



SAEC Days 20

Big Data for Safety: Unleashing the Power of Your Software Project Data

Reaching Maturity with Continuous Quality Process

Agenda

- ▶ **Context:** Quality layers

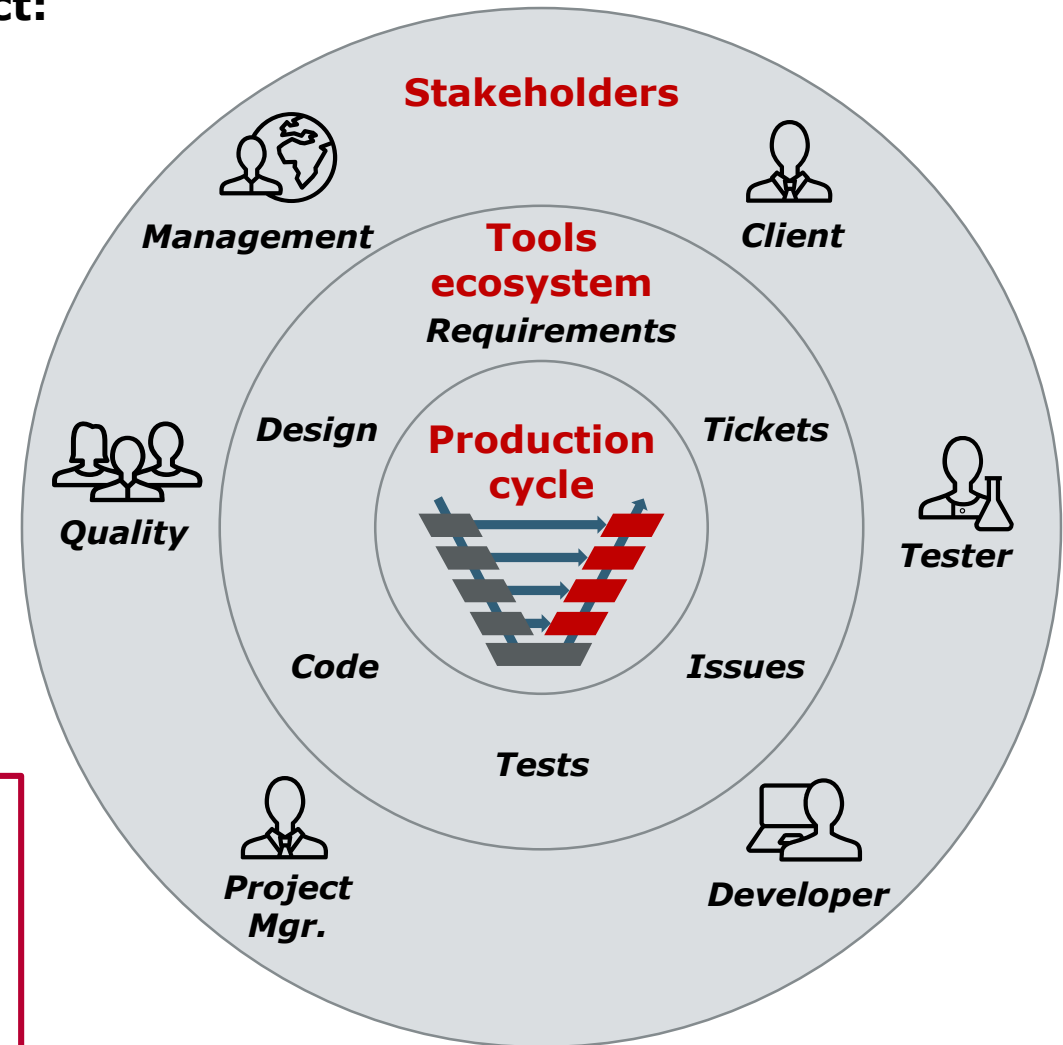
- ▶ **Concept:** Continuous Quality

- ▶ **Proposition:**
 - ▶ Quality Indicators and Evaluation
 - ▶ Quality Gates
 - ▶ Quality in Continuous Integration

Handling multiple Quality layers

Quality in an industrial systems / software project:

- ▶ Is expected by **several Stakeholders** with specific needs
- ▶ Depends on **multiple tools** with varying versions, data models and formats
- ▶ Uses large data sets during the **production cycle** continuously generated



Reaching quality is complex and data-intensive

Handling quality layers must be:

- Efficient
- Cost-effective

Continuous Quality



"Quality is not an act, it is a habit."

(inspired by Aristotle)

Quality must continuously be evaluated and monitored

**To meet
Quality objectives**

Agree on quality evaluation

Break down quality into scopes

Monitor quality drifting

**To comply
with Standards**

CMMI ASPICE

ISO 9001 ISO 25010

Corporate DO-178

**To improve
Maturity**

At Product level

At Project level

At Process level

Why and how to address quality?

▶ **Quality benefits**

- ▶ Corporate Image
- ▶ Customer Satisfaction
- ▶ Business Revenue

▶ **Quality effort strategies**

▶ **No effort**

- > High risk of quality drop
- > Negative impact on business benefits

▶ **One shot effort** (audit, certification)

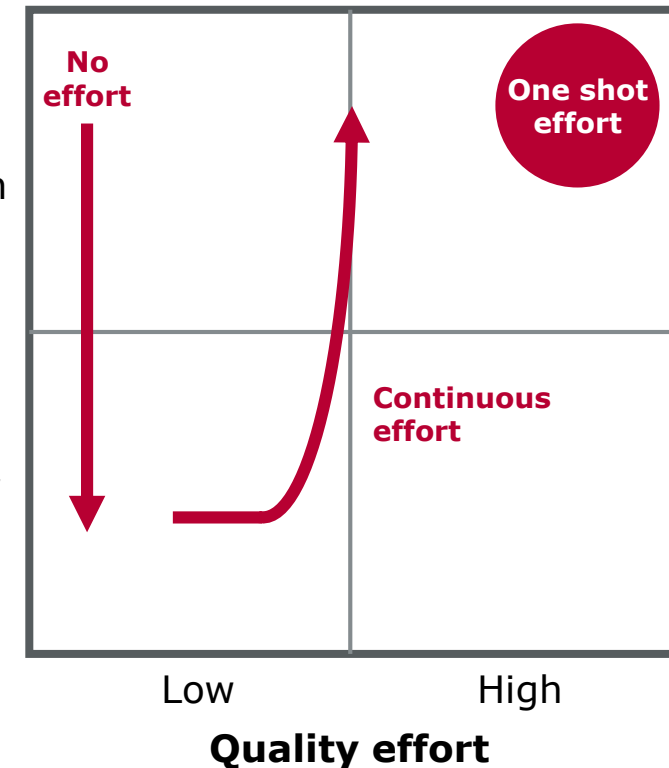
- > Costly
- > Temporary marketing & business boost

▶ **Continuous effort**

- > Initial cost of new M&T, of stakeholders
- > Distributed among actors and processes
- > Lasting positive impacts

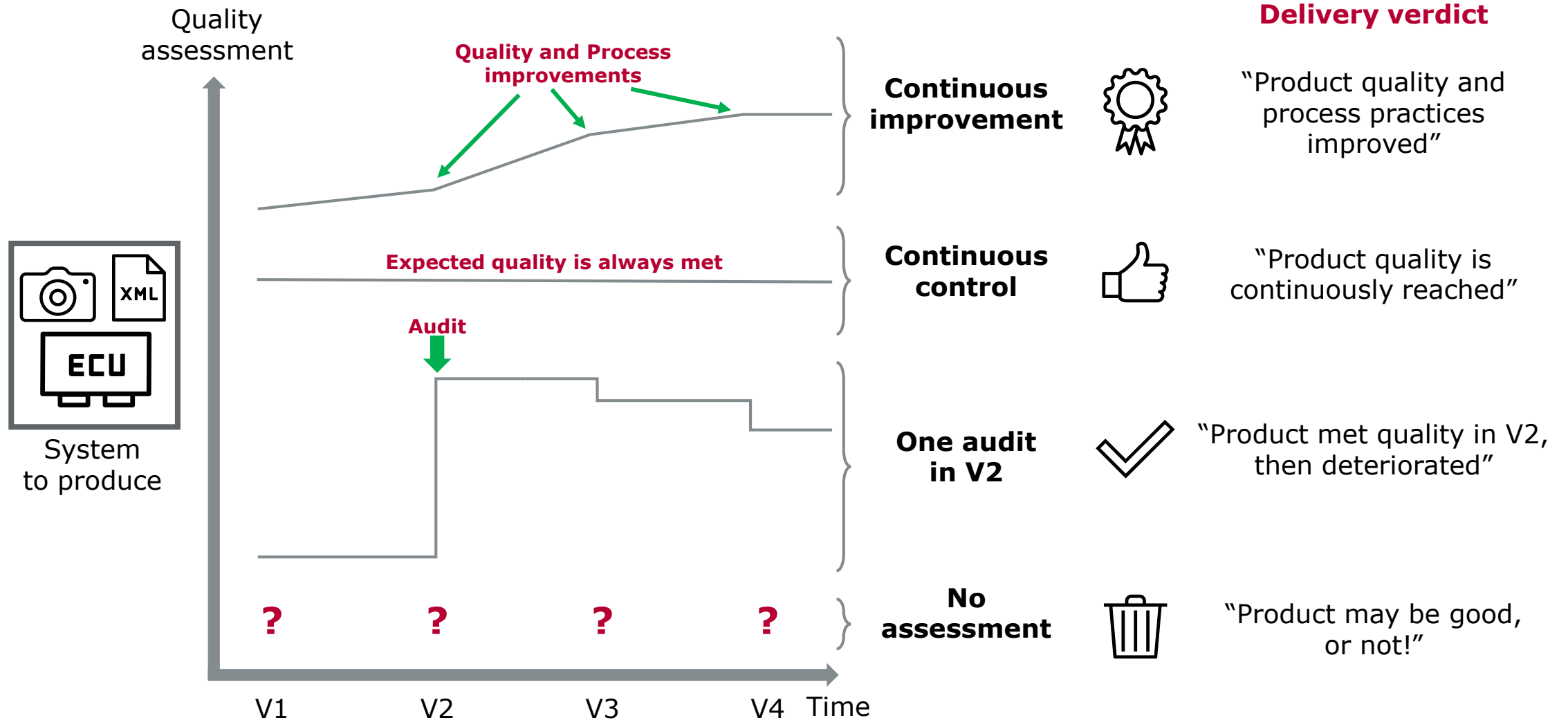
Image
Satisfaction
Revenue

Quality
benefits



Continuous Quality has long lasting benefits

Benefits of continuous quality



Quality Indicators

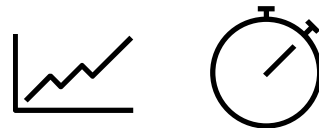
▶ **Achieve product quality**

- ▶ Check Design
- ▶ Track Requirements
- ▶ Monitor Tests
- ▶ Analyze Code



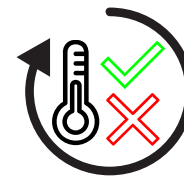
▶ **Achieve project quality**

- ▶ Manage costs
- ▶ Anticipate delays
- ▶ Follow objectives and milestones



▶ **Achieve process quality**

- ▶ Enforce best practices
- ▶ Optimize methods
- ▶ Verify tools usage



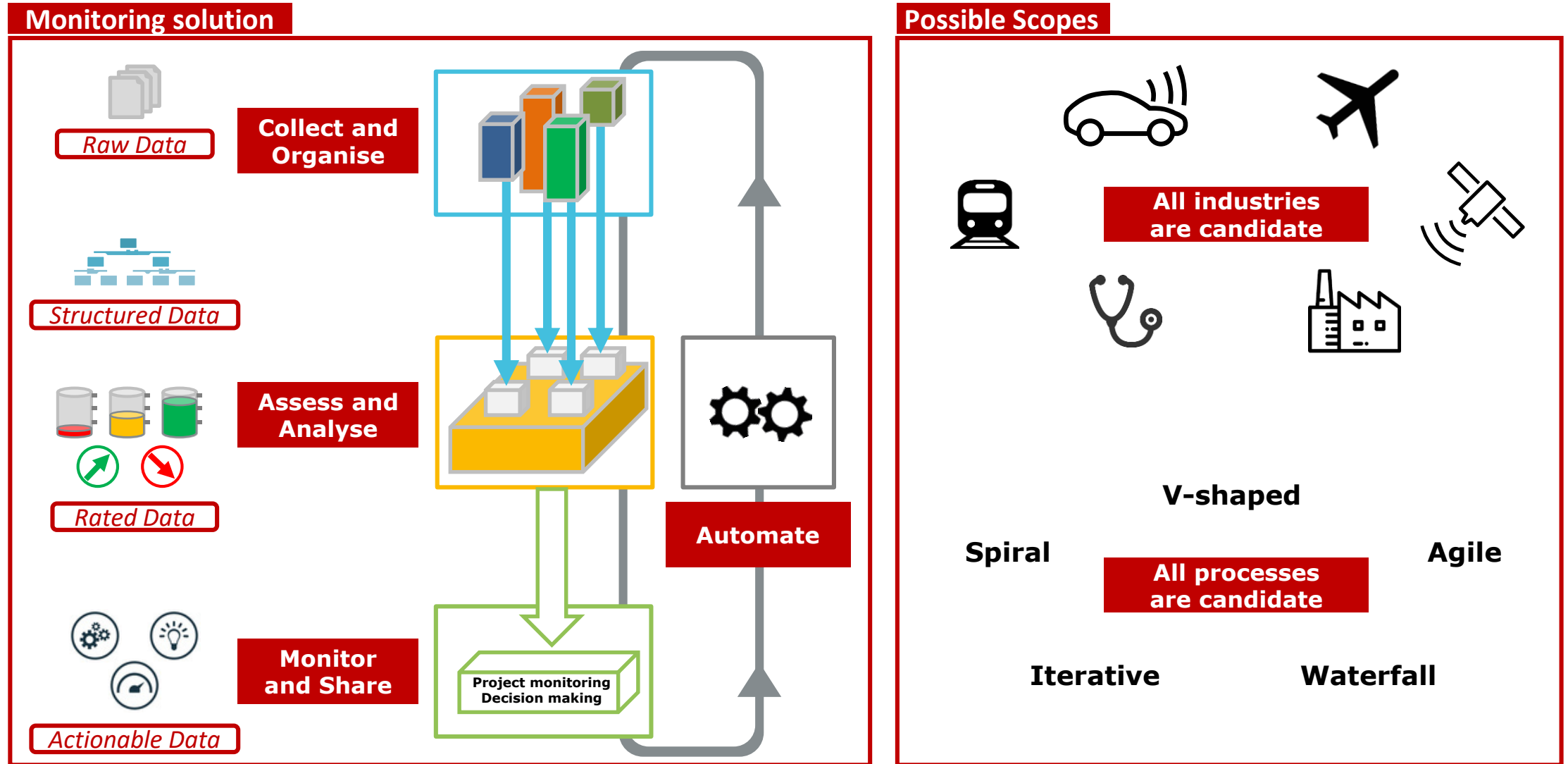
By using indicators

- ▶ Shared by all stakeholders
- ▶ Computed by a deterministic method
- ▶ Continuously generated by automation

**Software production is a natural fit
for indicators automation**

Indicators are essential to quality monitoring

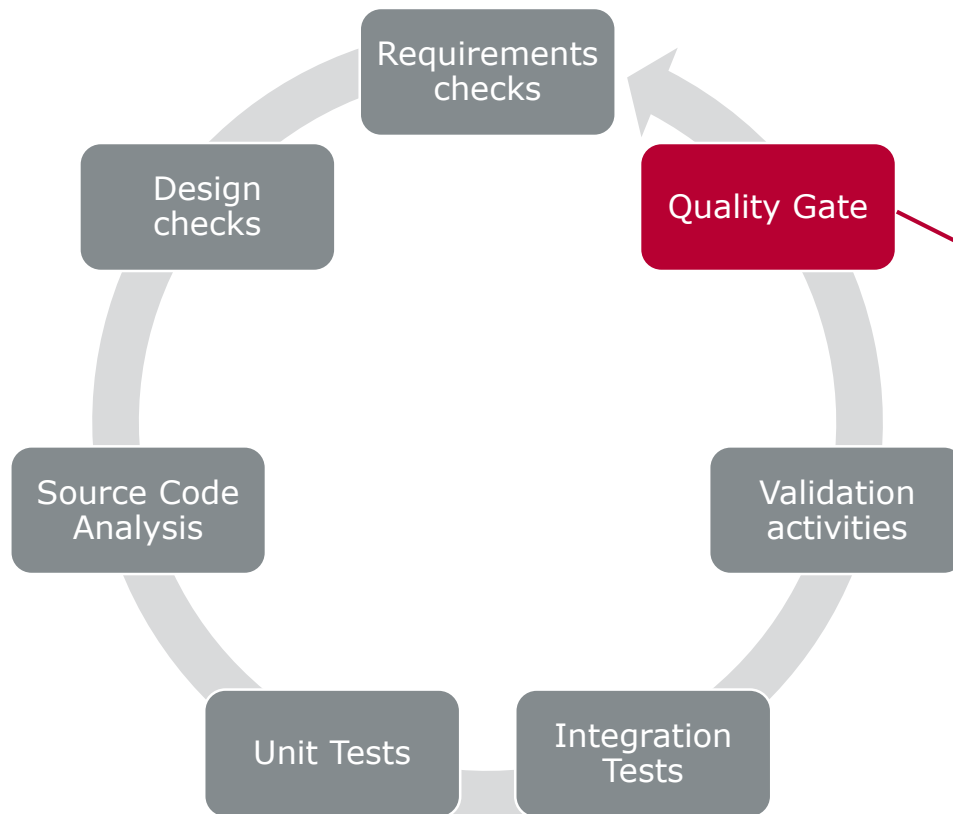
Quality Evaluation overview



Quality Gates



Classic DevOps phases



A Quality Gate is a checkpoint in the cycle

It should be:

- ▶ **Flexible**
Some projects don't need all phases
Checkpoints vary with the project lifecycle
- ▶ **Replicable**
Massive deployments need Quality Gate templates
- ▶ **Automated**
Business Intelligence analytics

Quality Gates examples



Objective

Checkpoint

Details

Check
basic Coverage objective

"Global Code Coverage must be >50%"

Coverage is aggregated at project level

Encourage
good practices

"Don't accept any violations on new source code"

Based on "New Technical Debt" computation

Inspect
risk-based Coverage compliance

"Expected coverage has to depend on SIL breakdown"

Level	Stmt.	Branch	MC/DC
Level A	80%	50%	0%
Level B	100%	80%	50%
Level C	100%	100%	80%
Level D	100%	100%	100%

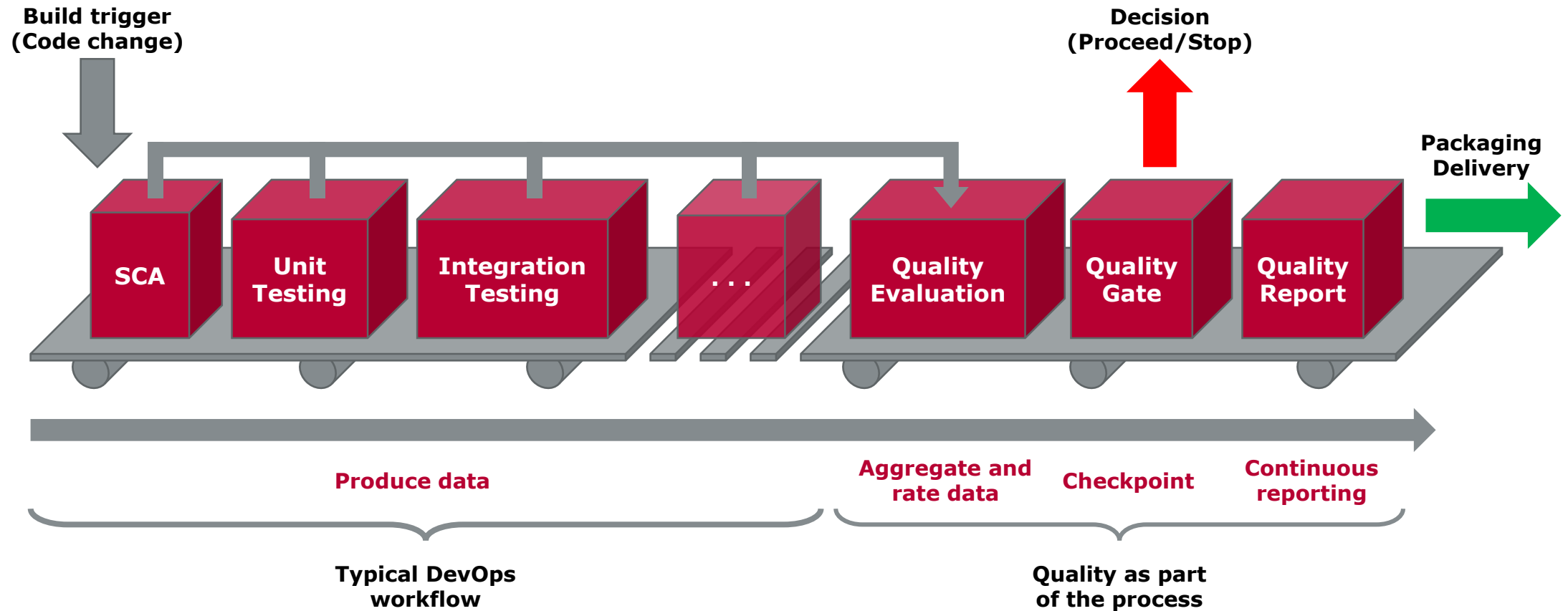
Monitor
development processes

"Requirement/Code/Tests stability must comply with project lifecycle"

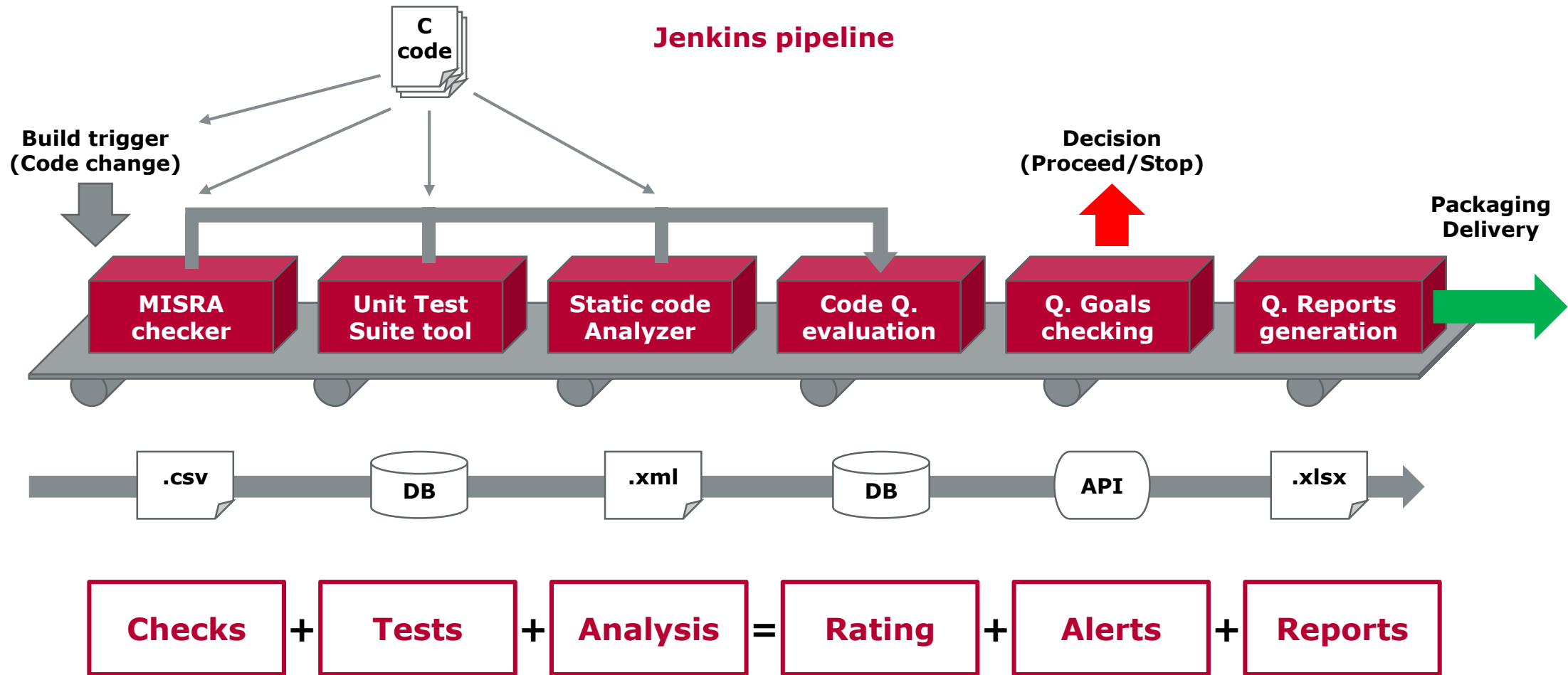
Uses trend and traceability analysis

Quality in Continuous Integration

Continuous Integration pipeline (Jenkins, GitLab)



Quality in Continuous Integration example



Continuous Quality with Squire

Project versions
Created by Continuous Integration

Project breakdown
Fed by tools ecosystem

Quality indicators
Grouped by categories

The screenshot displays the Squire application interface. At the top, there's a navigation bar with 'Squire' and various menu items like 'Explorer', 'Projects', 'Favourites', etc. The main content area is divided into several sections:

- Project Portfolios:** A tree view on the left showing project versions (V1 to V7) and artifacts like 'machine.c' and 'player.c'.
- Key Performance Indicator:** A central section showing a 'Higher Quality' bar chart with levels A through G, and a 'Lower Quality' section with metrics like 'Technical Debt' and 'Complexity Volume'.
- Quality rating:** A callout box pointing to the KPI section, stating 'Aggregating Indicators, Trends, Objectives'.
- Monitoring features:** A callout box pointing to the top right, listing 'Qual. Gates, Reports, Action plans'.
- ISO25010 Quality Breakdown:** A radar chart showing metrics like Efficiency, Security, Portability, and Reliability.
- Code Cloning Trends:** A line graph showing trends from V1 to V6.
- Function Complexity Map:** A heatmap showing complexity levels for various functions like 'send_score(int)' and 'machine_plays()'.
- Coding Rule Violations:** A bar chart at the bottom showing the number of violations across versions V1 to V6.

Copyright © 2020 Vector Informatik GmbH - All rights reserved - https://www.vector.com/squire

Conclusion

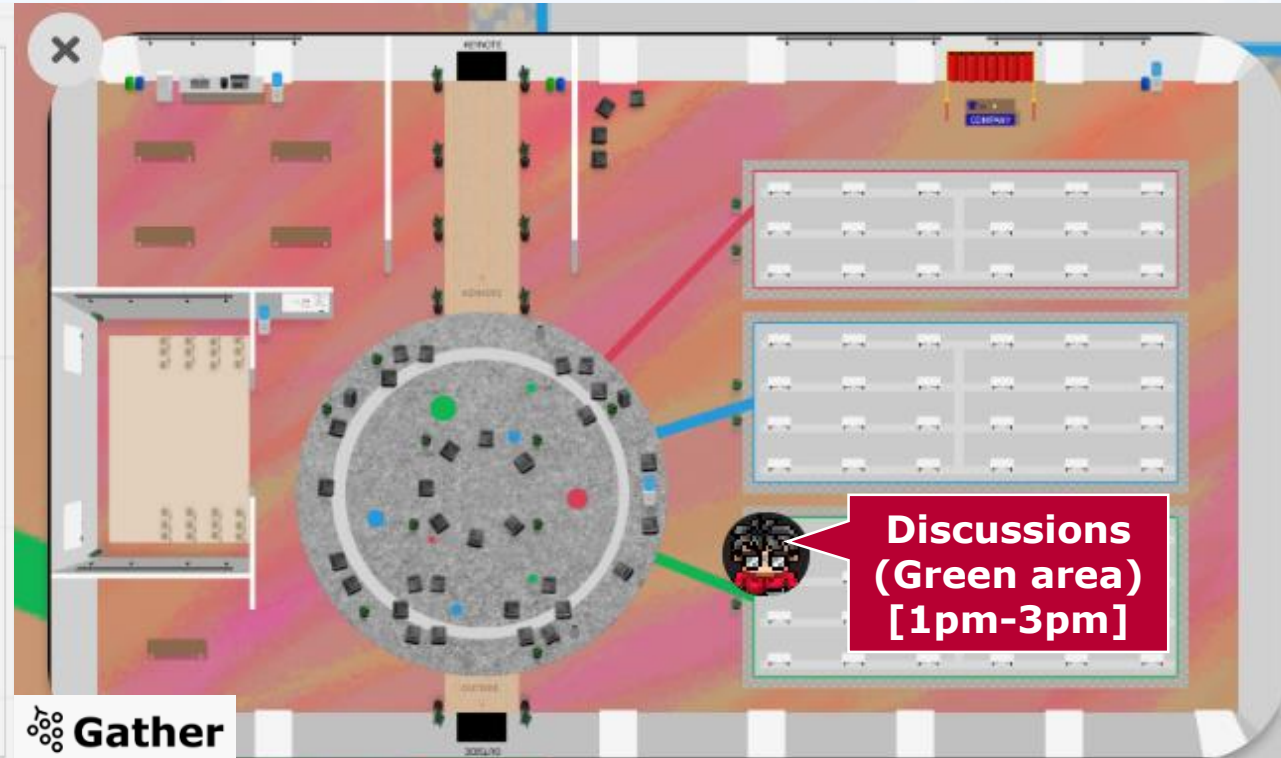
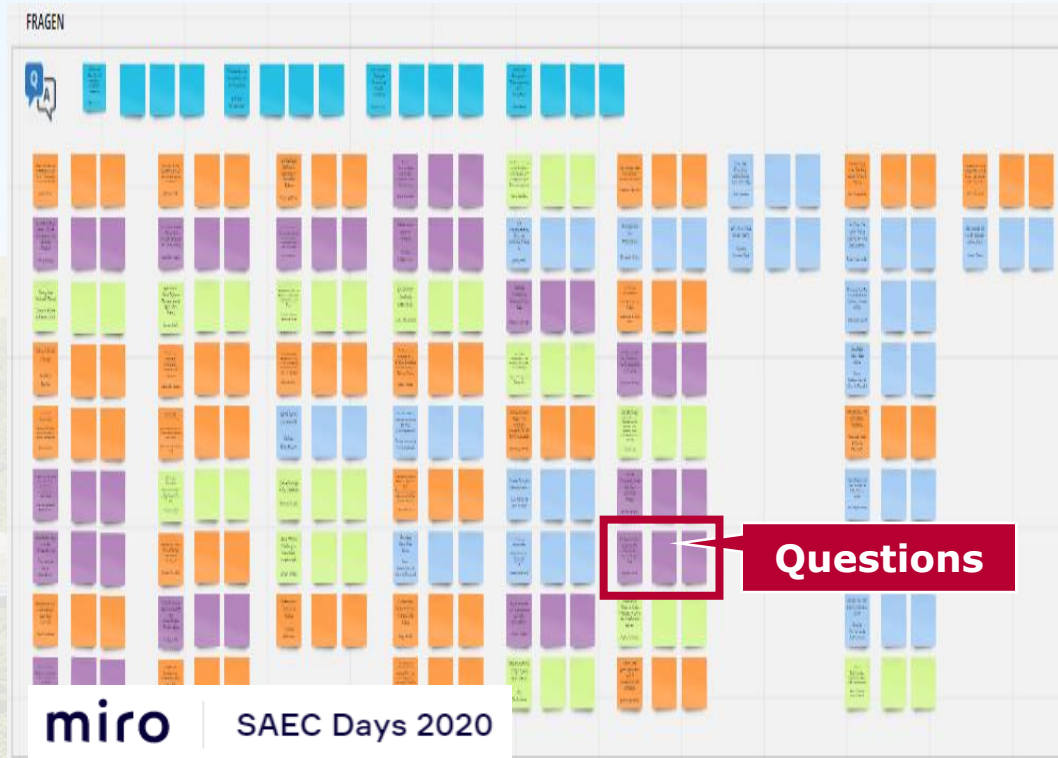


Continuous Quality Benefits

- ▶ **Daily benefits**
 - ▶ Distributed work
 - ▶ Shared responsibility
 - ▶ Continuously monitored quality

- ▶ **Long term benefits**
 - ▶ Smoothed initial cost
 - ▶ In-process quality
 - ▶ Improved maturity

Thank you for your attention. Questions are welcome!



More info on Vector and Squore: www.vector.com/squore 

Author: Flavien Huynh (flavien.huynh@vector.com) 

Vector Toulouse