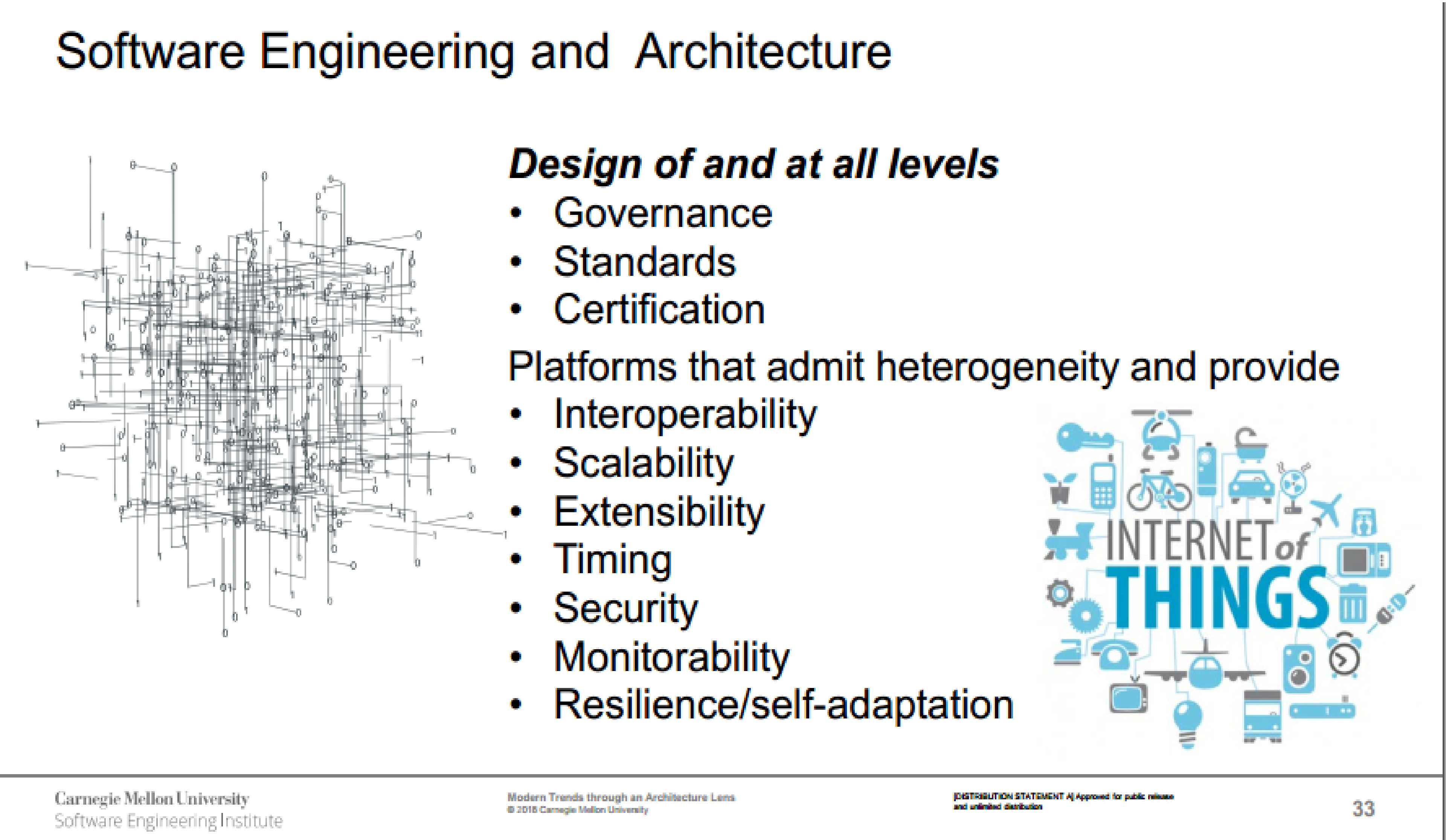
- Human/Machine collaboration
- Safety
- Timing
- Security
- Adaptation
- Edge case handling

网络-物理-社会生态

Cyber-Physical-Social Ecosystems

INTERNET of THINGS
Emergency
Ultra-Large-Scale Systems
The Software Challenge of the Future
Software Engineering Institute
Carnegie Mellon

软件工程和软件架构



Software Engineering and Architecture

Design of and at all levels

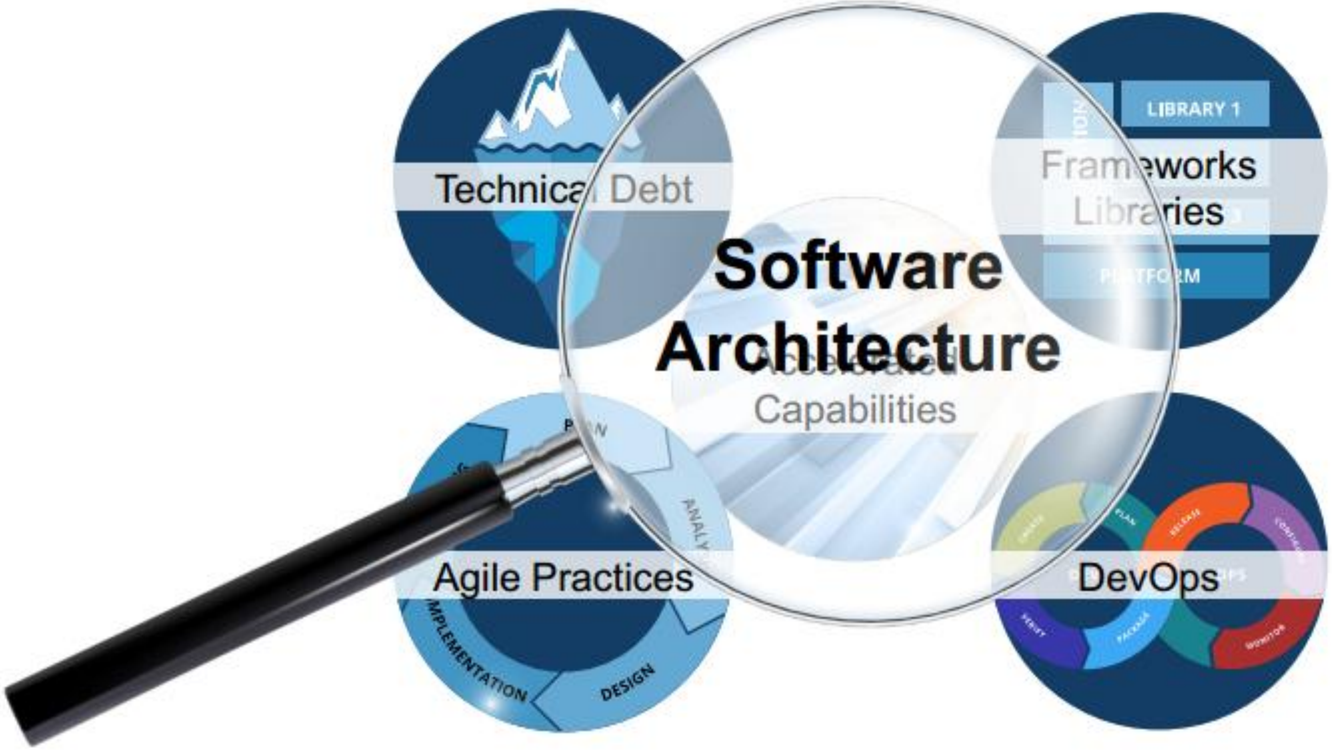
- Governance
- Standards
- Certification

Platforms that admit heterogeneity and provide

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

INTERNET of
THINGS

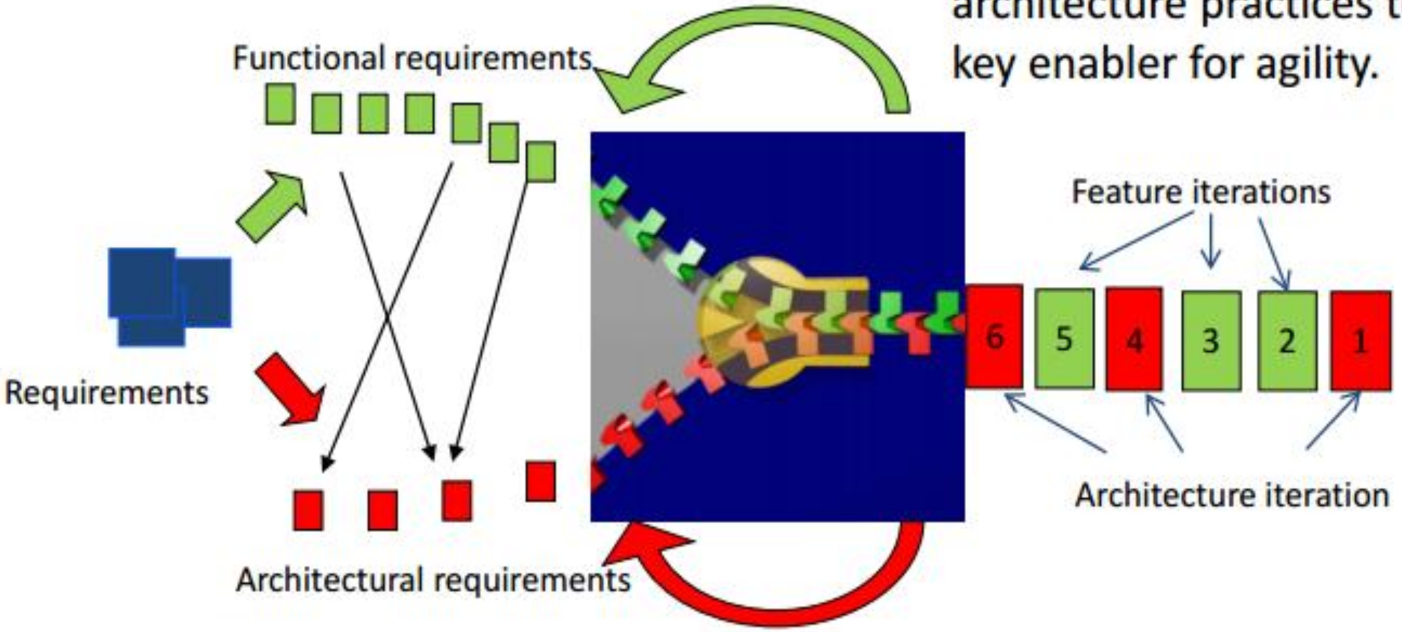
软件工程和软件架构


Technical Debt
LIBRARY 1
Frameworks
Libraries
PLATFORM
Software
Architecture
Capabilities
PLAN
ANALY
Agile Practices
IMPLEMENTATION
DESIGN
CREATE
PLAN
RELEASE
CONFIGURE
DevOps
VERIFY
PACKAGE
MONITOR

集成敏捷开发和软件架构的实践

Integrated Agile/Architecture Practices

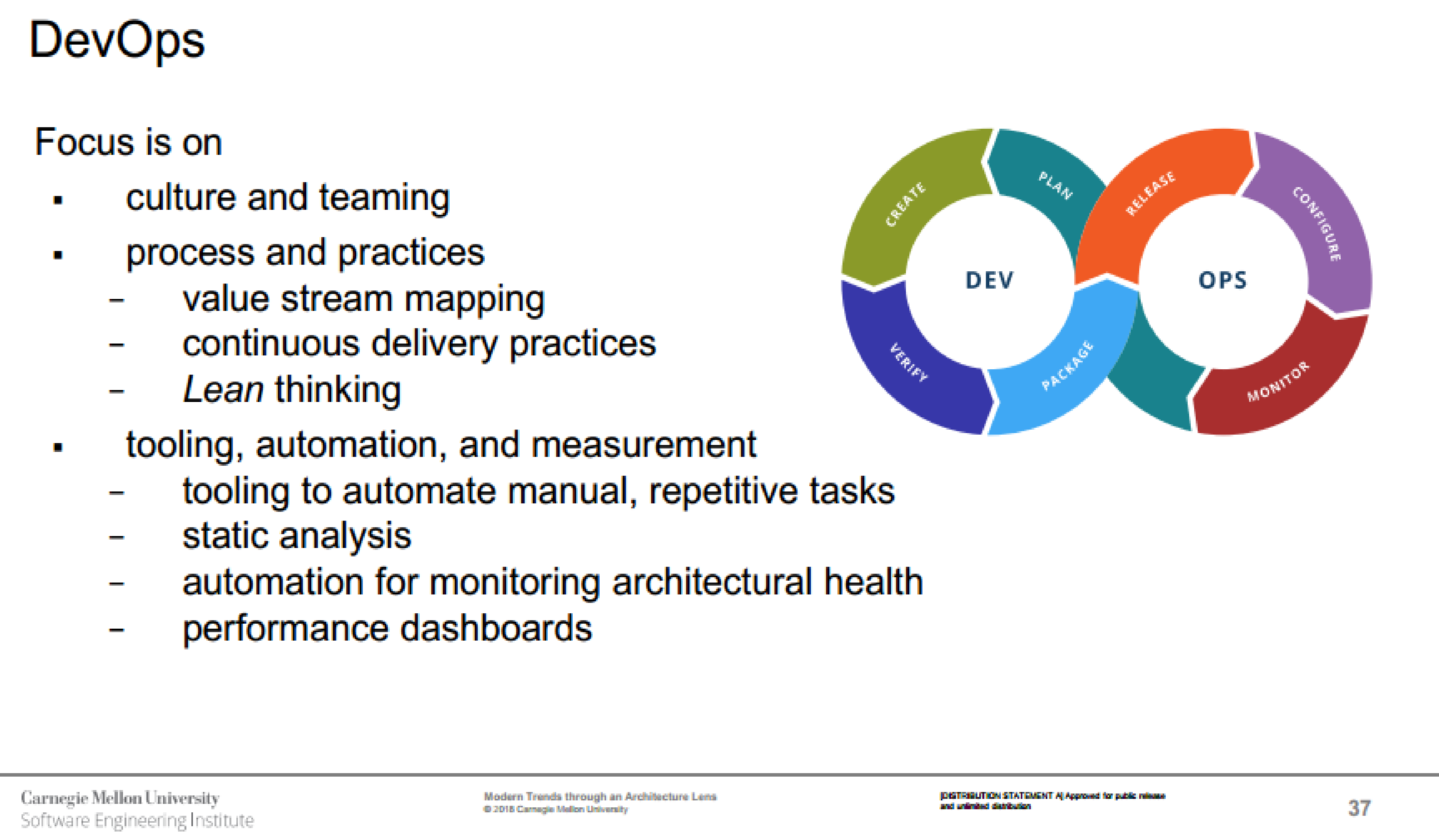
Successful project teams find architecture practices to be a key enabler for agility.



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: Development & Operations



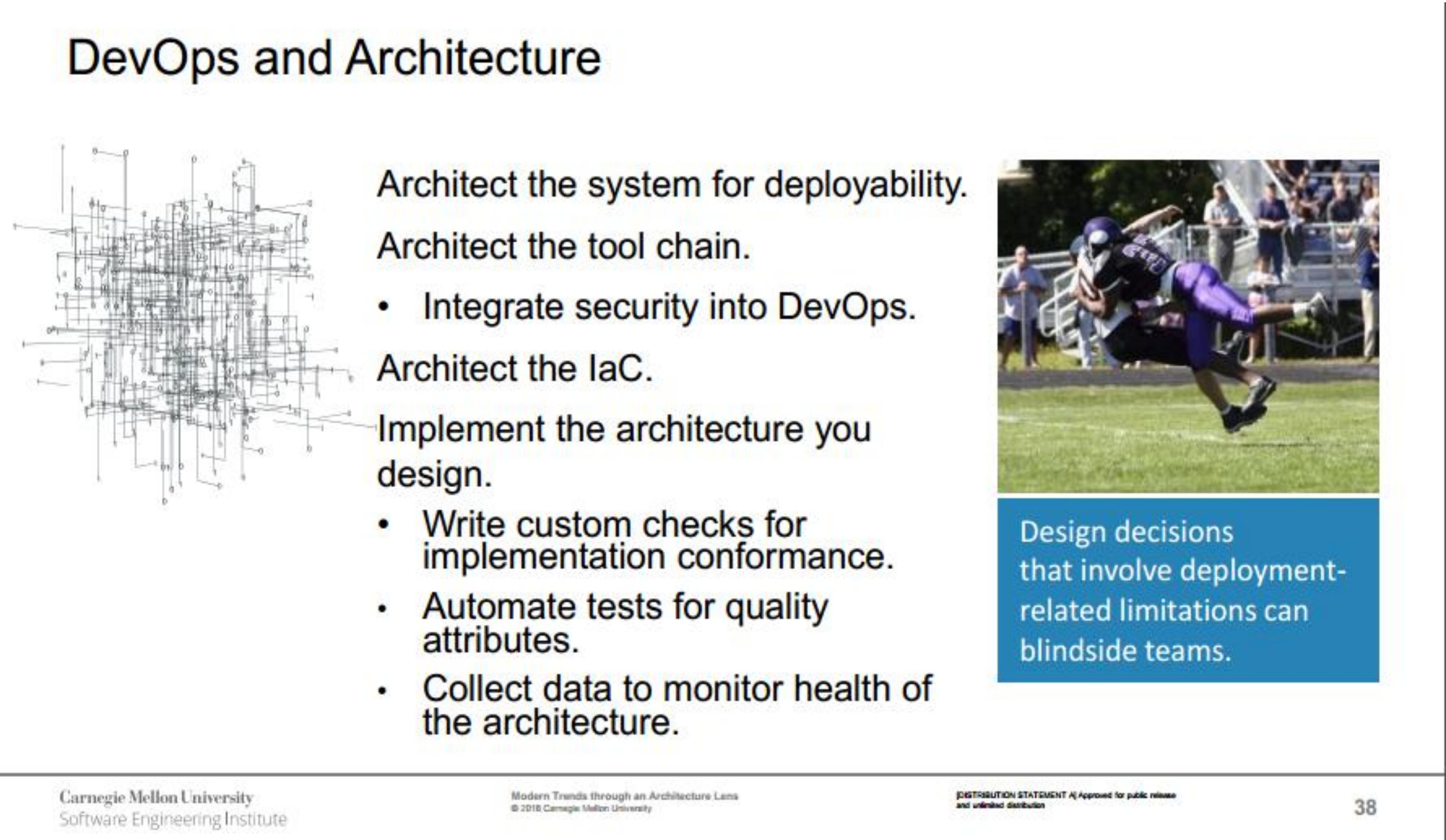
DevOps

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

DEV
OPS
CREATE
PLAN
VERIFY
PACKAGE
RELEASE
CONFIGURE
MONITOR

DevOps和软件架构



DevOps and Architecture

Architect the system for deployability.

Architect the tool chain.

- Integrate security into DevOps.

Architect the IaC.

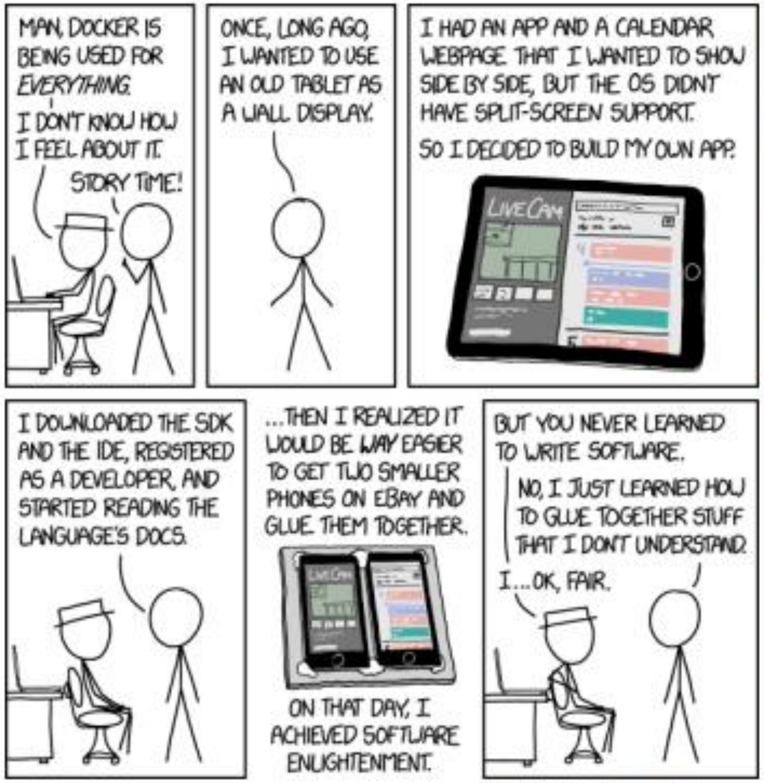
Implement the architecture you design.

- 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.

框架、代码库、容器等

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.
LIVE CAM
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.
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.

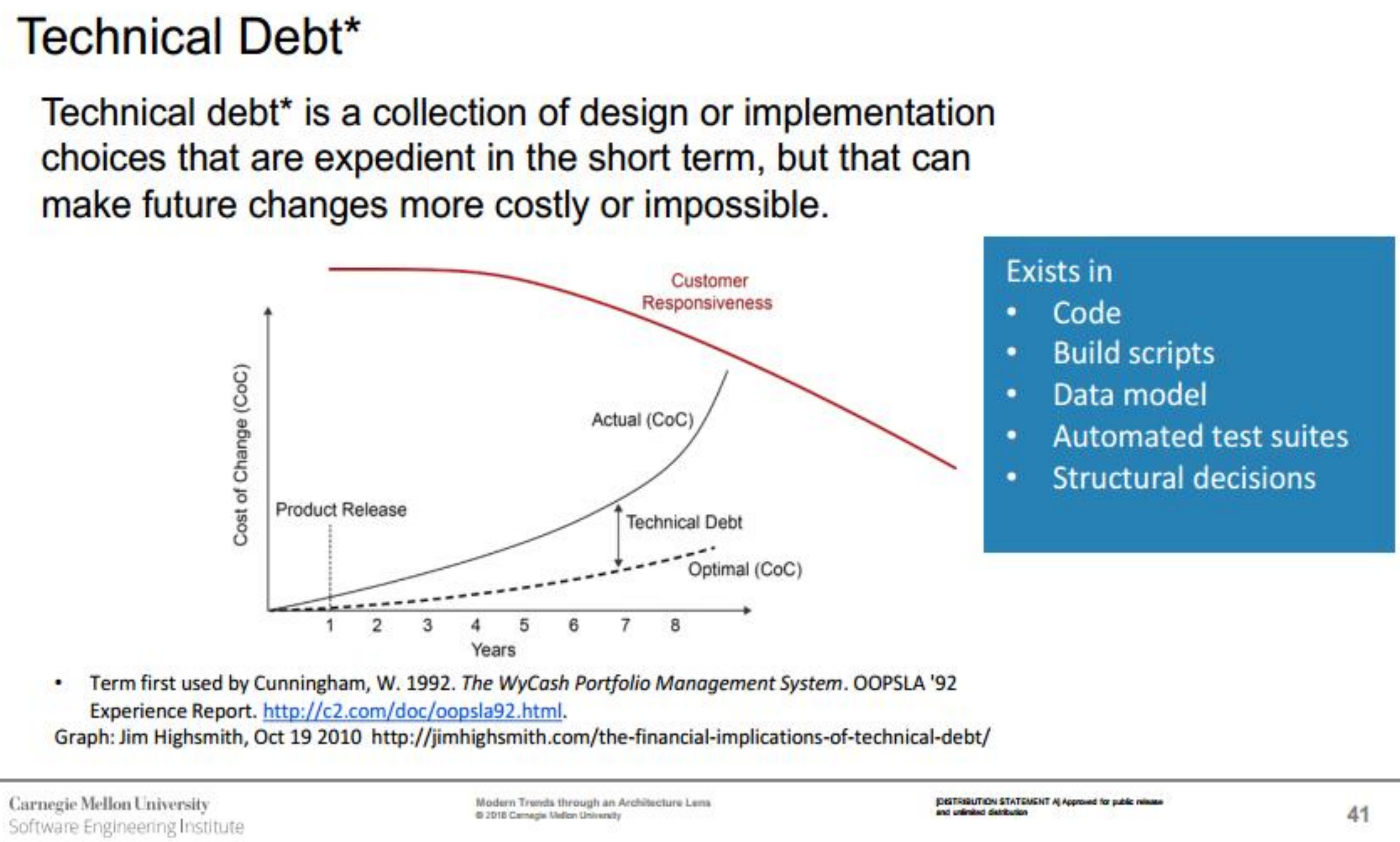
Rapid construction through assembly.

Architecture is implicit.

Undesirable behavior can occur.

Debilitating technical debt can occur.

技术债务



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

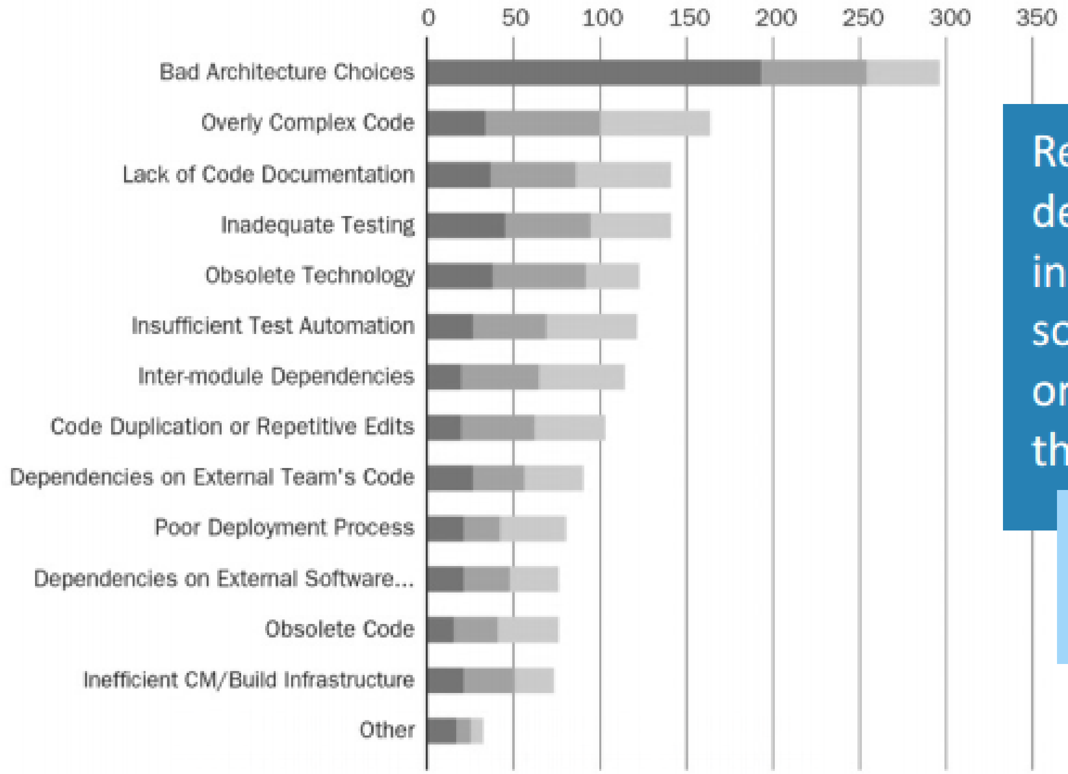
- Code
- Build scripts
- Data model
- Automated test suites
- Structural decisions

- 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/

软件架构和设计平衡问题

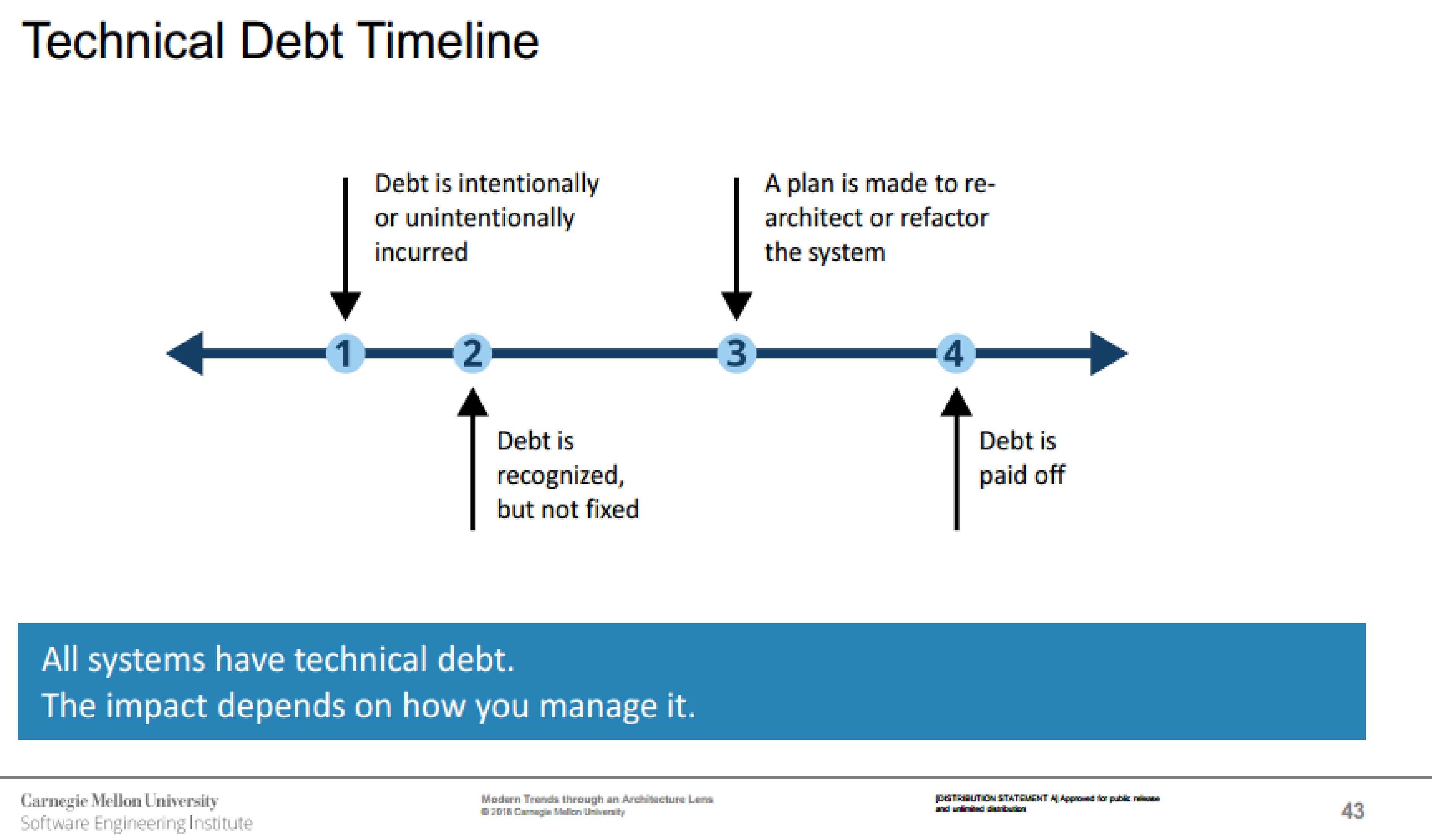
Software Architecture and Design Tradeoffs Matter



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

技术债务时间线



Technical Debt Timeline

Debt is intentionally or unintentionally incurred

Debt is recognized, but not fixed

A plan is made to re-architect or refactor the system

Debt is paid off

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

Carnegie Mellon University
Software Engineering Institute

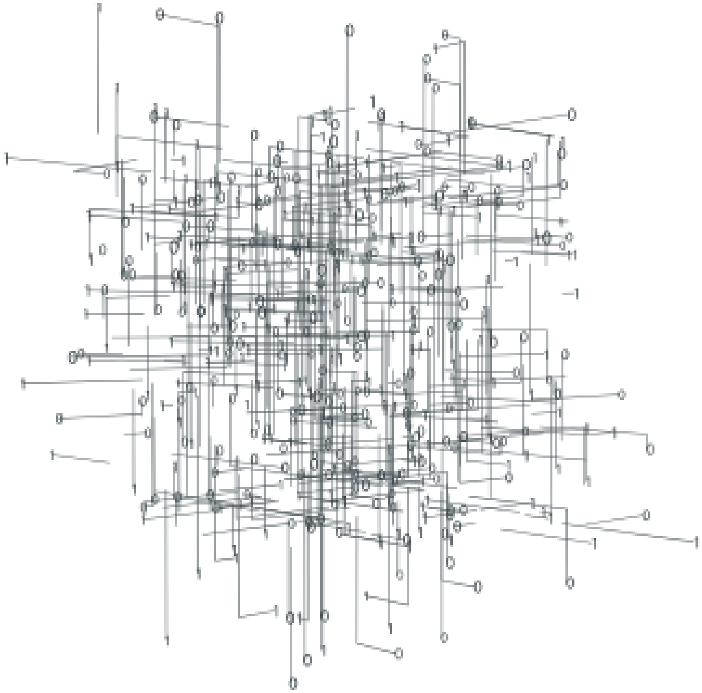
Modern Trends through an Architecture Lens
© 2018 Carnegie Mellon University

[DISTRIBUTION STATEMENT A] Approved for public release and unlimited distribution

43

软件架构需要考虑的因素

Net Sum Architectural Needs



Tradeoffs, decisions, structure persists.

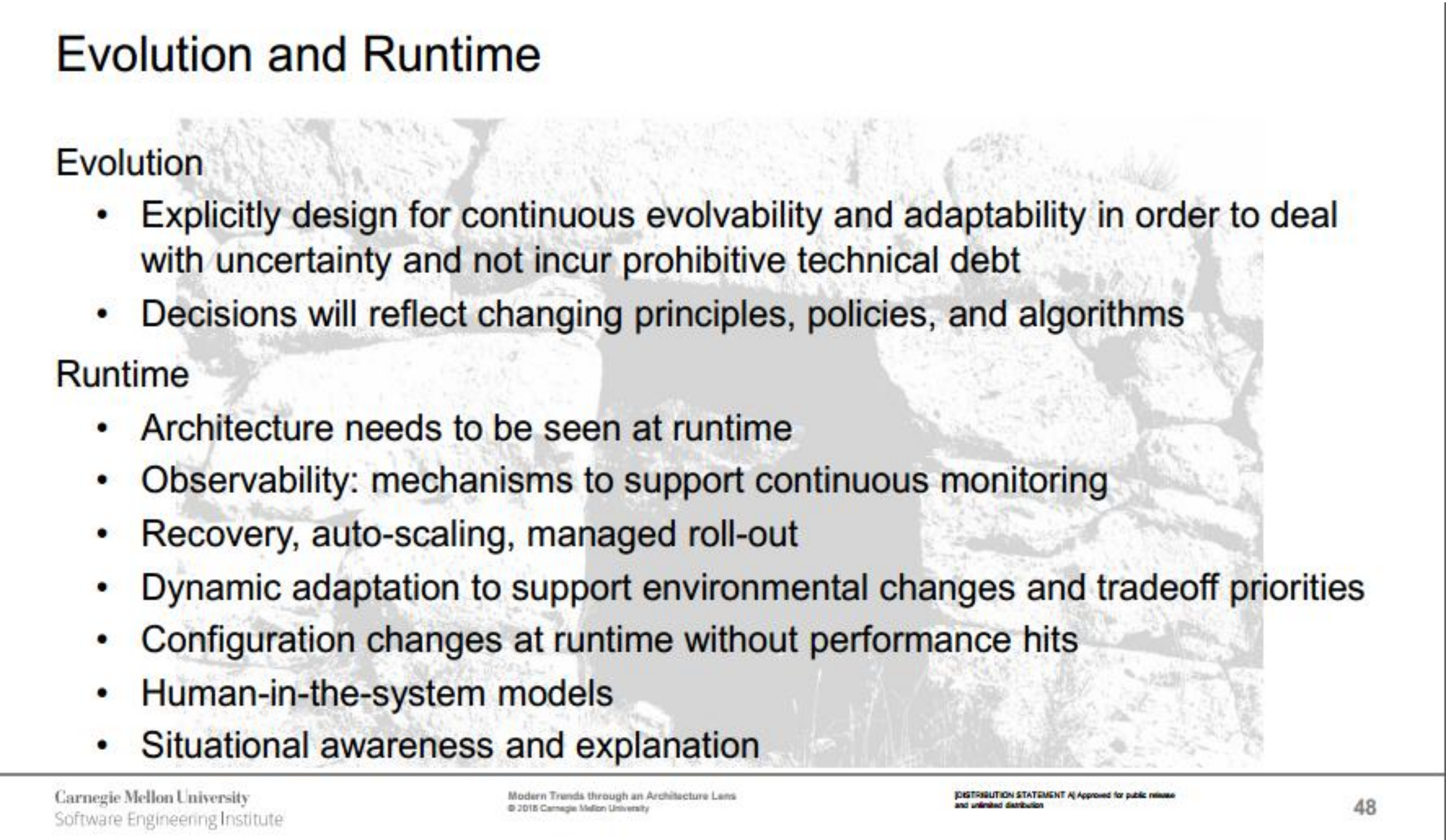
Security needs are heightened.

Different quality attributes at the fore.

New focus on

- Evolution
- Runtime
- Data
- ML
- Automation

改进和实时



Evolution and Runtime

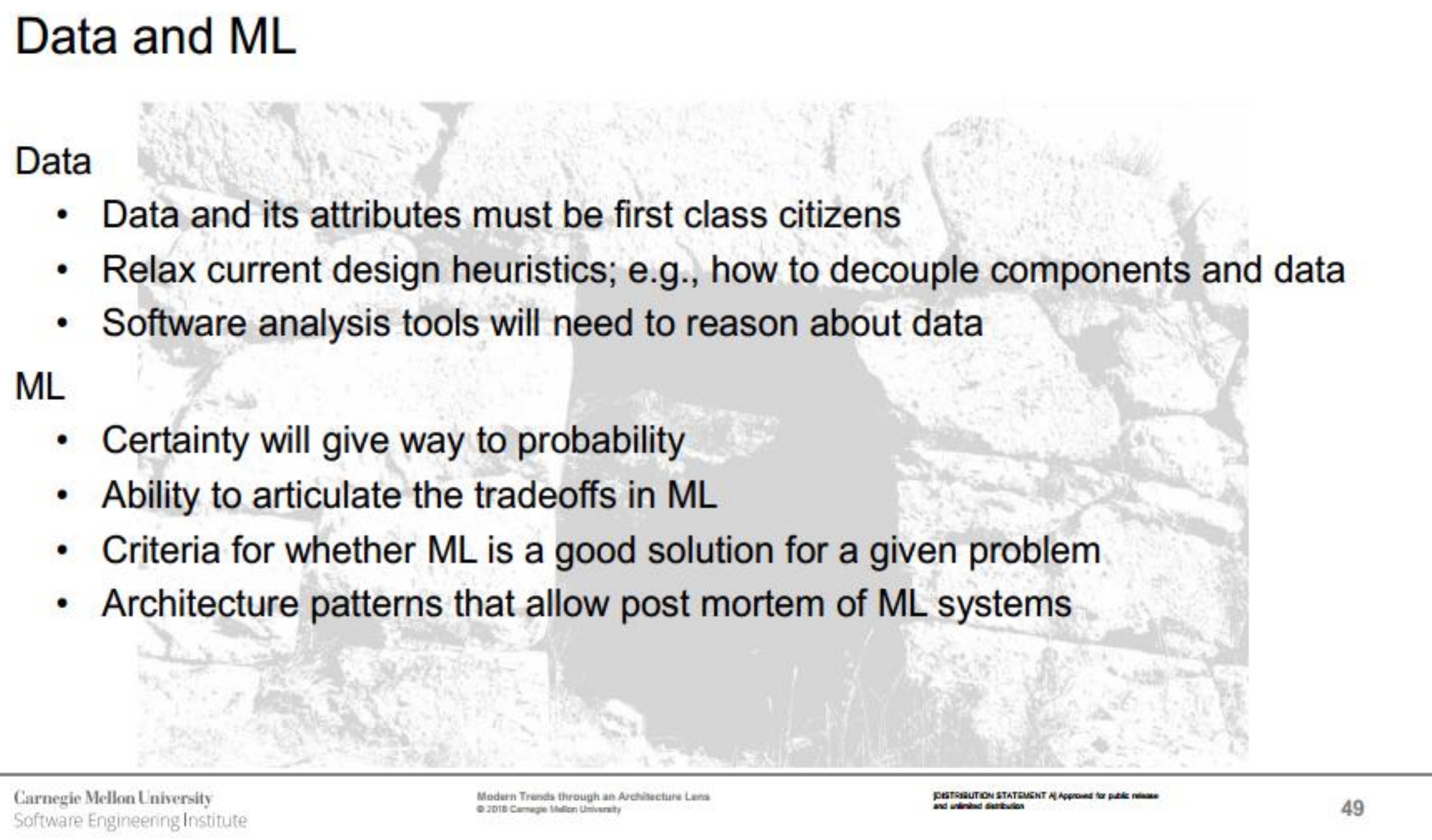
Evolution

- 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

数据和机器学习



Data and ML

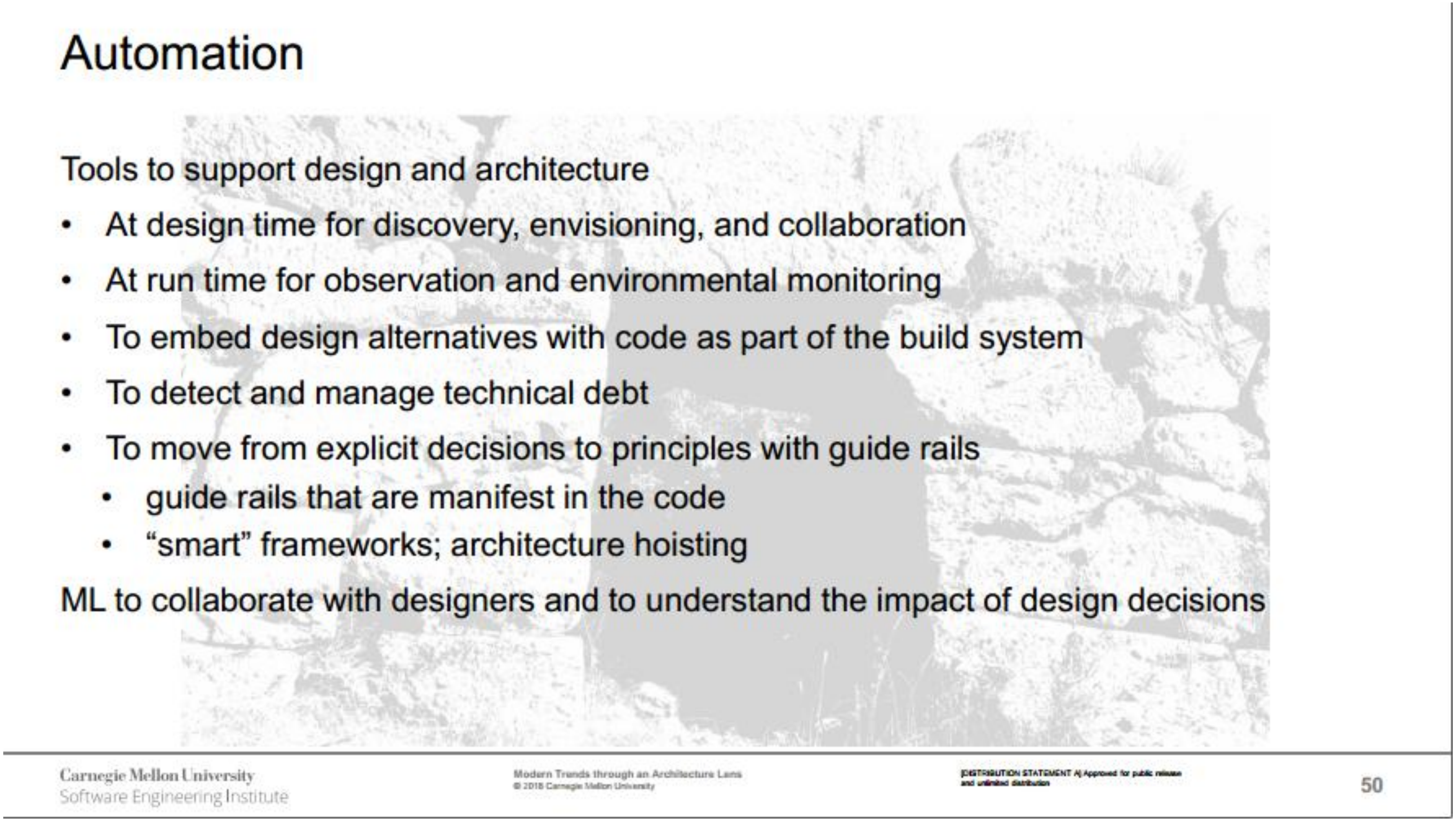
Data

- 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

自动化



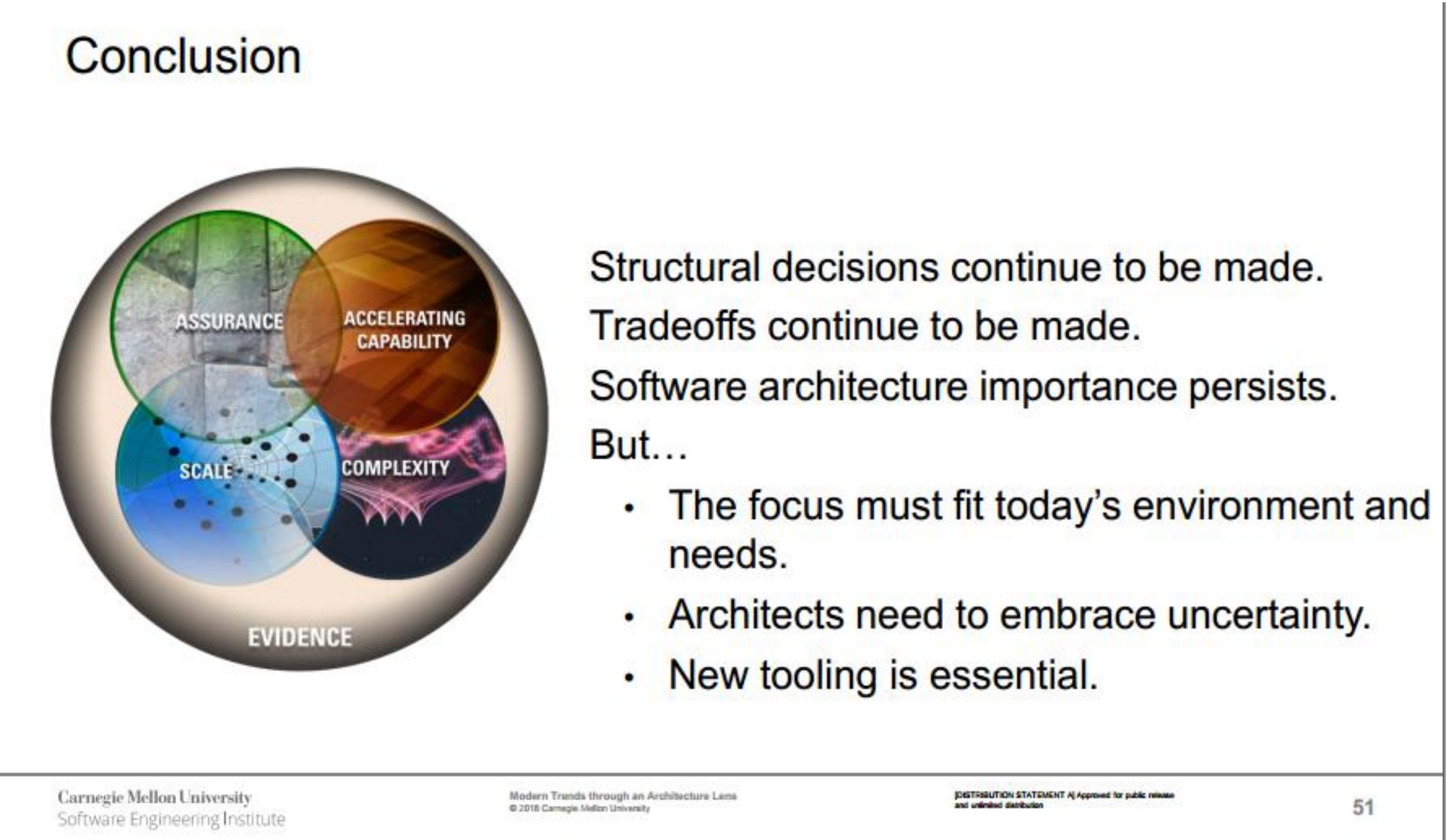
Automation

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

结束语



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.