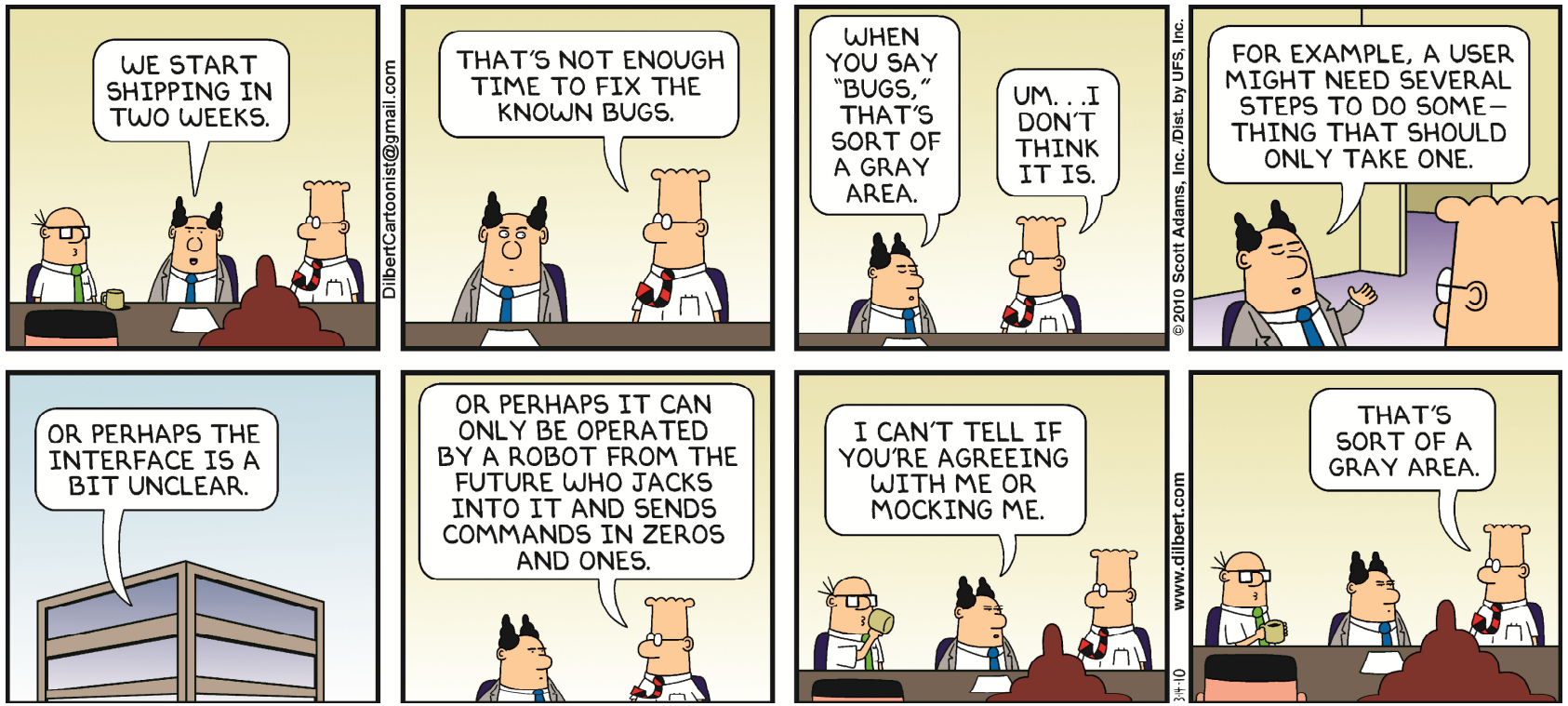
Agile Development and Software Architecture: Understanding Scale and Risk

Software Engineering Institute
Carnegie Mellon University
Pittsburgh, PA 15213

Robert L. Nord
SSTC, April 2012
In collaboration with Felix Bachmann, Ipek Ozkaya, Rob Wojcik, Bill Wood

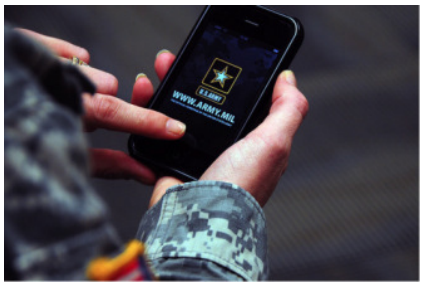
Software Engineering Institute | Carnegie Mellon

The challenge -1



Tradeoffs and their dependencies must be supported by both Agile software development and architecture practices.

The challenge -2



First, more capabilities

Then, more infrastructure

need to monitor to gain insight into life-cycle efficiency

First, more infrastructure

Then, more capabilities

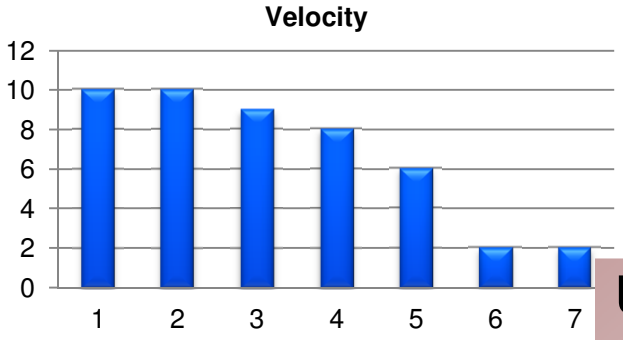
underestimated re-architecting costs

neglected cost of delay to market

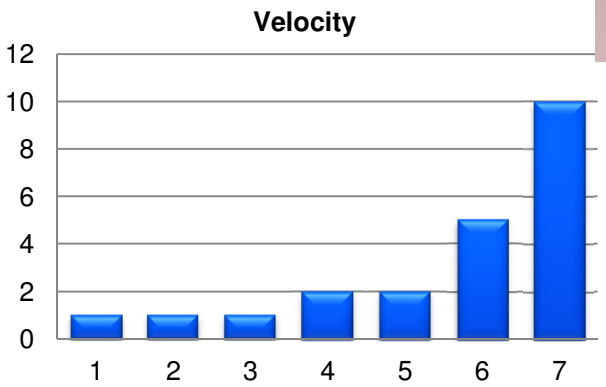
Brown, N., Nord, R., and Ozkaya, I. "Enabling Agility Through Architecture." *Crosstalk 23,* 6 (Nov./Dec. 2010): 12–17.

Increased visibility into delivery

Focus on Priority

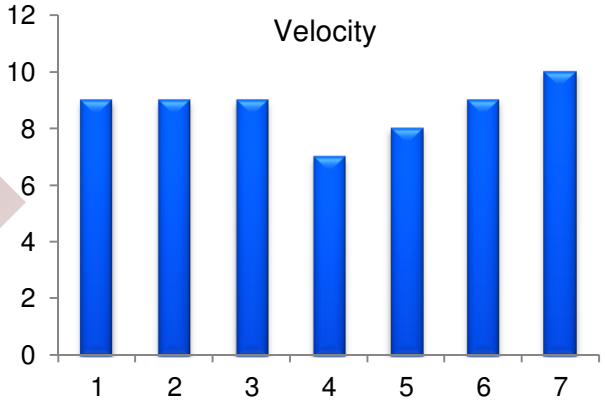


Focus on Cost



Use metrics to monitor & select development tasks

Focus on Integrated Value



Agenda

Symptoms of failure

Concepts of scale

Root-cause analysis

Architectural tactics that can help

Typical problems and their resolution

Symptoms of failure

- Teams spend almost all of their time fixing defects, and new capability development is continuously slipping.
- Integration of products built by different teams reveals that incompatibilities cause many failure conditions and lead to significant out-of-cycle rework.
- Progress toward meeting milestones is unsatisfactory because unexpected rework causes cost overruns and project completion delays.

Today's Challenge Dealing with Large Organizational Changes

Yesterday's Agile:
Teams got better at building software

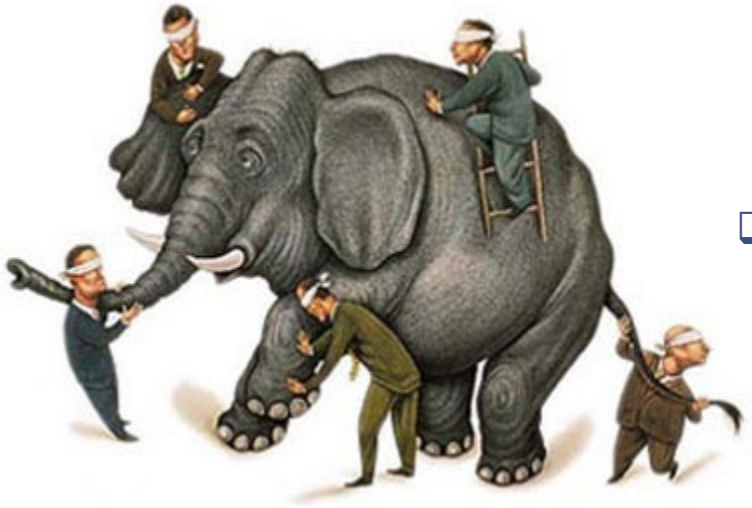
- Velocity
- Reliability
- Code quality
- Improvement
- Cohesion

Today's Agile
Moving the rest of the business

- Timelines have changed
- Collaboration is critical
- Priorities are larger than the development team
- Value needs clearer definition

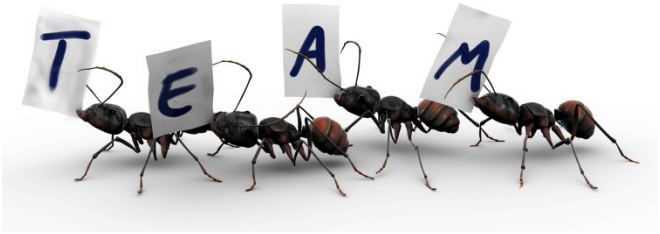
Grant, T. "Navigate the Future of Agile and Lean." *Forrester*, January 10, 2012.

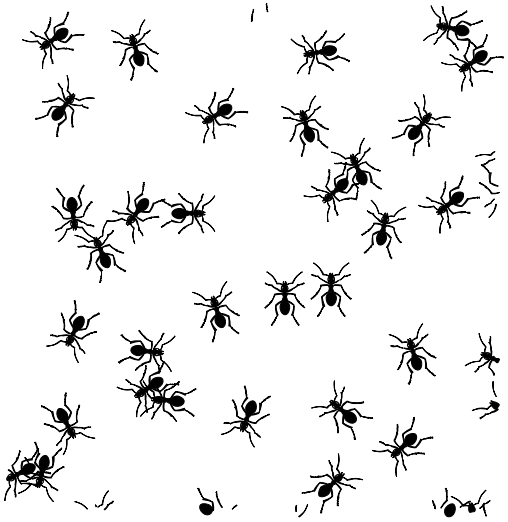
A closer look at scale: Scope



- Is the project in a new domain or technology?
- Does the project have new requirements such as standards compliance, system testing, and integration lab environments?
- Is there a need to align systems engineering and software development activities?

A closer look at scale: Team





- Are there multiple teams that need to interact, both internal and external to the organization?

at are the dependencies
ween the work products of
:em and software engineers?

:s the end-to-end delivery of
ures require resources from
tiple teams?

A closer look at scale: Time

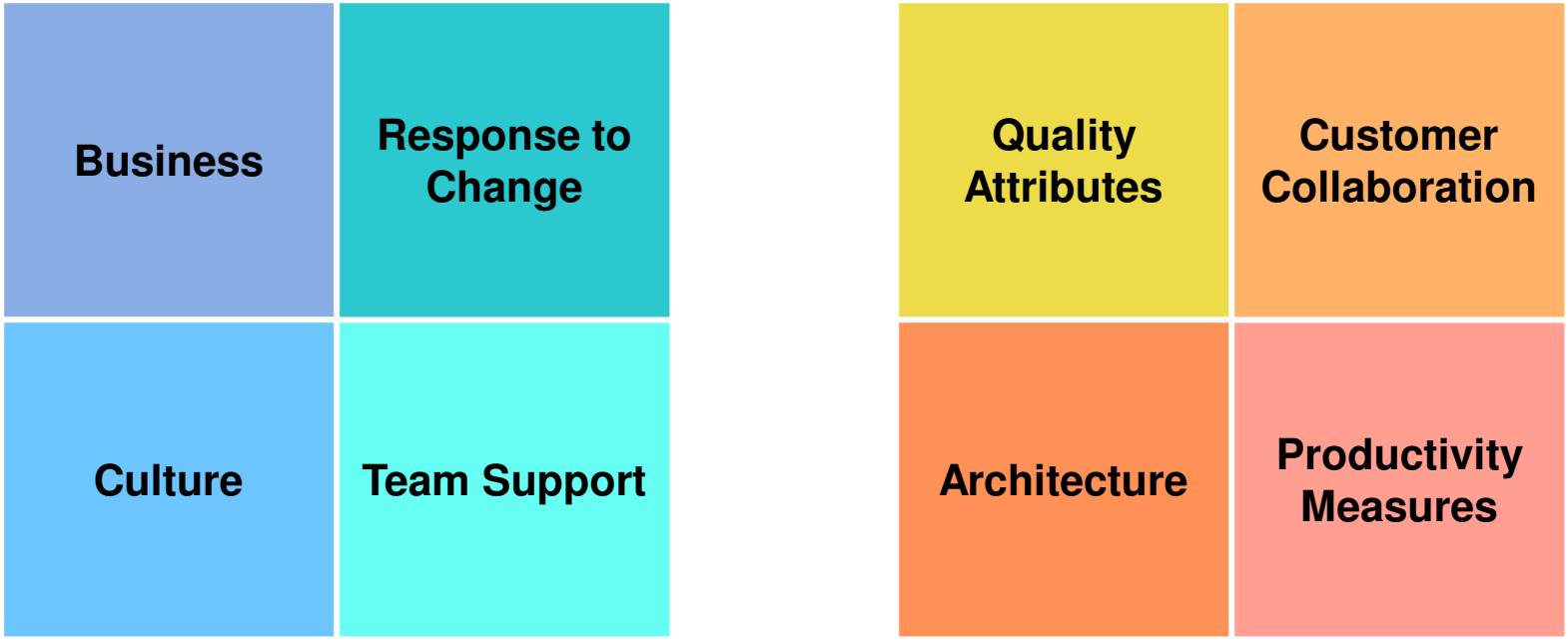
- Does the work require different schedule constraints for releases?
- How long is the work product expected to be in service?
- How important are sustainability and evolution?

Polling question

How would you describe your development context?

1. We follow one Agile methodology as closely as we can
2. We use a mix of Agile methodologies
3. We mix Agile and non-Agile techniques
   a. Waterfall and agile hybrid
   b. Architecture and agile principles
4. We do not apply any Agile method

Root-cause analysis



Understand symptoms by investigating Agile software development and software architecture principles and practices in the context of the product and the project.

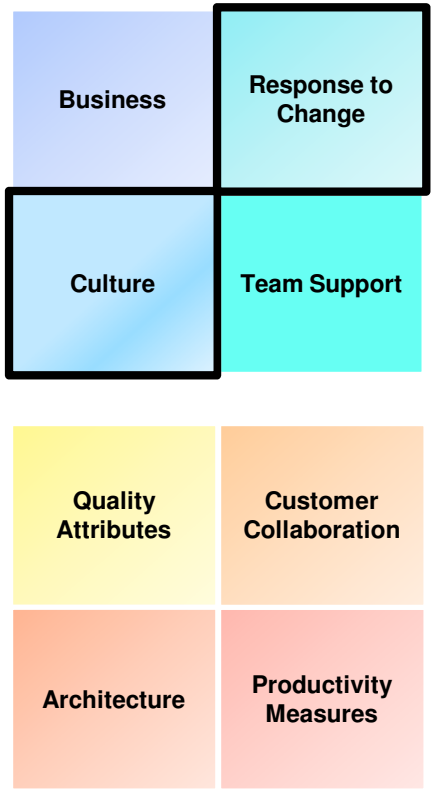
Root-cause analysis

Response to change

- Dynamic environment and changing requirements
- Necessary technology and processes
- Impact of uncertainty

Culture

- Skills and knowledge and clear responsibilities.
- Clear communication among teams
- High-level management support



Root-cause analysis

Quality attributes

- Quality attribute requirements tied to business goals
- Quality attribute analysis
- Measurement environment

Architecture

- Evidence that the architecture satisfies quality attribute requirements
- Architectural issues (*e.g.*, technical debt)
- Timeline of critical decisions

Architectural tactics to consider

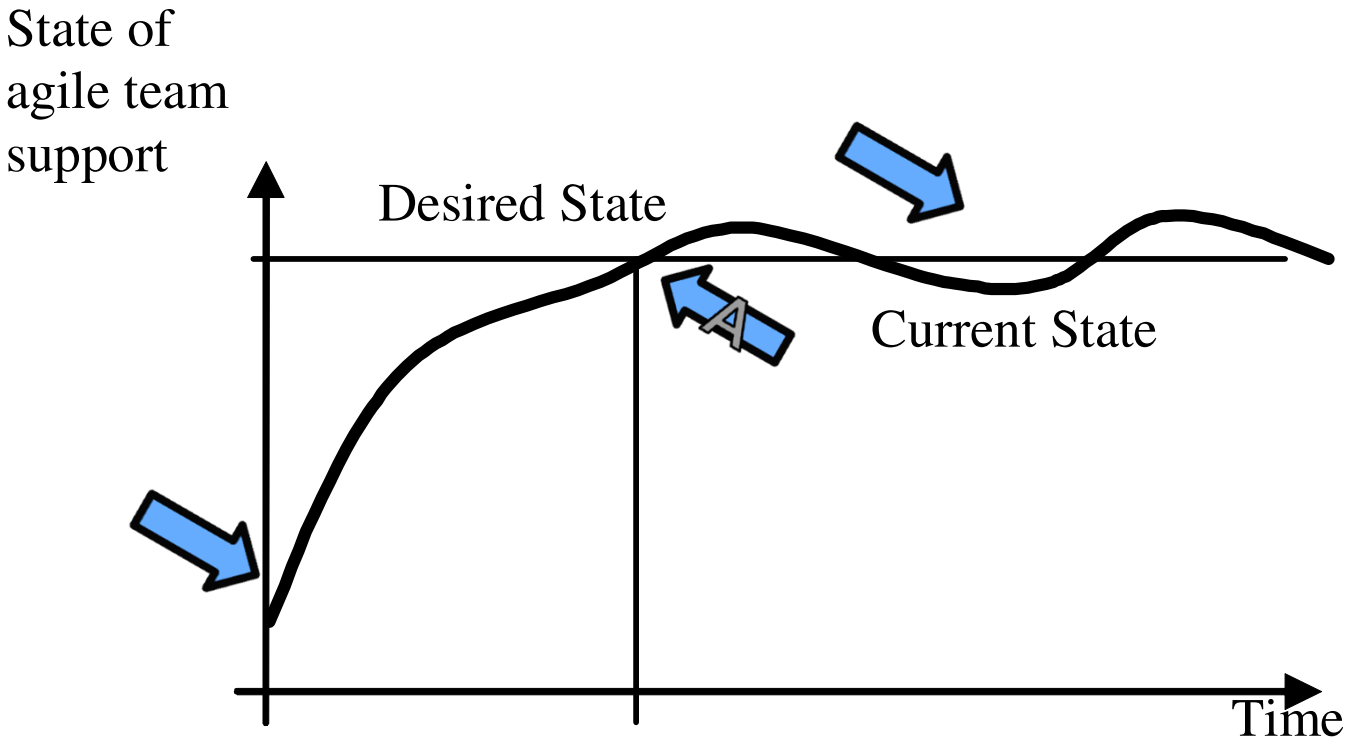
Align feature and system decomposition.

Create an architectural runway.

Use matrix teams and architecture.

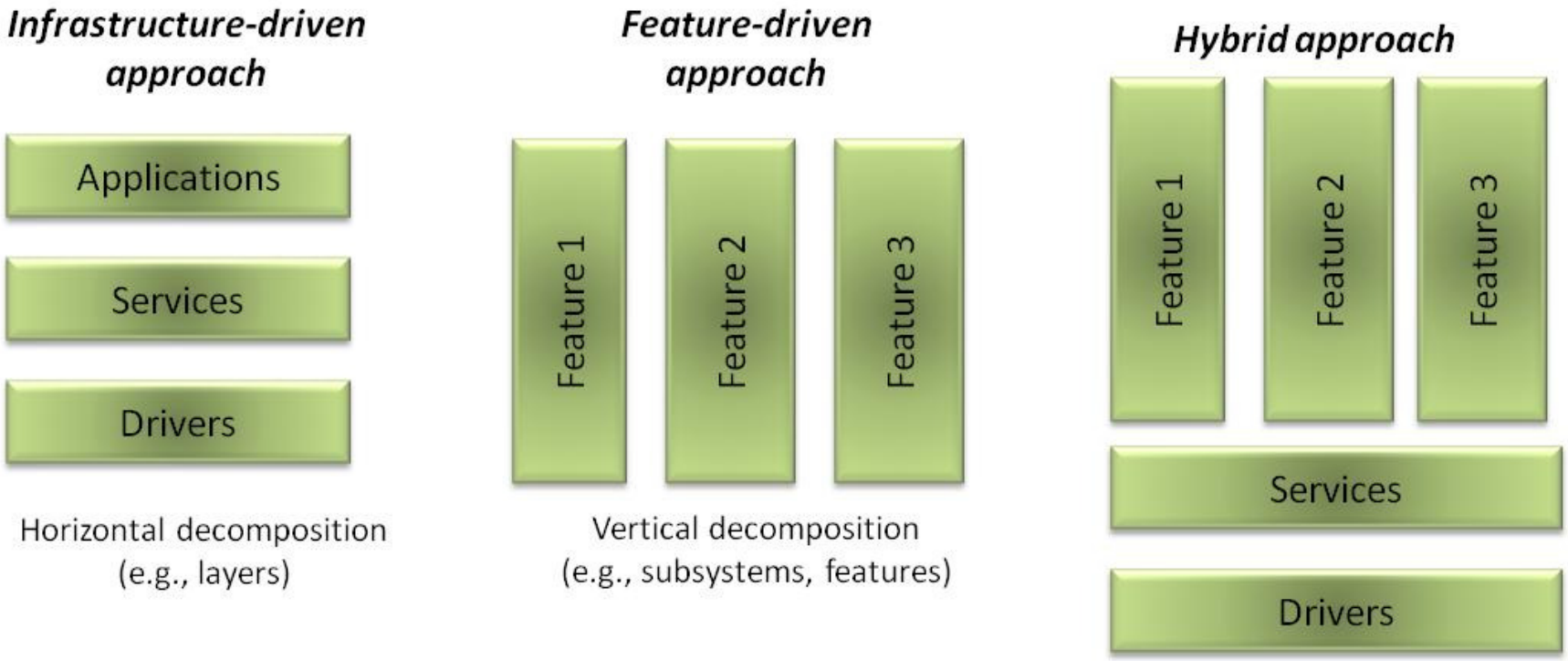
Support for Development Teams Over Time

Desired state that enables agile teams to quickly deliver releases that stakeholders value.

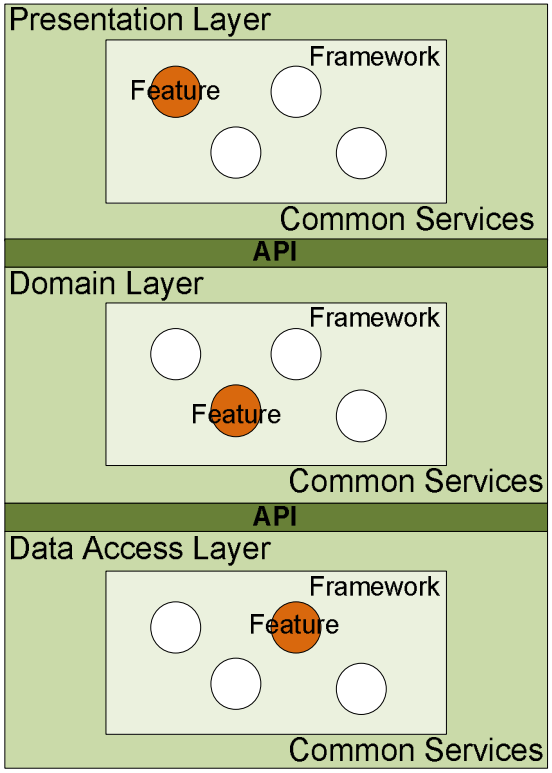


Align feature and system decomposition

Tension between high-priority features (vertical decomposition) versus common reusable services (horizontal decomposition)



Align feature and system decomposition



Layered architecture with frameworks

Dependencies between stories and architectural elements

enables staged implementation of infrastructure in support of achieving stakeholder value.

Dependencies among elements

Low-dependency architectures are a critical enabler for scaling up Agile development.[1]

Dependencies among stories

High-value stories may require implementation of lower value stories as precursors.[2]

Decouple teams and architecture to ensure parallel progress as the number of teams increases.

1. Poppendieck, M., and Poppendieck, T. *Leading Lean Software Development.* Addison-Wesley Professional, 2009.
2. Denne, M., and Cleland-Huang, J. *Software by Numbers*. Prentice Hall, 2003.

Create an architectural runway

The architectural runway provides the degree of architectural stability to support the next *n* iterations of development.

In a Scrum project environment, the architectural runway may be established during Sprint 0.

Sprint 0 might have a longer duration than the rest of the sprints.

The bigger the system, the longer the runway.
Principles of Agile Architecture[1]

1. Leffingwell, D. *Agile Software Requirements*, Addison-Wesley, 2011

Use matrix teams and architecture

In its simplest instantiation, a Scrum development environment consists of:

- a single co-located, cross-functional team
- with skills, authority, and knowledge to specify requirements
- and architect, design, code, and test the system.

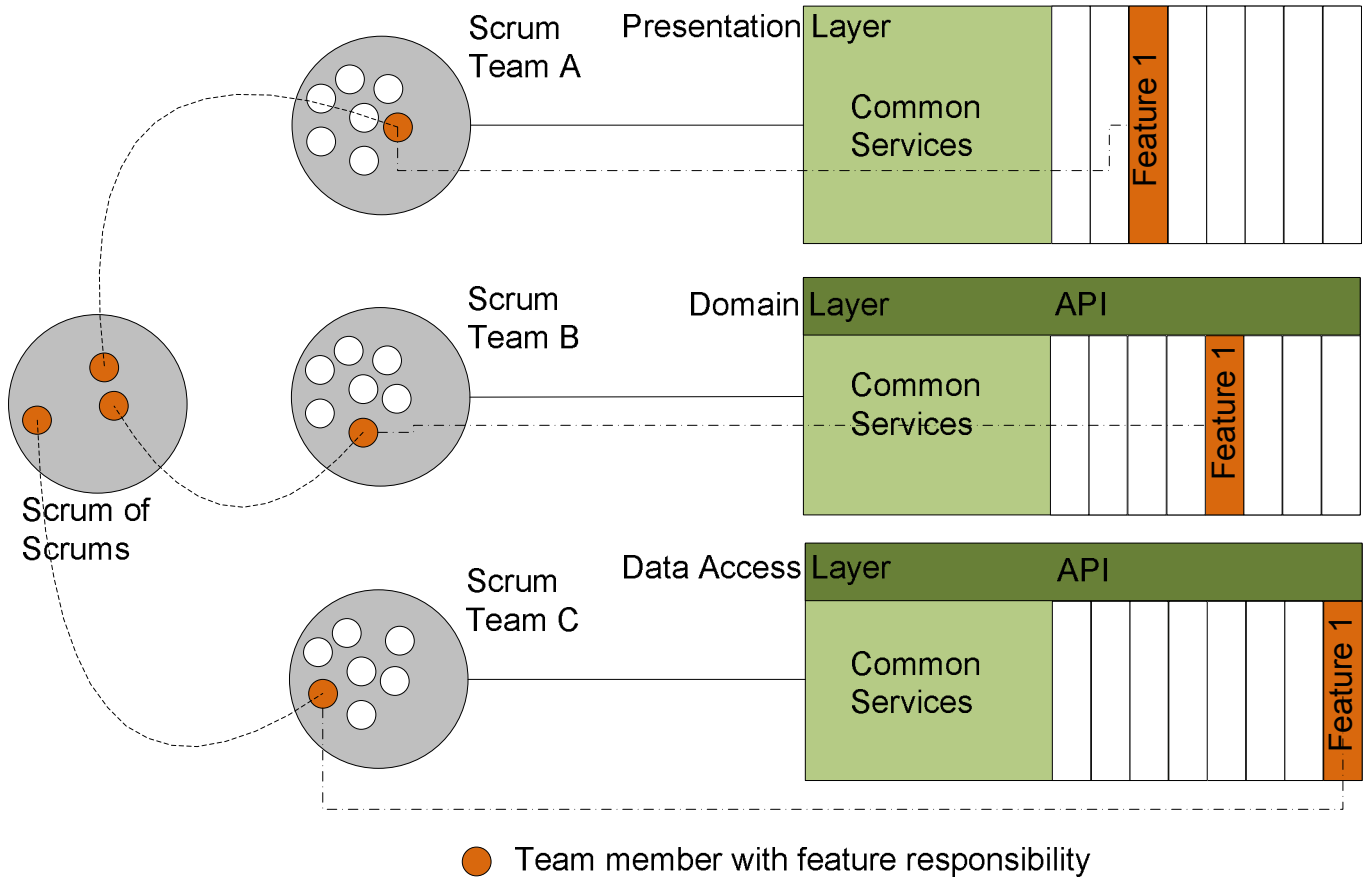
As systems grow in size and complexity, the single-team model may no longer meet development demands.

Strategies to scale up the overall agile development organization include:

- replication of team structure and responsibilities
- hybrids of vertical and horizontal team organization.

Use matrix teams and architecture

Feature development in parallel



Root-cause analysis: Typical problem 1

Symptom

- Scrum teams spend almost all of their time fixing defects, and new feature development is continuously slipping.

Root-cause

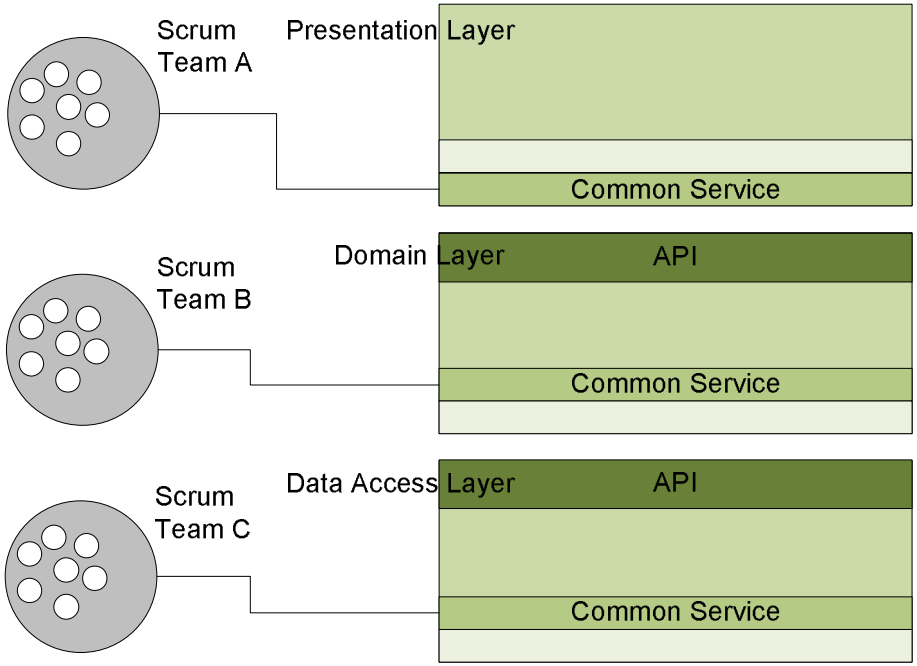
Inability to manage scope and time at scale

- Initial focus was "general" rather than "product specific."
  - Time pressure to deliver became the top priority.
  - The team delivered an immature product.
  - A plethora of variation parameters interact detrimentally.
- There are three different cycles:
  - Customer release (annually, many variants)
  - IV&V Testing (quarterly, 4 variants)
  - Developmental (monthly, 1 variant)

Solution

Stabilize the architecture.

- Add guidelines over time.
- Reduce the number of "variant parameterizations."
- Make everyone play from the same sheet music.
- Postpone adding new features.



Replan the release cycles and time boxes.

Revisit the testing strategy and team assignments against variants.

Root-cause analysis: Typical problem 2

Symptom

- Integration of products built by different Scrum teams reveals that incompatibility defects cause many failure conditions and lead to significant out-of-cycle rework.

Root-cause

Inability to manage teams at scale

- Cross-team coordination is poor, even though there are many coordination points and much time spent.
- Different teams have different interpretations of interfaces.
- The product owner on each team does not see the big picture.
- A mismatch exists between the architecture and development.

Solution

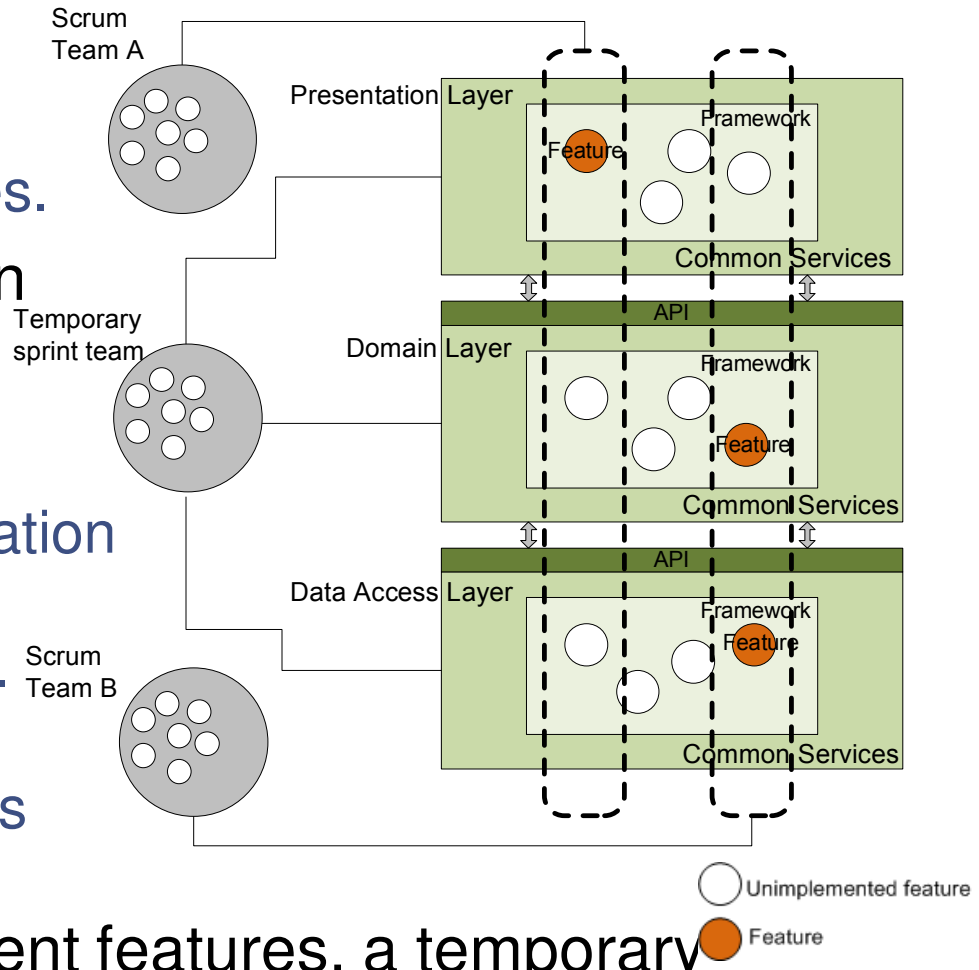
Stabilize to remove failures.

- Postpone adding new features.

Identify and collapse common services across teams.

Use an architectural runway.

- infrastructure allows incorporation of anticipated requirements without excessive refactoring.
- as important as requirements epics that drive the company's vision forward.



Teams are assigned to different features, a temporary team prepares layers and frameworks for future teams.

Final thoughts

No one tactic alone can take any project to success.

Systematic root-cause analysis is essential for understanding risks arising in large-scale software development.

There are different aspects of scale to be managed with different approaches, such as scope, team, and time.

Embracing the principles of Agile software development and software architecture provide improved visibility of project status and better tactics for risk management.

References

Ambler, S. *The Agile Scaling Model (ASM): Adapting Agile Methods for Complex Environments*. IBM developerWorks, 2009. https://www.ibm.com/developerworks/mydeveloperworks/blogs/ambler/entry/agile_scaling_model?lang=en

Brown, N., Nord, R., and Ozkaya, I. "Enabling Agility Through Architecture." *Crosstalk 23,* 6 (Nov./Dec. 2010): 12–17.

Denne, M., and Cleland-Huang, J. *Software by Numbers*, Prentice Hall, 2003.

Kruchten, P. "What Color Is Your Backlog?" Agile Vancouver talk, 2009. http://files.me.com/philippe.kruchten/vuldw4

Larman, C., and Voddle, B. *Scaling Lean & Agile Development.* Addison-Wesley, 2009.

Leffingwell, D. *Scaling Software Agility*. Addison-Wesley, 2007. http://scalingsoftwareagility.wordpress.com/2008/09/09/enterprise-agility-the-big-picture-10-the-system-team

Poppendieck, M., and Poppendieck, T. *Leading Lean Software Development*. Addison-Wesley Professional, 2009.

Upcoming

Bachmann, F., Nord, R., Ozkaya, I.

Architectural Tactics to Support Rapid and Agile Stability, CrossTalk: The Journal of Defense Software Engineering, May/June 2012.

IEEE Software Special Issue on Technical Debt, Nov/Dec 2012.

Guest Editors: Philippe Kruchten, Robert Nord, and Ipek Ozkaya

Contact Information

Robert L Nord
Research, Technology, and System Solutions Program
Architecture Practices Initiative
Email: rn@sei.cmu.edu

Software Engineering Institute
Customer Relations
4500 Fifth Avenue
Pittsburgh, PA 15213-2612
USA

This material is based upon work supported by the Department of Defense under Contract No. FA8721-05-C-0003 with Carnegie Mellon University for the operation of the Software Engineering Institute, a federally funded research and development center.