Entwicklungsmuster

Für den Umgang mit Änderungen

Dr. Michael Jastram – Jama Software

13. März, 2019 – ReConf

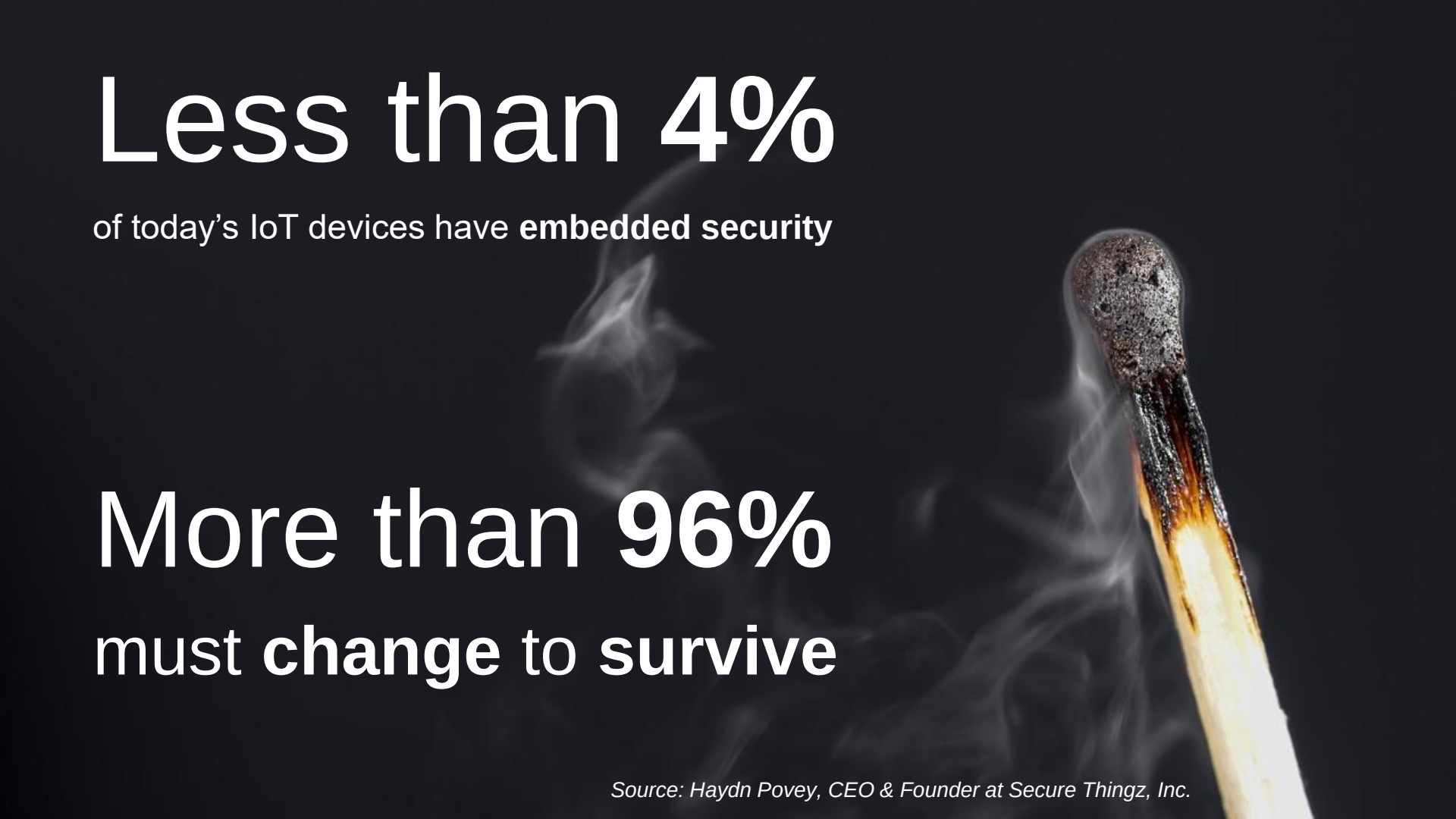
CONFIDENTIAL

Less than **4%**

of today's IoT devices have **embedded security**

More than **96%**

must **change** to **survive**



Source: Haydn Povey, CEO & Founder at Secure Thingz, Inc.

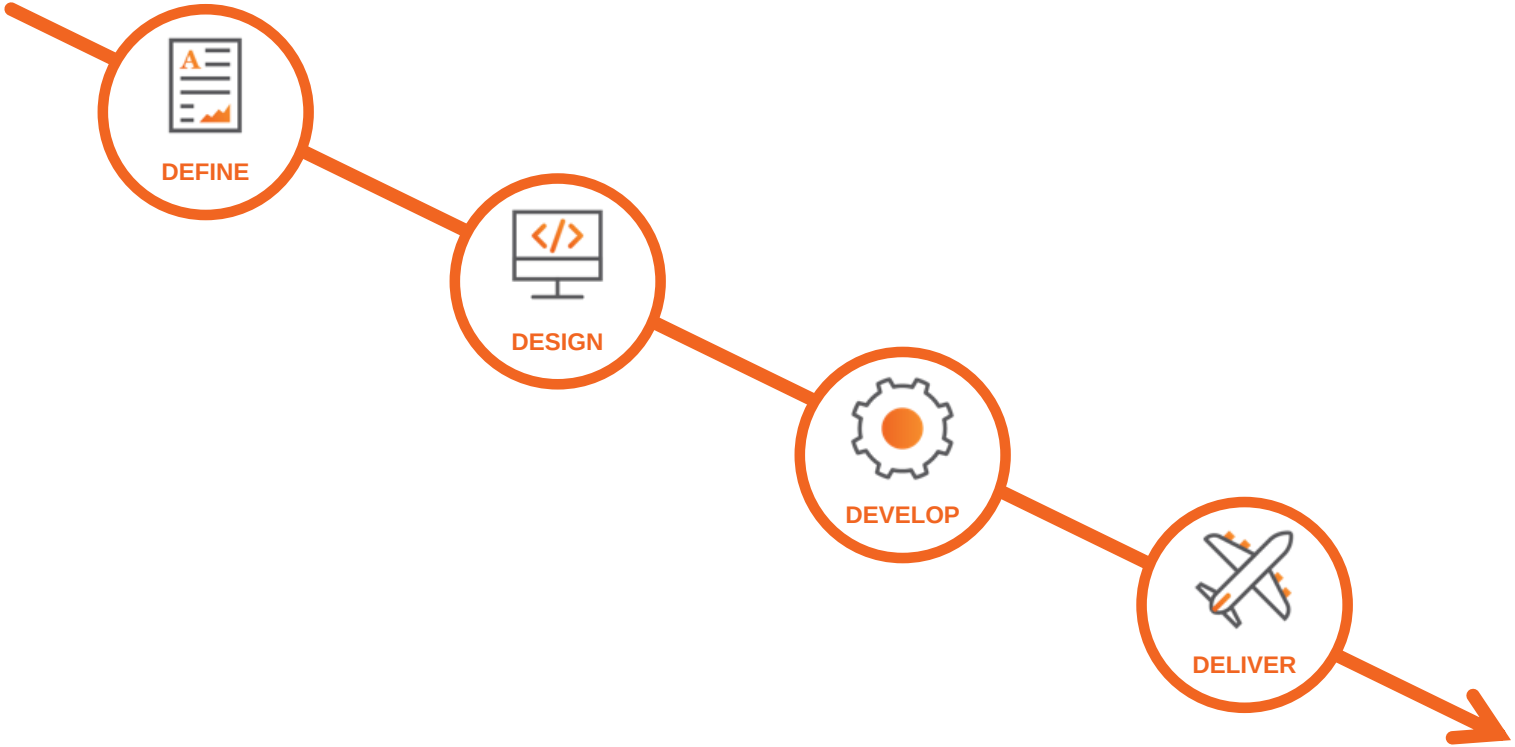
In the last 15 years, 52% of the Fortune 500 companies **have disappeared.**

Average life expectancy of a company:

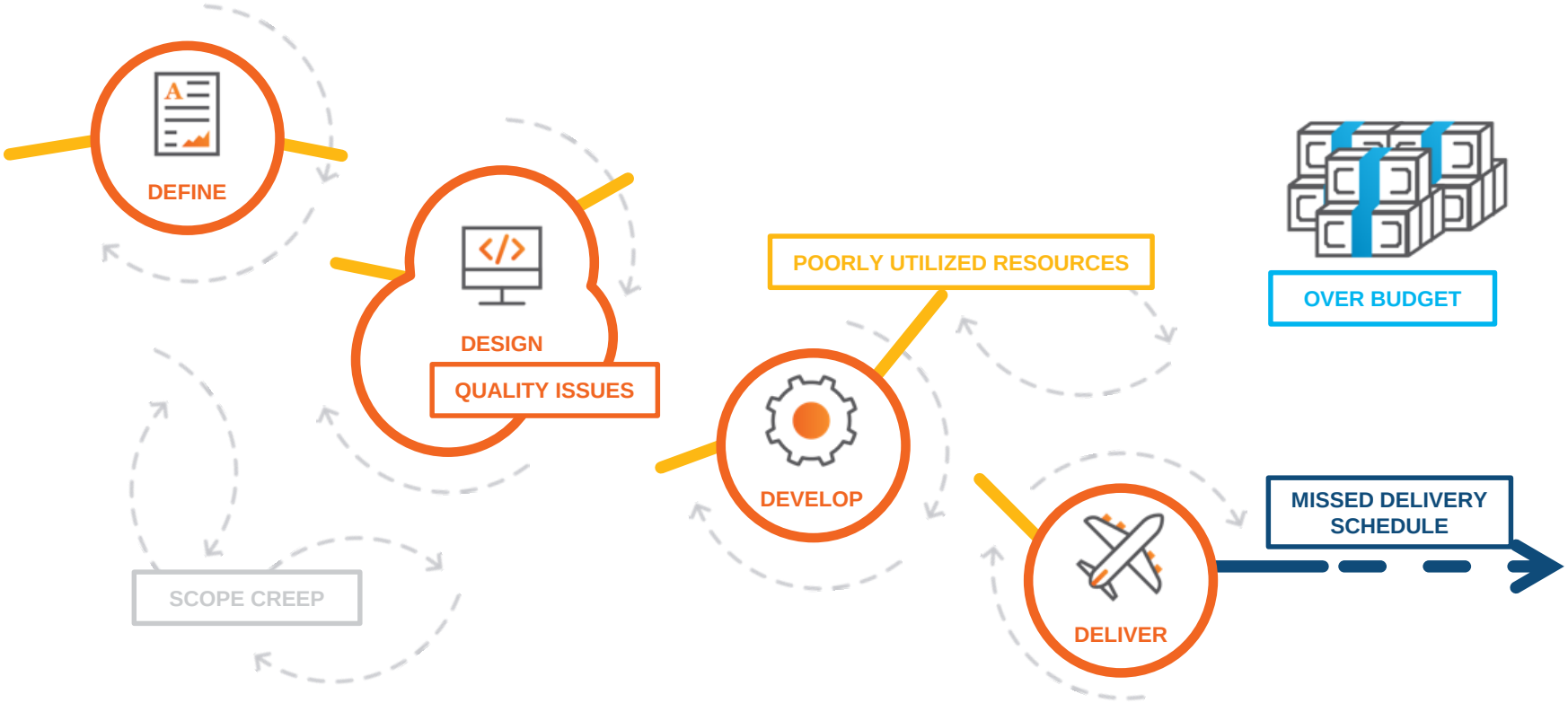
75 Years in 1955 **vs.** **15 Years** in 2015

Source: Fortune 500

“Waterfall” Product Development



The Challenge



Is “Agile” the Solution?

Individuals and interactions over processes and tools

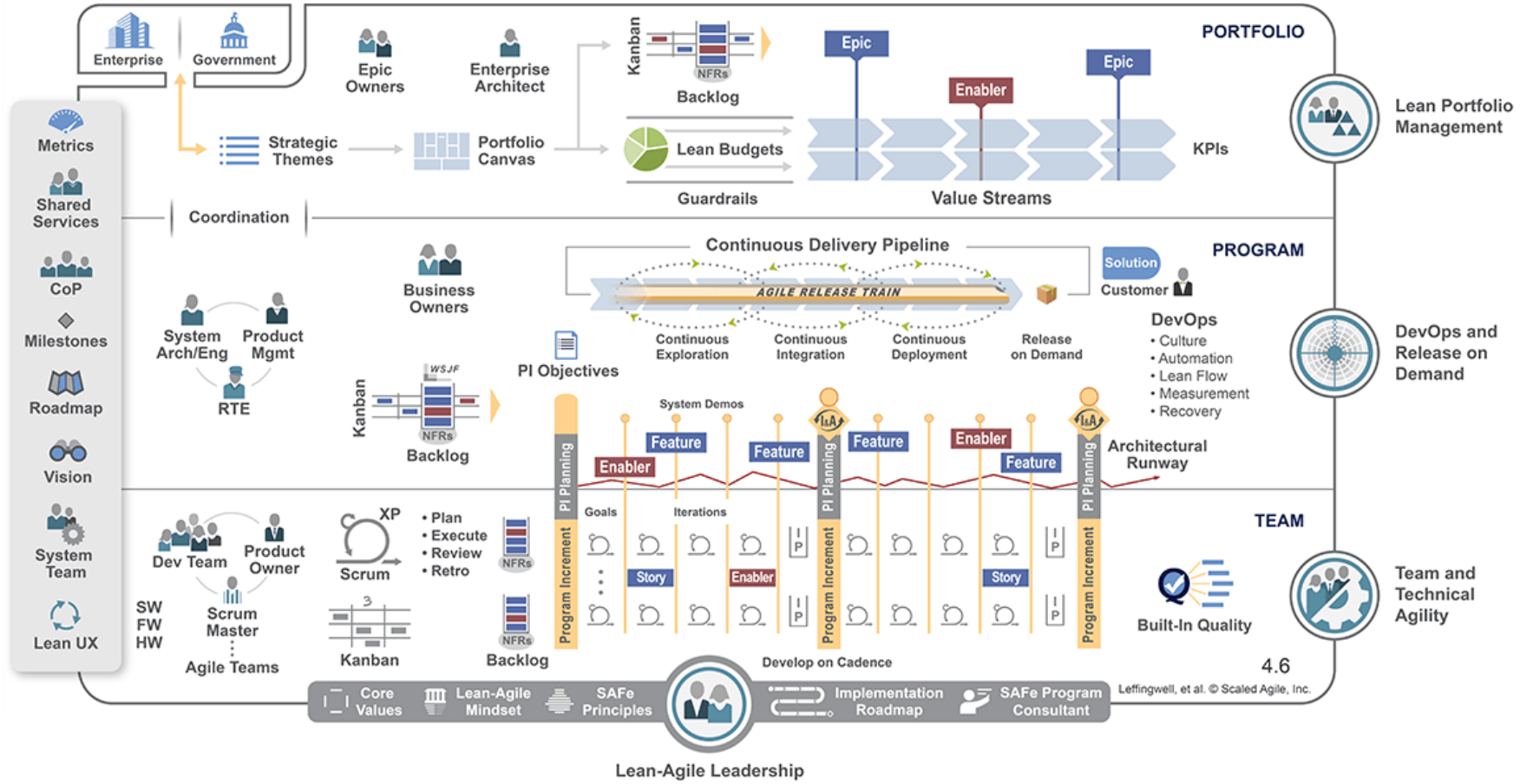
Working software over comprehensive documentation

Customer collaboration over contract negotiation

Responding to change over following a plan

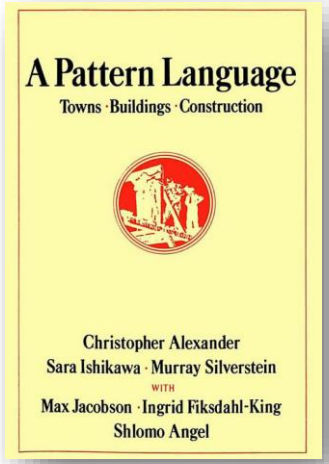
Yes, but…

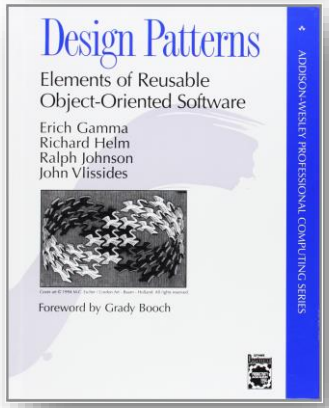
Scaled Agile (SAFe)



What are Patterns?

- Initially developed by architect Christopher Alexander as a ”language to describe common architectural problems and their solutions”
- Adapted in Software Engineering (Design Patterns) to capture “Elements of Reusable Object-Oriented Software”
- Ever since, pattern languages have been developed for many fields
- Great way for capturing best practices without having to worry about their interplay initially





Scope of Patterns for Mastering Change

People

Processes / Methods

Tools

People

- Source for finding pain points
- No action without “buy-in”
- Key roles: Owner and Sponsor

Processes and Methods

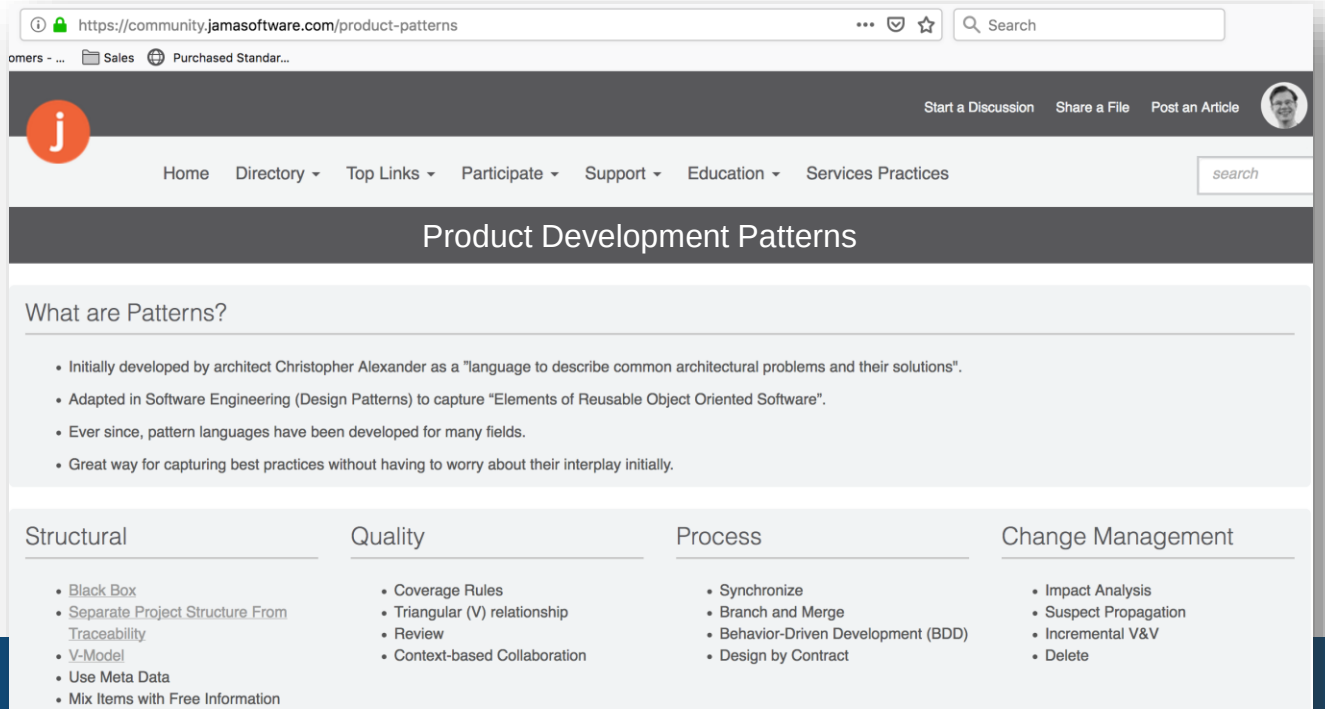
- Always preexisting (if not, pick a framework)
- Should be tweaked, not drastically changed (if possible)
- Require regular alignment with people and tools

Tools

- Product development is impossible without tools
- Most tools are “good enough”
- Alignment can drastically improve things

Patterns Online

https://community.jamasoftware.com/product-patterns



Step 1: Identify Pain

Example: Smart Thermostat

Problem: Coordination between System and Subsystem teams cumbersome

Step 2: Measure

What to measure

↓ requirements-related defects

↓ requirements review and approval cycle times

↓ design cycle times

↓ engineering ‘rework’

↓ % engineering effort devoted to rework/remediation

↑ % engineering effort devoted to new features

↓ FTE time spent in unproductive meetings

↓ time eliciting and documenting/decomposing requirements for traceability

↓ QA cycle times with reduced level of effort

↓ customer/user-identified bugs (complaints/tickets)

↑ avg customer sat/NPS response

↑ features per release

↓ missed deadlines/release dates

↓ late-stage defects (post-development)

↓ late-state changes/churn (post-development)

↓ time developing and documenting risk management analysis

↓ in audit prep time (internal/external)

↓ in audit findings (ITSec)

↑ test coverage earlier in lifecycle

↑ early-stage changes/churn (pre-development)

↓ # escaped bugs

↑ team confidence per release (on-time/on-spec/on-budget)

↑ customer-centric feature requests added to backlog & delivered

↓ # patch releases or out of cycle bug fixes

Step 3: Identify Pattern

Example: Smart Thermostat

Problem: Coordination between System and Subsystem teams cumbersome

	Structural	Quality	Process	Change Mgmt
Black Box	x			X
Define Workflows		x	x	x
Separate Project Structure From Traceability	x		x	
V-Model	x	X	x	X
Define and Enforce a Data Model		x	x	
Interfaces	X	X		X
Behavior/Test Driven Development (BDD)			x	x
Consistency Criteria		x	x	
Branch and Merge			x	x
Evolve Hardware and Software Together	x		x	
Functional Descriptions	x	X		
Good Requirements		x	x	
Review		x	x	
Don't Repeat Yourself (DRY)	x	x		
One-Pager			X	X
Use Meta Data		x	x	
Engineering Efficiency Empowerment			X	X
Define Hand-Over Points		X	X	

Example: Smart Thermostat

Problem: Coordination between System and Subsystem teams cumbersome

Pattern: Interfaces ★

By Michael posted 20 hours ago Edit 0 Like

Intent

Be explicit about identifying and documenting interfaces

Motivation

As the complexity of products grows, we need robust decomposition (see Black Box). This in turn requires stability in the interfaces. That's why interfaces need special attention. Having the right interfaces in the right places can make a huge difference in being able to react to changes, robustness and flexibility.

Interfaces exist on many levels: It can be about connecting two physical components. It can be about consuming software libraries. It can be about a combination of hardware and software (e.g. a USB connection). It can be about talking to machines or humans. The key here is to recognize those interfaces that matter with respect to robustness and flexibility, and to document those properly.
Interfaces also go hand in hand with architecture, which provides a framework for placing the interfaces.

Applicability

Use the pattern when:

- You define your architecture and align it with your product description
- You design a product line, or modular product
- You intend on consuming third-party components, especially if they will be exchanged at run time

Structure

We distinguish between physical, signal and software interfaces. Physical interfaces are well understood and typically unproblematic (liquids, gases, forces, electricity, etc,). Signals are typically carried by an electric connection and follow a protocol. Last, software interfaces can be completely independent from physical interfaces and concern the interplay of software.

Implementation

Depending on the level of formality that you need, you can simply create an informal component description where you list all the interfaces. On the other end of the spectrum, you use a dedicated modeling tool (Enterprise Architect, Cameo, etc.).

Related Patterns

- Black Box
- Evolve Hardware and Software Together
- One-Pager
- Define Hand-Over Points

0 comments 4 views

One-Pager

Step 4: Implement Pattern

Example: Smart Thermostat

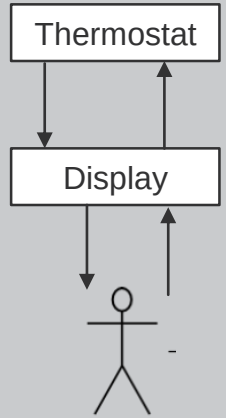
Solution: One-Pager

Thermostat Display

Interfaces

Functions

Notes



Step 5: Measure Outcome

Case Studies using One-Pagers

Customer in the Marine Industry

- Struggling to introduce SE
- Subsystem Departments are “Islands”
- SysML for Architecture not accepted
- One-Pagers drastically improved alignment

Wikispeed

- Builds cars on a weekly schedule
- Independent teams build the components
- Assembly in a weekly 1-hour session
- One-Pager aligns teams
- https://se-trends.de/extreme-manufacturing/

Disecting a Pattern

Intend

Also Known As

Motivation

Applicability

Structure

Consequences

Implementation

Related Pattern

Disecting a Pattern: **Branch and Merge**

Intend

Avoid Copy & Page

Disecting a Pattern: **Branch and Merge**

Also Known As

- Reuse
- Synchronization

Disecting a Pattern: **Branch and Merge**

Motivation

Copy and Paste is very easy, but in the long run, we regret it. Branching (and optionally merging) is an alternative that allows you to drastically reduce rework, thereby saving time and improving quality.

Branching and merging has been state of the art in software development for 20 years, and it's time that we apply it to product development as well.

Disecting a Pattern: **Branch and Merge**

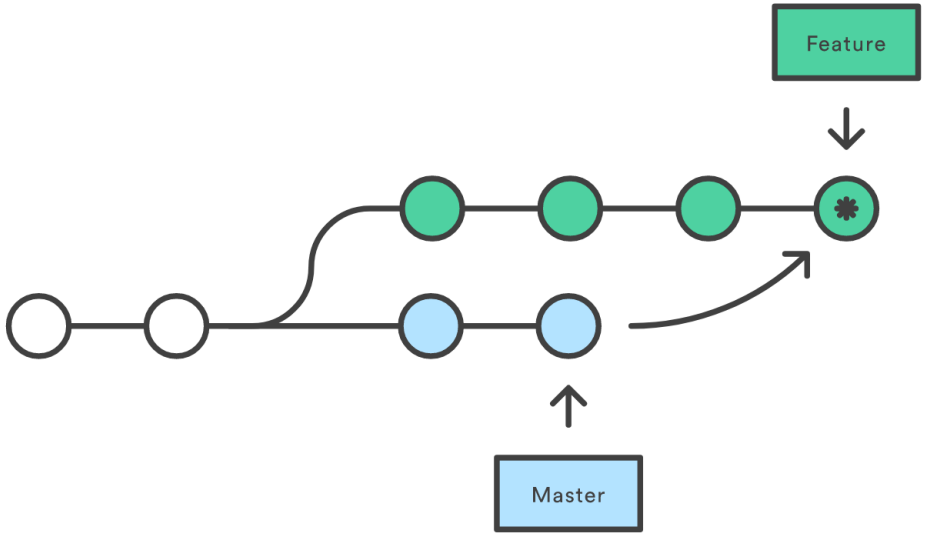
Applicability

Use the pattern:

- Reuse of standard content
- Asynchronous, independent work
- Family of similar content

Disecting a Pattern: **Branch and Merge**

Structure



Branching Image source: Jens Lechtenbörger, published under the Creative Commons license CC BY-SA 4.0.

Disecting a Pattern: **Branch and Merge**

Consequences

Benefits

- Save work in parallel
- Keep trace from branch to trunk
- Quality & Speed

Liabilities

- Messy without process
- Merge can revert changes
- Can be overkill

Disecting a Pattern: **Branch and Merge**

Implementation

Branching and Merging requires proper tool support, and on the right level of granularity. For instance, you can use MS Word for branching and merging on the document level, but it quickly becomes cumbersome with the size of the document.
Jama Connect provides reuse capabilities that are powerful enough for dozens to hundreds of branches.
Beyond that, and with additional needs like parametrization, you can use a dedicated tool like Pure Variants.

Disecting a Pattern: **Branch and Merge**

Related Patterns

- DRY
- Evolve Hardware and Software Together
- Define Hand-Over Points
- Define Workflows

Where We Are Today

	Structural	Quality	Process	Change Mgmt
Black Box	x			x
Define Workflows		x	x	x
Separate Project Structure From Traceability	x		x	
V-Model	x	x	x	x
Define and Enforce a Data Model		x	x	
Interfaces	x	x		x
Behavior/Test Driven Development (BDD)			x	x
Consistency Criteria		x	x	
Branch and Merge			x	x
Evolve Hardware and Software Together	x		x	
Functional Descriptions	x	x		
Good Requirements		x	x	
Review		x	x	
Don't Repeat Yourself (DRY)	x	x		
One-Pager			x	x
Use Meta Data		x	x	
Engineering Efficiency Empowerment			x	x
Define Hand-Over Points		x	x	

Bonus Pattern: **Engineering Efficiency Empowerment**

Motivation

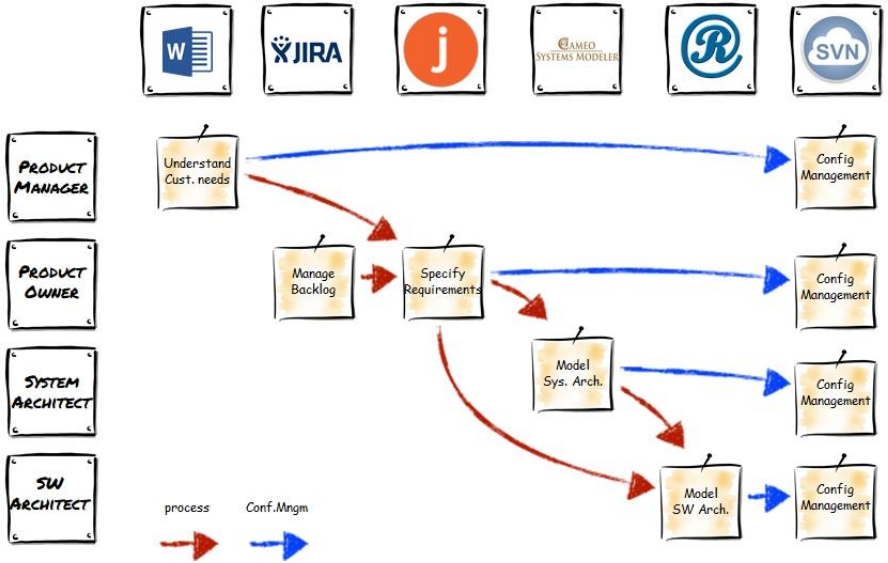
Systematically remove bottlenecks between tools

Implementation

https://eee-check.de/

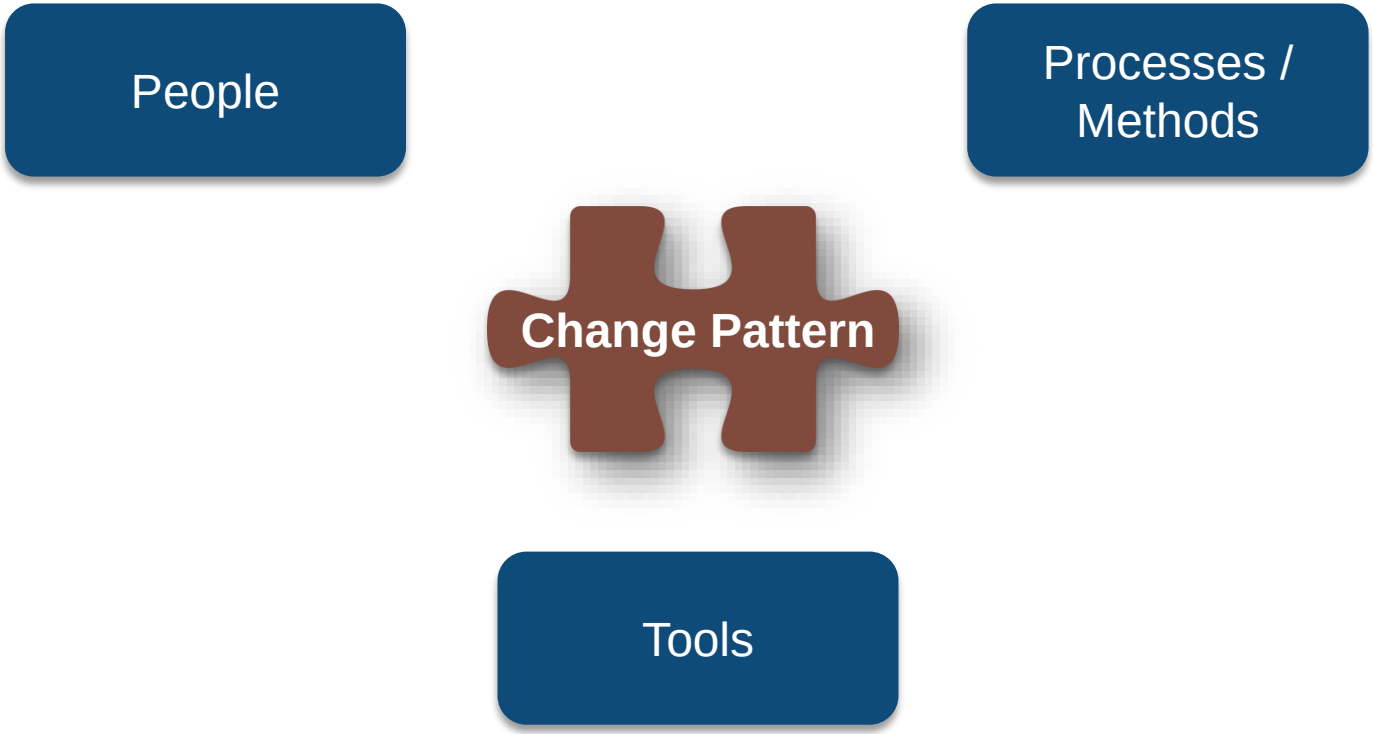
Authors

Tim Weilkiens & Andreas Willert



Patterns Online

https://community.jamasoftware.com/product-patterns



Questions…

Dr. Michael Jastram
Senior Solutions Architect
Jama Software

Blogs
jamasoftware.com/blog
formalmind.com/blog
se-trends.de

… and Thank You!