Resolving Chaos Arising from Agile Software Development

Software Engineering Institute
Carnegie Mellon University
Pittsburgh, PA 15213

Author
Date

High Level Alternatives

Approach 1. Blame the Agile development process, fire the folks who are controlling it and revert to previous development processes

Approach 2. Assess why the current approach is chaotic, determine ways (processes, technology, personnel) to stabilize the development, and then continue

Agile Terms

Scrum

- Scrum lead, product owner, developers with appropriate skills

Feature Batch

- New features fixed until batch completion (prioritized)
- Test procedures are developed concurrently

Timebox

- Team agrees to implement the **batch** in this timebox (weeks)

Alignment

- Vertical — change all components for new features
- Horizontal — component cohesion and consistent

Developmental Rhythms

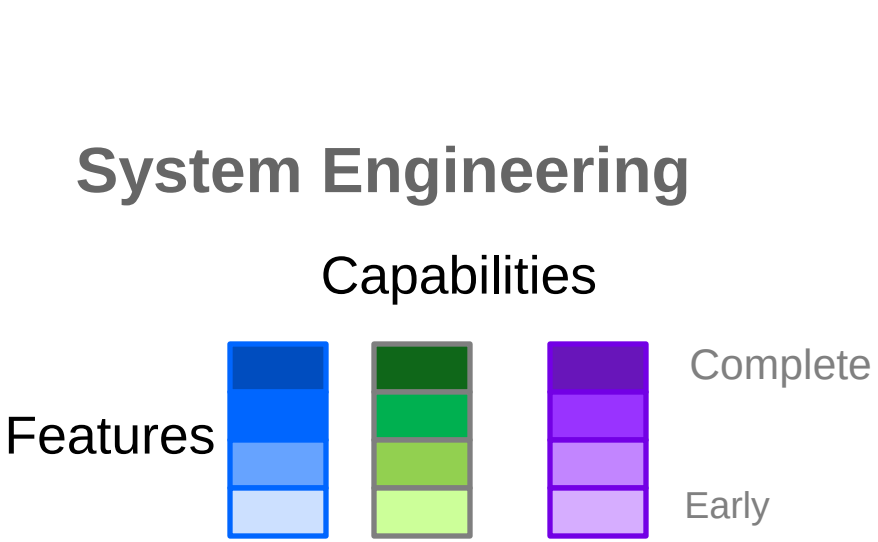
Planning Rhythms

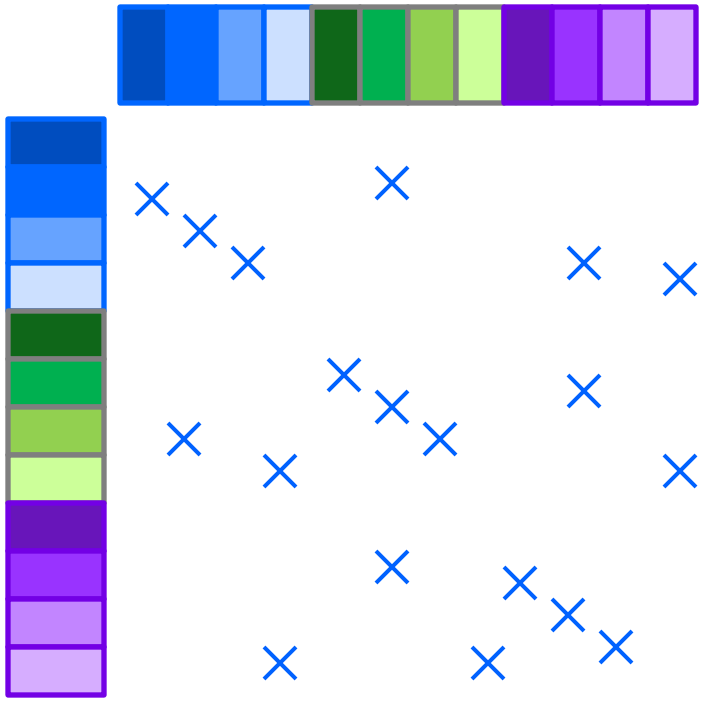
- Lifecycle (Program managers, System Engineers and Architects)
  - capabilities / features/ components/ tests for each milestone
  - relationships between them
- Milestone (System Engineers and Architects)
  - Capabilities, features, test scenarios, components built and integrated for each timebox within the milestone
  - describes the mappings between the above for each timebox
- Timebox (Architects, Development Managers)
  - allocates efforts to teams to accomplish the plan
  - Updates the plan to recover from : slippages, defects, unplanned workarounds

Implementation Rhythm (Architects and Implementers)

- Focuses on detailed work efforts by teams within timeboxes

Lifecycle Plan - System Engineer





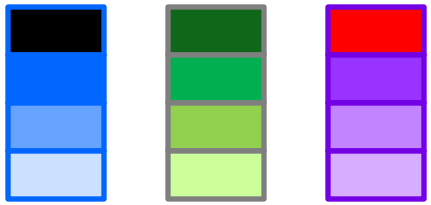
Matrix shows dependencies between capabilities

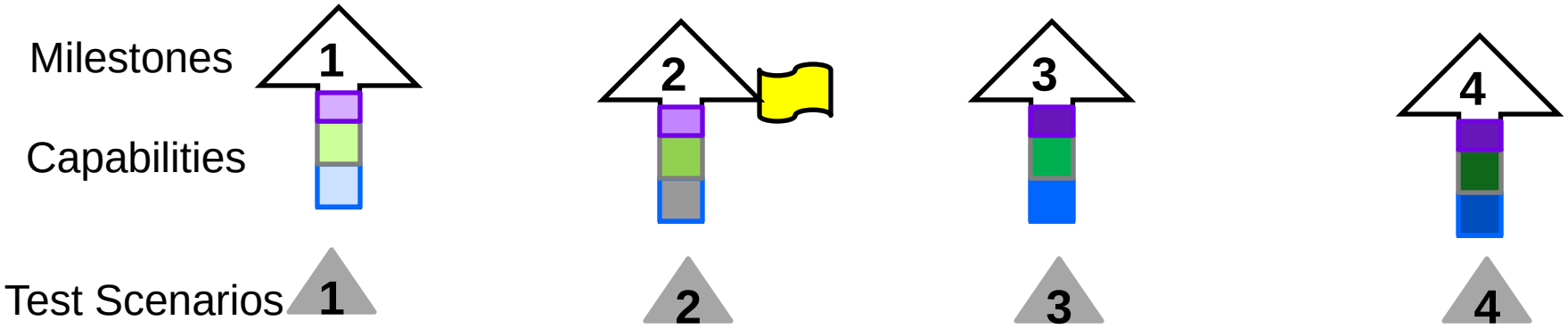
Lifecycle Plan - System Engineer

Initial System Engineering Plan

System Engineering

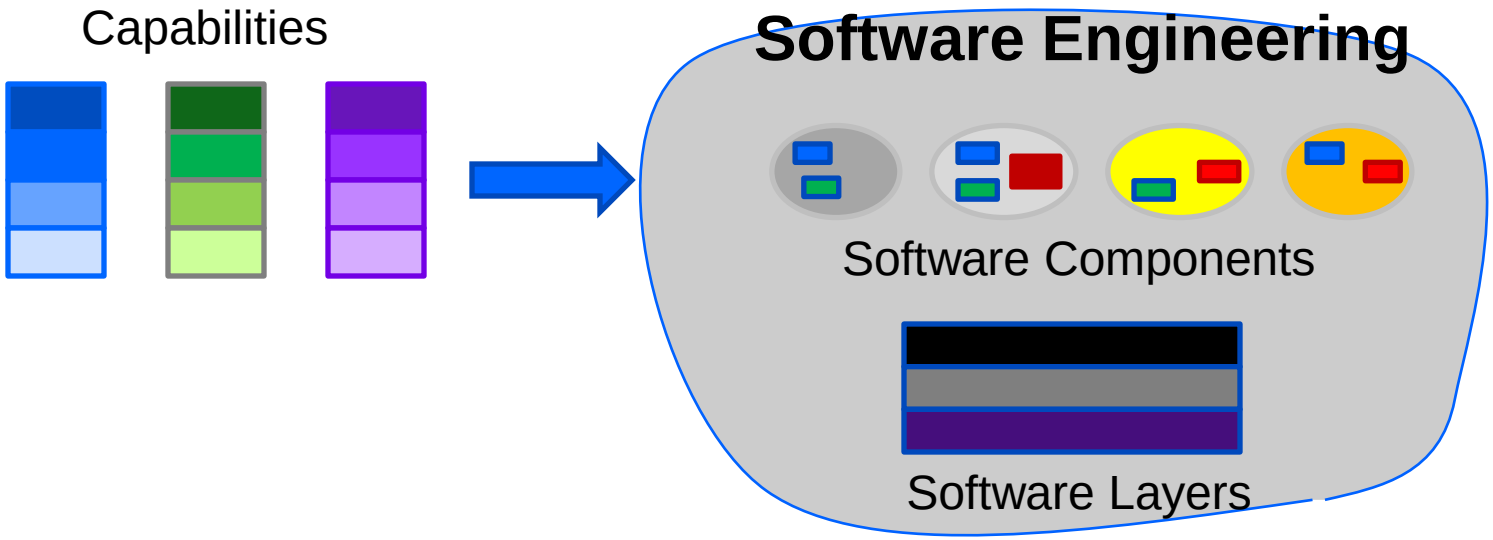
Capabilities



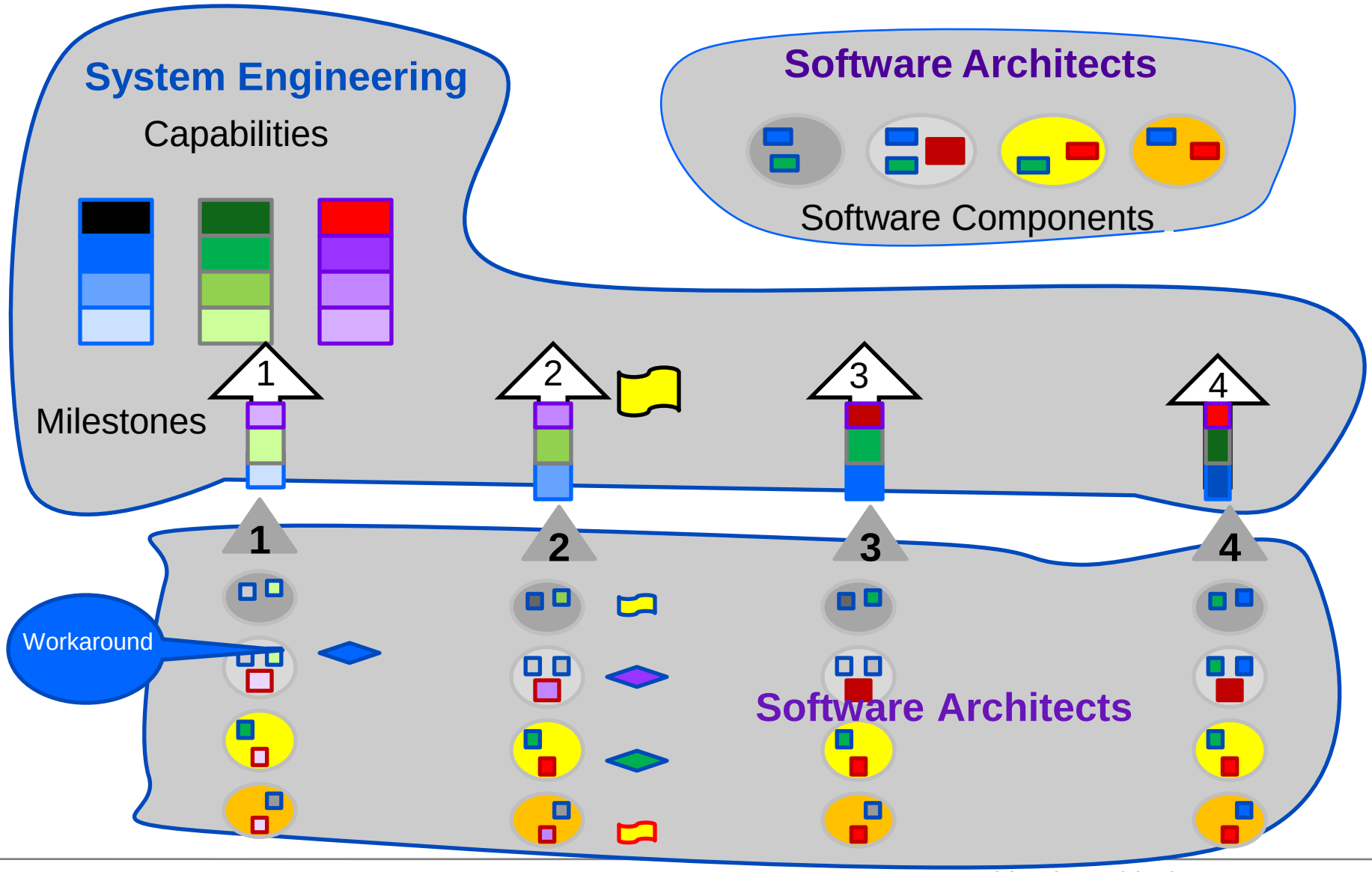


Lifecycle Planning - Software Architect

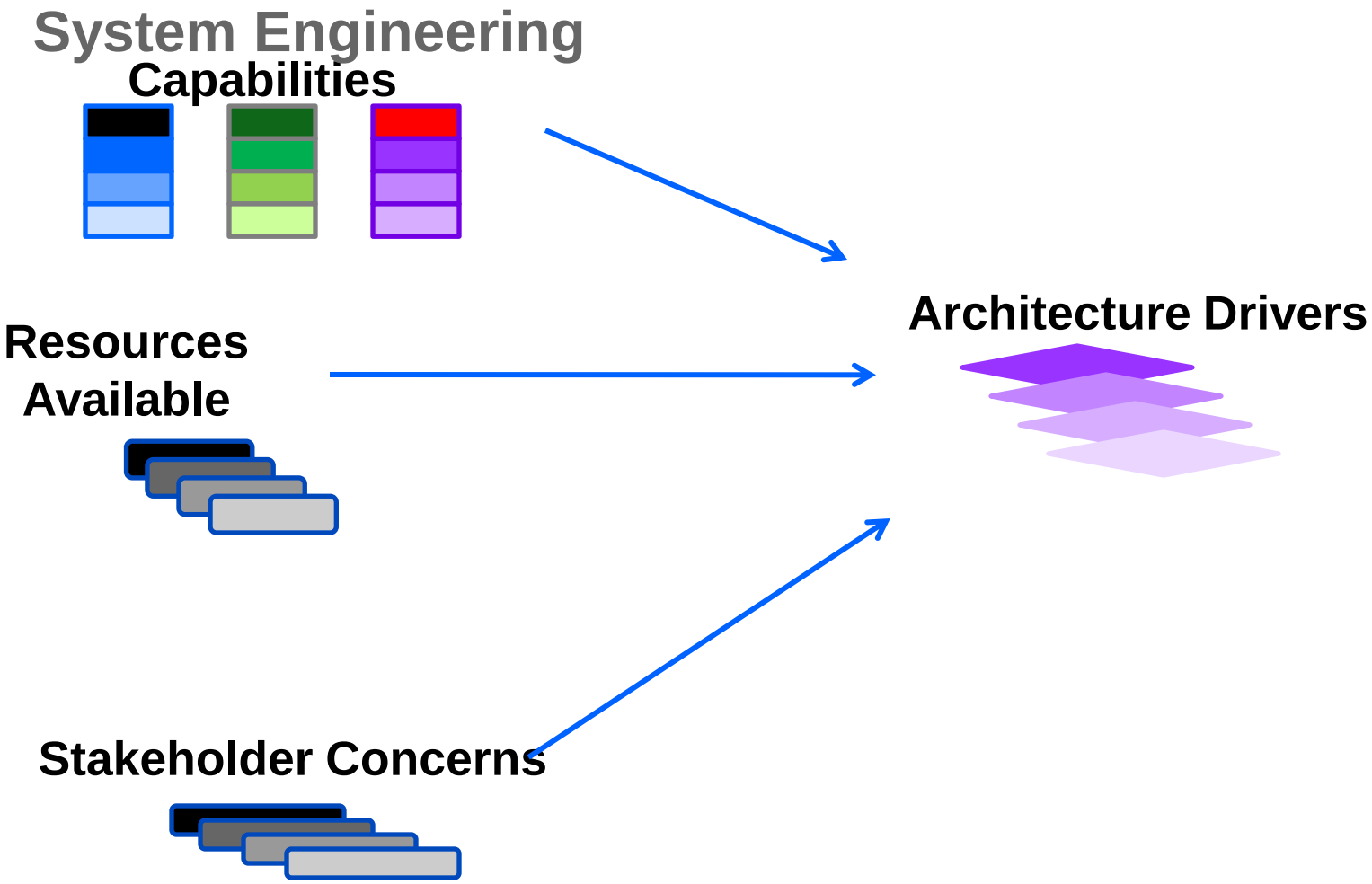
Mapping Capabilities to Software



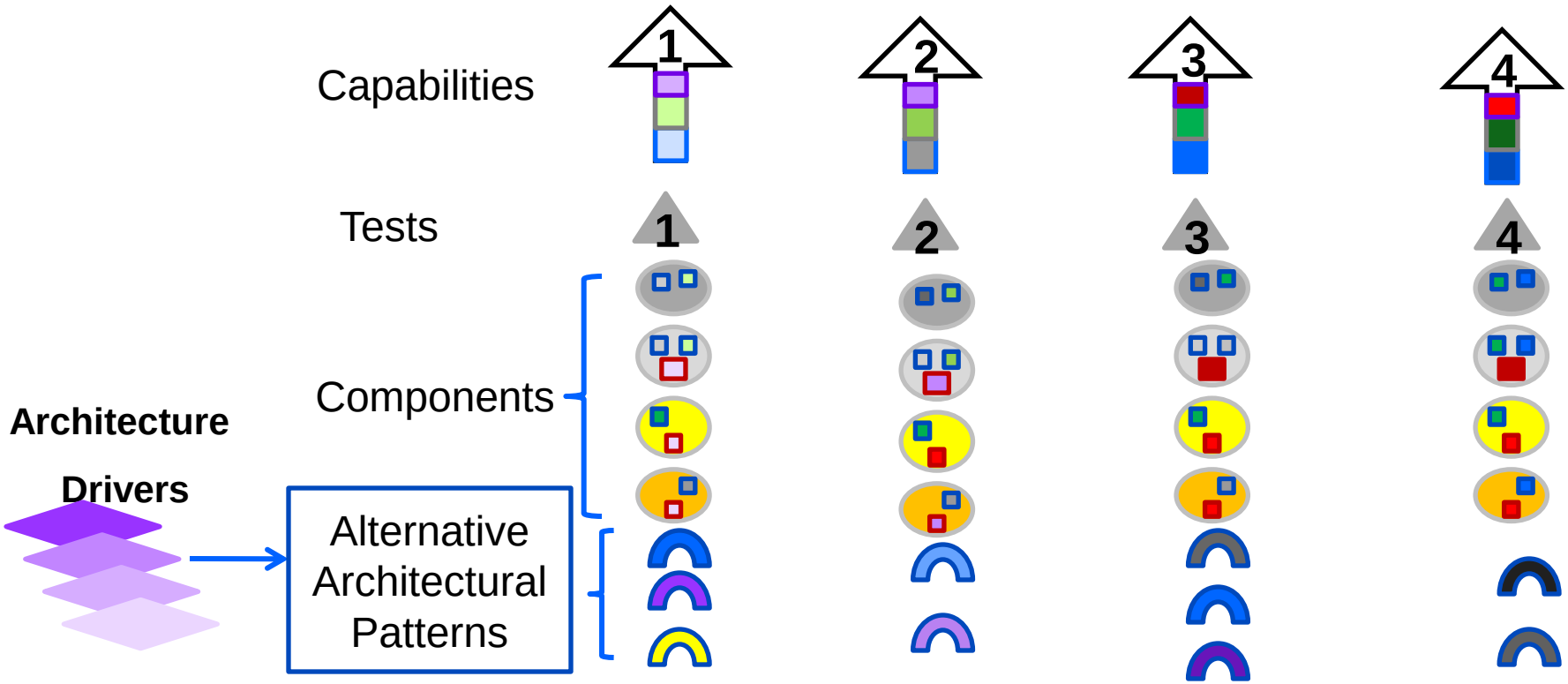
Life-Cycle Plan- SE and SA



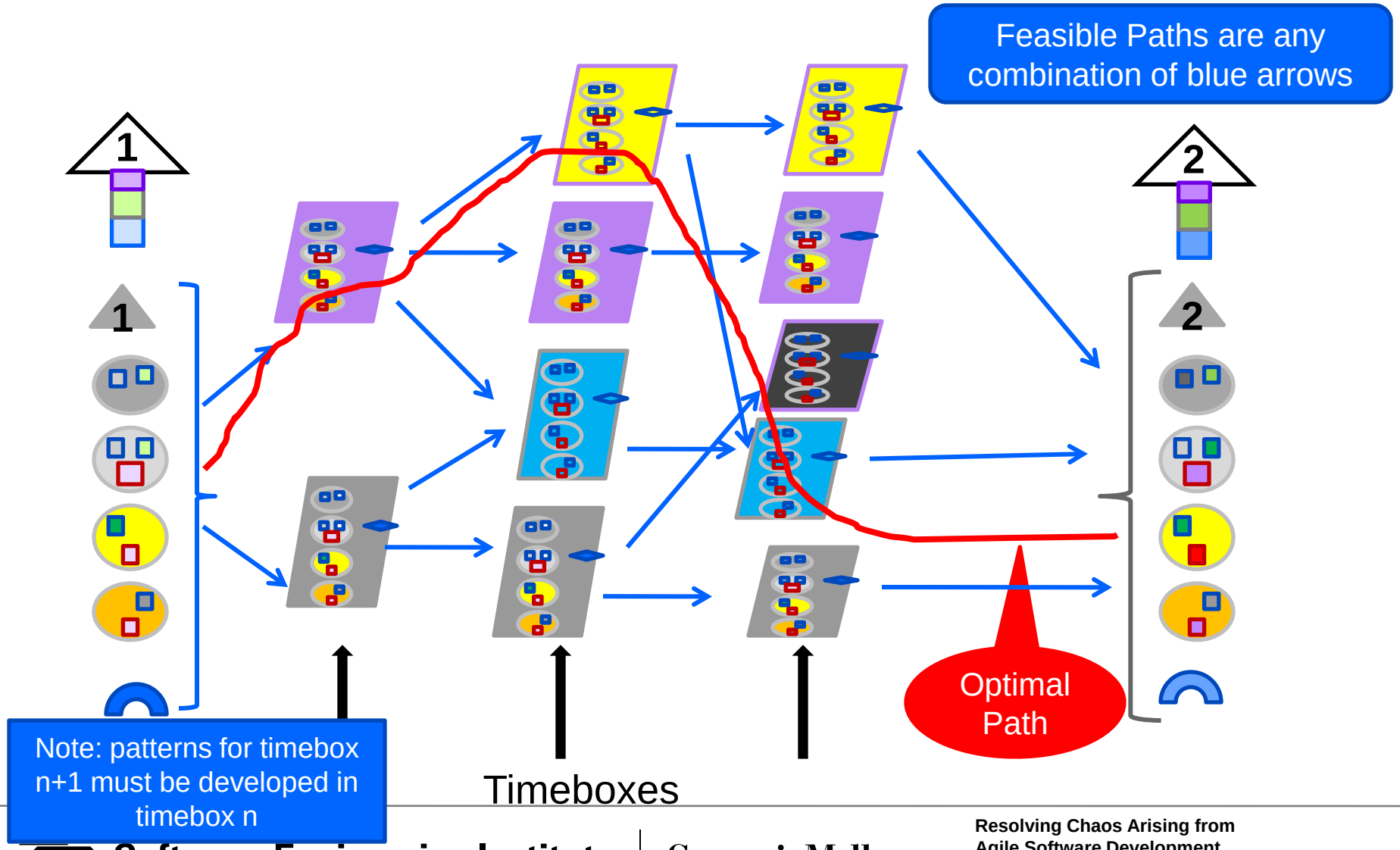
Life-Cycle Plan - SA - Alternatives



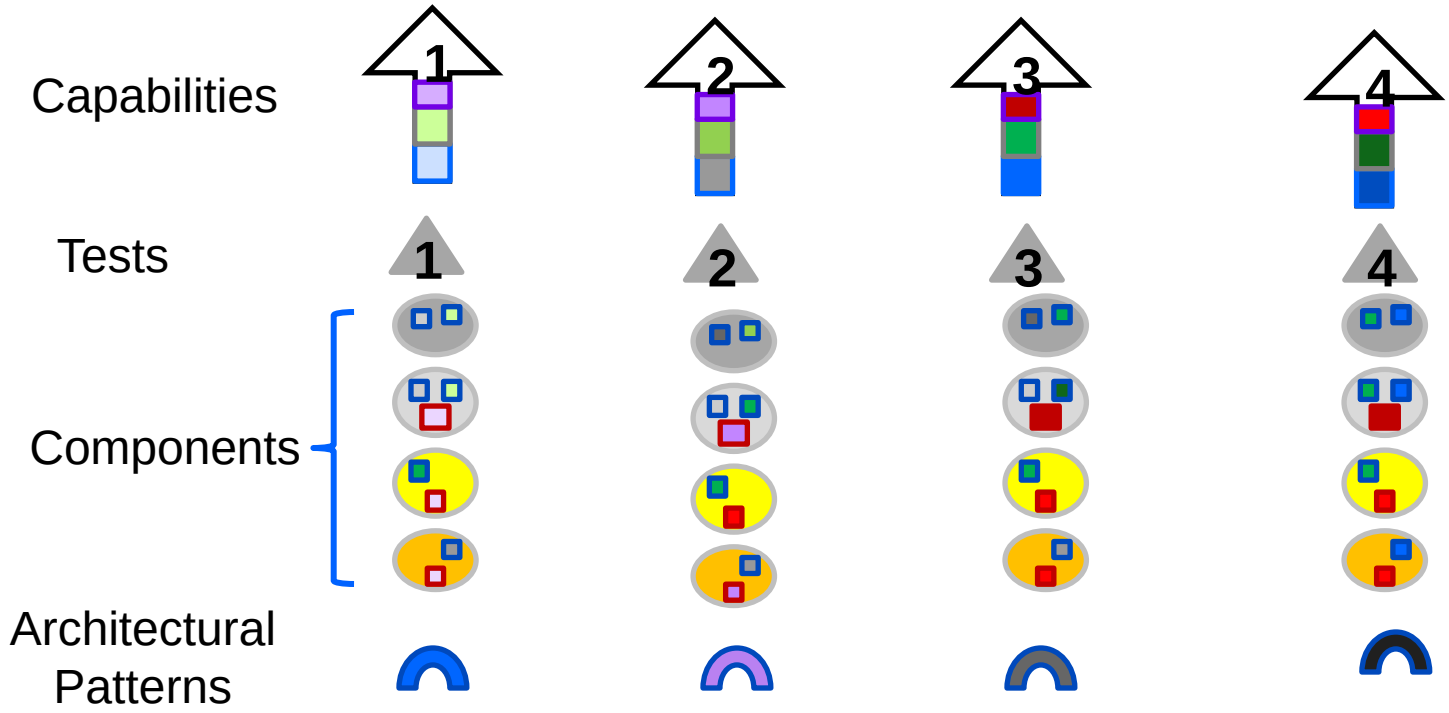
Architecture Driven Design (ADD)



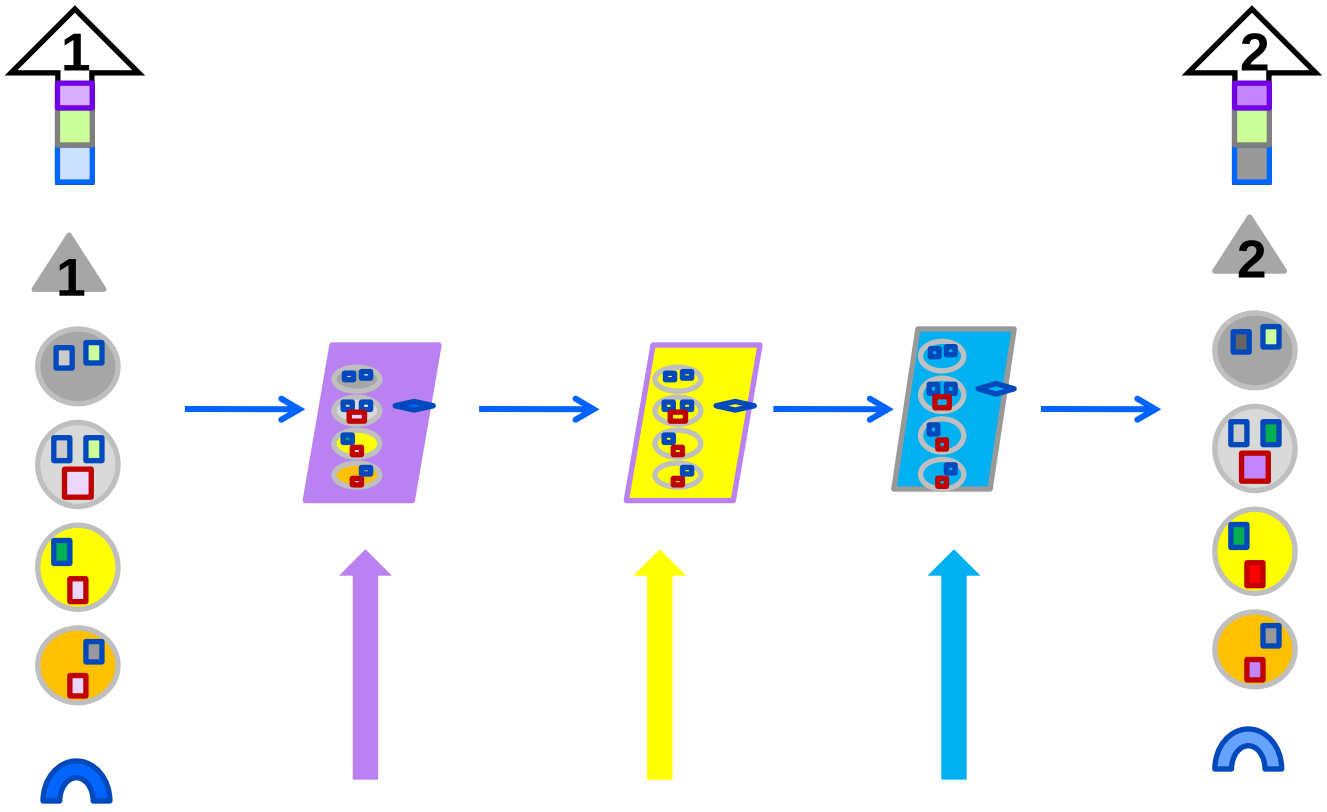
Milestone Plan- Alternatives



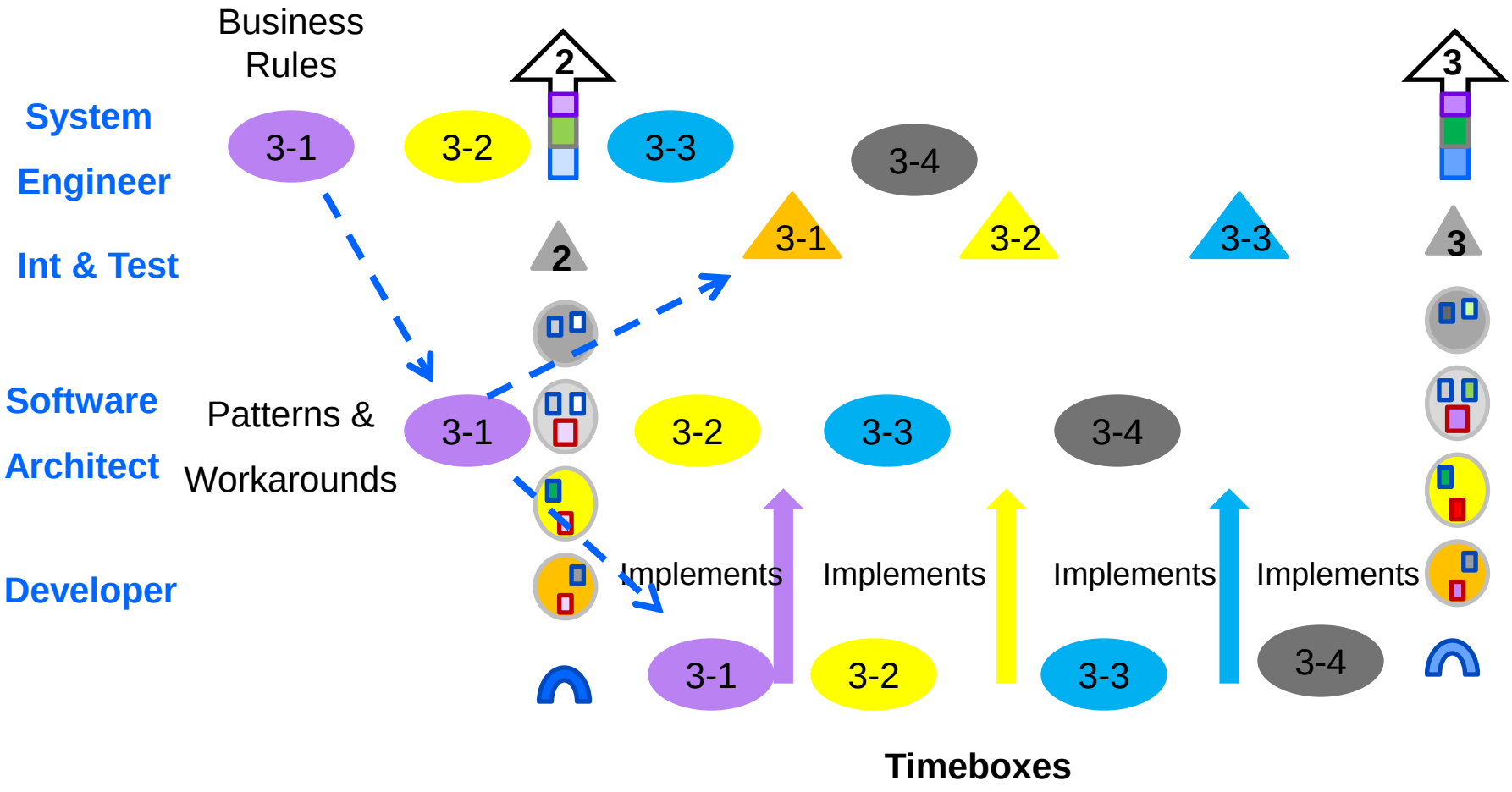
Life-Cycle Plan - SA - Patterns



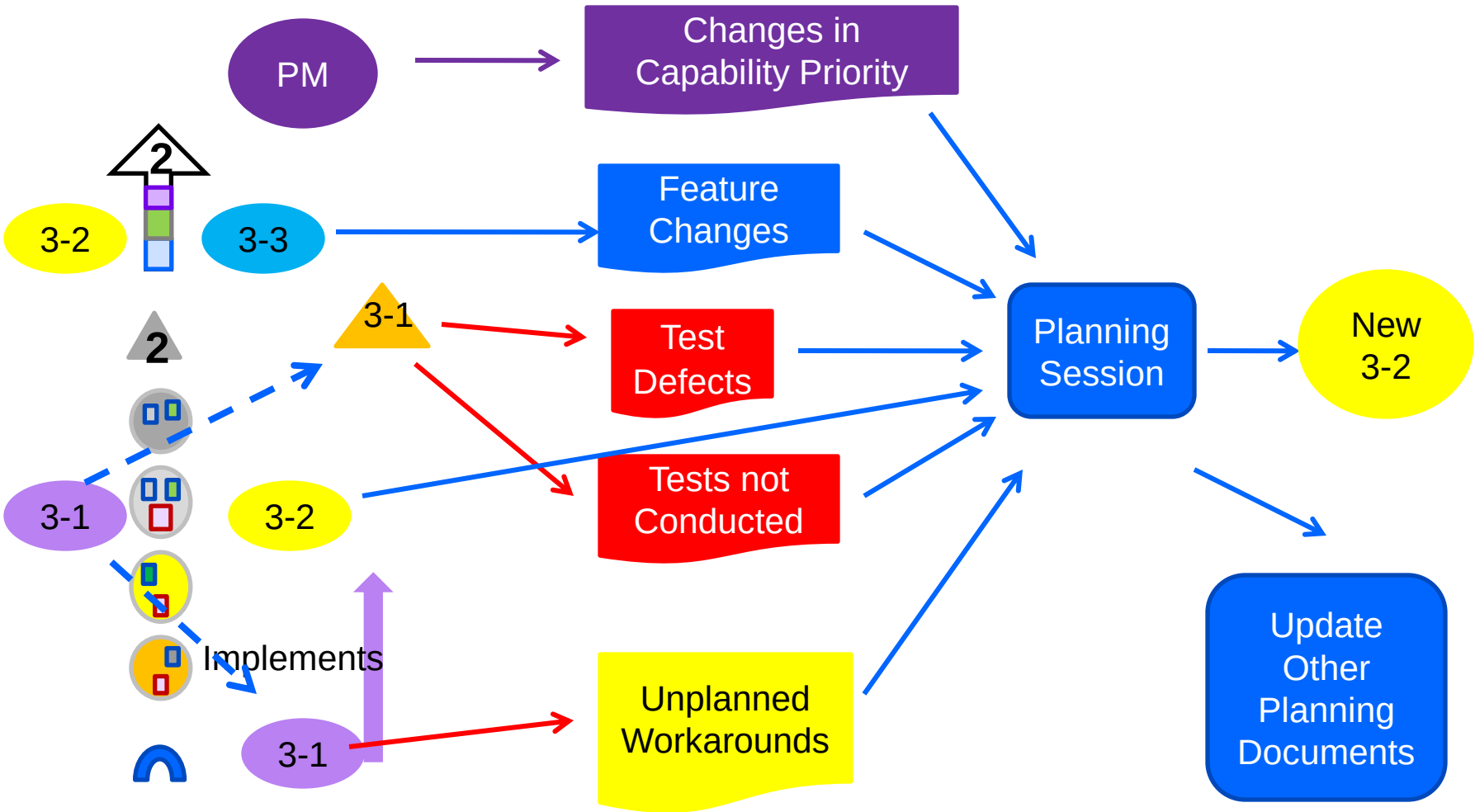
Milestone Plan - Batches Selected



Timebox Activities



Timebox Re-directions - 1



Convincing the PM to Stabilize

Show him examples of success for stabilizing

- and failure of approach1

Understand what went wrong- failure symptoms

Determine Root Cause of failures and mitigation approaches

Develop business scenarios with the PM

- How he would like the process to work

Questionnaire

Organization and process context

- Clashes between the agile process/organization/other processes

Product goals and vision

- Plans and fitness of practices

Product Context

- Architecture vs. coding vs. testing practices, skills and tools

Root Cause Analysis – Typical 1

Symptom

- Scrum teams are spending almost all of their time fixing defects, and new capability development is continuously slipping

Root Cause

- Initial focus was “far future/general” rather than “next delivery cycle/product specific”
  - Plethora of **variation parameters** that interact detrimentally
  - Time pressure to deliver became top priority
  - Delivered an immature product
- There are 3 different cycles
  - Customer Release (yearly, many variants); IV&V Testing (quarterly, 4 variants), and Developmental (monthly, 1 variant)

Solution

Stabilize the Architecture

- Build an architecture for current products
  - Rules, guidelines
  - Over a few timeboxes
- Reduce the # of “variant parameterizations”
- Make everyone play from the same sheet music
- Postpone adding new features

Re-plan the timeboxes

Re-visit the testing strategy/team assignments against variants

Root Cause Analysis – Typical 2

Symptom

- Integration of products built by different scrum teams reveals incompatibility defects causing many failure conditions, leading to significant out-of-cycle rework

Root Cause

- Cross team coordination is poor, even though there are many coordination points and much time spent
- Different interpretations of interfaces by different teams
- Product owner on each scrum team are not seeing the big picture
- Mismatch between Architecture and scrum development

Solution

Stabilize to remove failures

- Postpone adding new features

Identify and “collapse” common services across teams

Use Architectural Runway

- A system that has architectural runway contains *existing or planned infrastructure sufficient to allow incorporation of current and near term anticipated requirements without excessive refactoring*
- Architectural Runway is represented by *Infrastructure* initiatives that have the same level of importance as the larger scale requirements epics that drive the company’s vision forward

Root Cause Analysis – Typical 3

Symptom

- Progress towards meeting milestones is unsatisfactory

Root Cause

- Mapping of capability features to software components per scrum cycle is disorganized
- Some new features are unused in each cycle- wasted effort
- Developer assignment to teams is inflexible

Solution

Build more architectural planning views to align features between teams

Re-organize teams to better fit timebox workloads

Summary

Aligning agile methods with SoS engineering is complex and requires intricate decision making and planning

Re-planning timebox features is necessary

- NOT within timebox, but in-between

Questionnaire revealed many issues, which could be used to indentify root-causes and develop action items to recover program stability.

Contact Information Slide Format

Presenter / Point of Contact
William Wood
RTSS
Telephone: +1 412-268-7723
Email: wgw@sei.cmu.edu

U.S. Mail
Software Engineering Institute
Customer Relations
4500 Fifth Avenue
Pittsburgh, PA 15213-2612
USA

Web
www.sei.cmu.edu
www.sei.cmu.edu/contact.cfm

Customer Relations
Email: info@sei.cmu.edu
Telephone: +1 412-268-5800
SEI Phone: +1 412-268-5800
SEI Fax: +1 412-268-6257

NO WARRANTY

THIS CARNEGIE MELLON UNIVERSITY AND SOFTWARE ENGINEERING INSTITUTE MATERIAL IS FURNISHED ON AN “AS-IS" BASIS. CARNEGIE MELLON UNIVERSITY MAKES NO WARRANTIES OF ANY KIND, EITHER EXPRESSED OR IMPLIED, AS TO ANY MATTER INCLUDING, BUT NOT LIMITED TO, WARRANTY OF FITNESS FOR PURPOSE OR MERCHANTABILITY, EXCLUSIVITY, OR RESULTS OBTAINED FROM USE OF THE MATERIAL. CARNEGIE MELLON UNIVERSITY DOES NOT MAKE ANY WARRANTY OF ANY KIND WITH RESPECT TO FREEDOM FROM PATENT, TRADEMARK, OR COPYRIGHT INFRINGEMENT.

Use of any trademarks in this presentation is not intended in any way to infringe on the rights of the trademark holder.

This Presentation may be reproduced in its entirety, without modification, and freely distributed in written or electronic form without requesting formal permission. Permission is required for any other use. Requests for permission should be directed to the Software Engineering Institute at permission@sei.cmu.edu.

This work was created in the performance of Federal Government Contract Number FA8721-05-C-0003 with Carnegie Mellon University for the operation of the Software Engineering Institute, a federally funded research and development center. The Government of the United States has a royalty-free government-purpose license to use, duplicate, or disclose the work, in whole or in part and in any manner, and to have or permit others to do so, for government purposes pursuant to the copyright license under the clause at 252.227-7013.