ERICSSON

BLEKINGE TEKNISKA HÖGSKOLA · BTH ·

SERL Sweden
LEADING SOFTWARE ENGINEERING

Utilizing CI Environment for Evaluation of Software Quality Attributes

- Liang Yu

CI: Continuous Integration

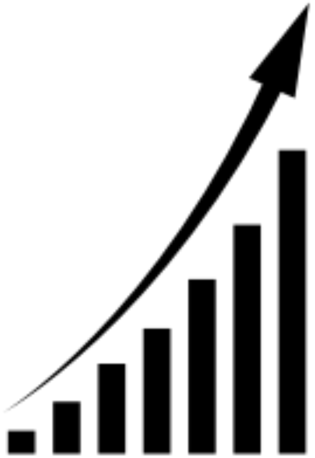
Context

- Quality attributes of the System Under Test (SUT[1])
  - Evaluation focused on non-functional aspects
  - Metrics
- The use of CI
  - CI environment data

[1] Laranjeiro, N., Agnelo, J., & Bernardino, J. (2021). A systematic review on software robustness assessment. *ACM Computing Surveys (CSUR)*, *54*(4), 1-65.

Objectives

- More capacity for assessing quality attributes
- Shorter feedback loop for the evaluation
- Continuous quality improvements



Problems

- How to know the capabilities of CI environments for quality attribute evaluation

- What metrics exist to support the evaluation of quality attributes in automated manners

- A CI solution may not be suitable for everyone

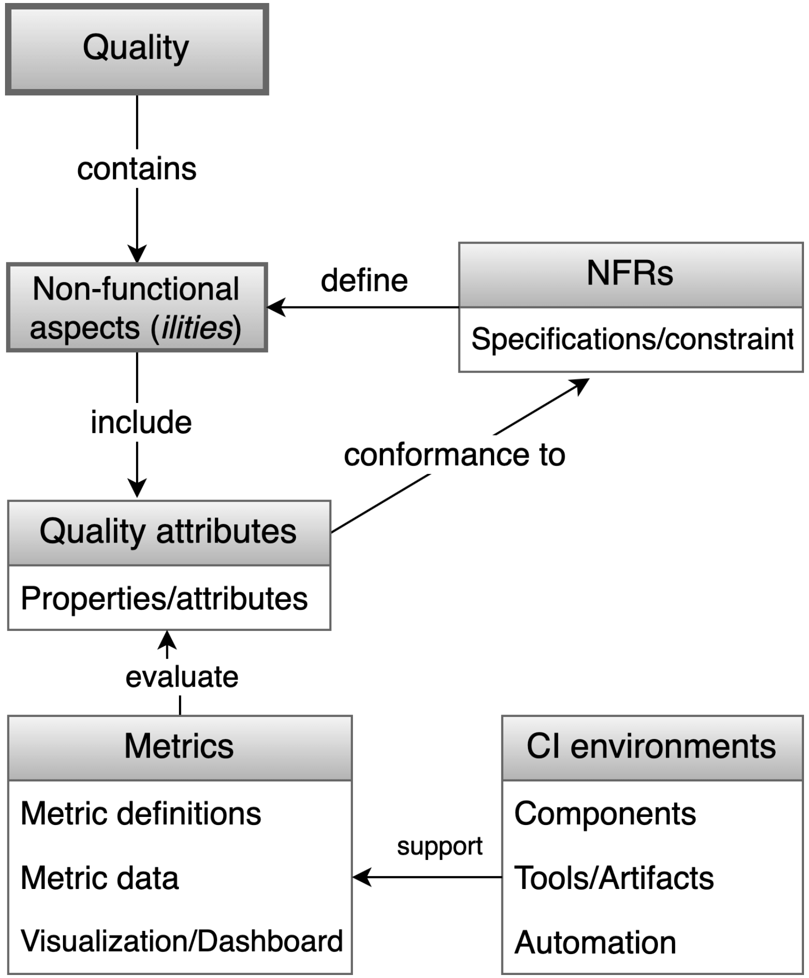
What we do

- What is the state-of-the-art of quality attribute evaluation using CI?
  - A CI base-line model[1]
- How to utilize CI environments for quality attribute evaluation?
  - Empirical evidence[2] - metrics for the evaluation
- How to maximize the use of CI environments?
  - Evolve CI to fit the need of quality attribute evaluation

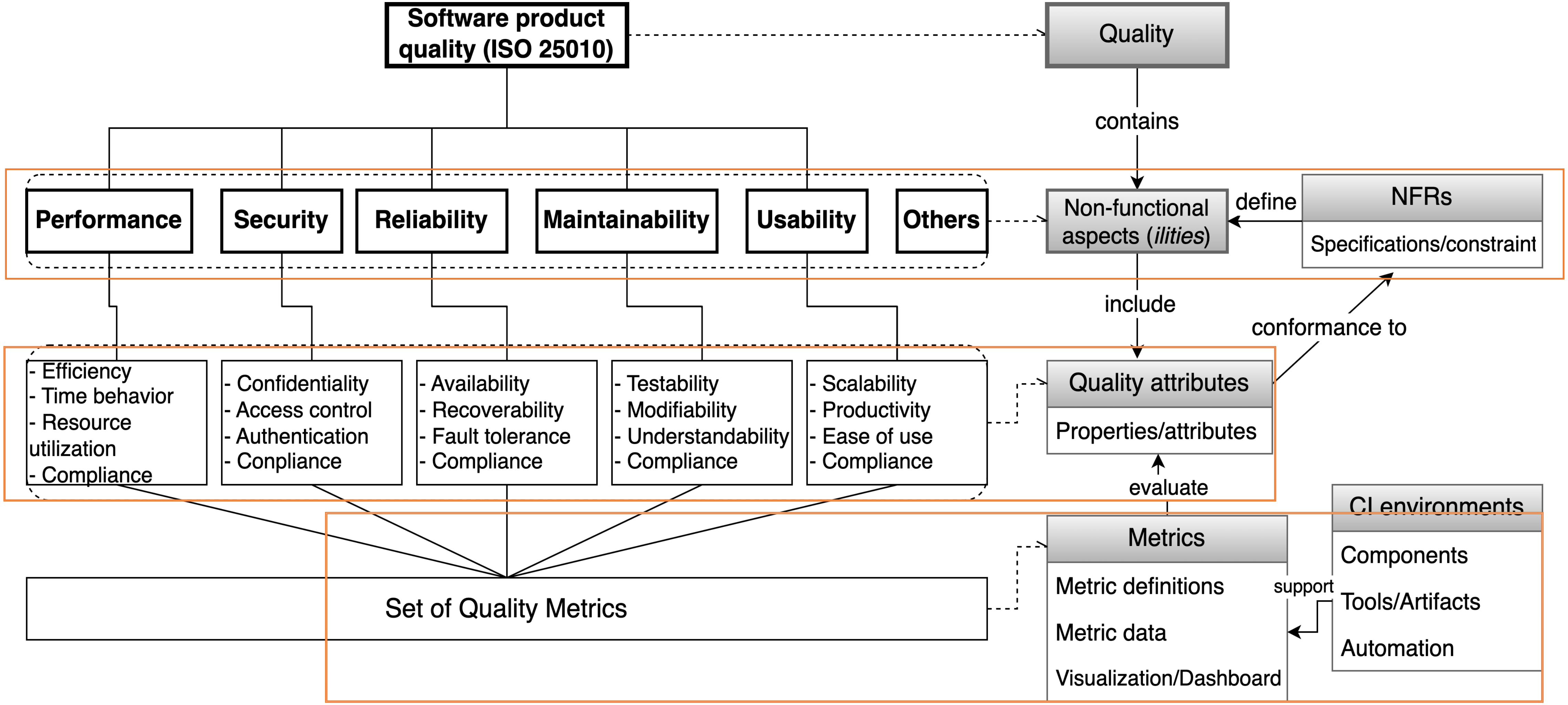
[1] Yu, L., Alégroth, E., Chatzipetrou, P., & Gorschek, T. (2020). Utilising CI environment for efficient and effective testing of NFRs. *Information and Software Technology*, *117*, 106199.
[2] Yu, L., Alégroth, E., Chatzipetrou, P., & Gorschek, T. (2023). Automated NFR testing in Continuous Integration Environments: A multi-case study of Nordic companies. Empirical Software Engineering.

State of the art (SOTA)

SOTA: quality model

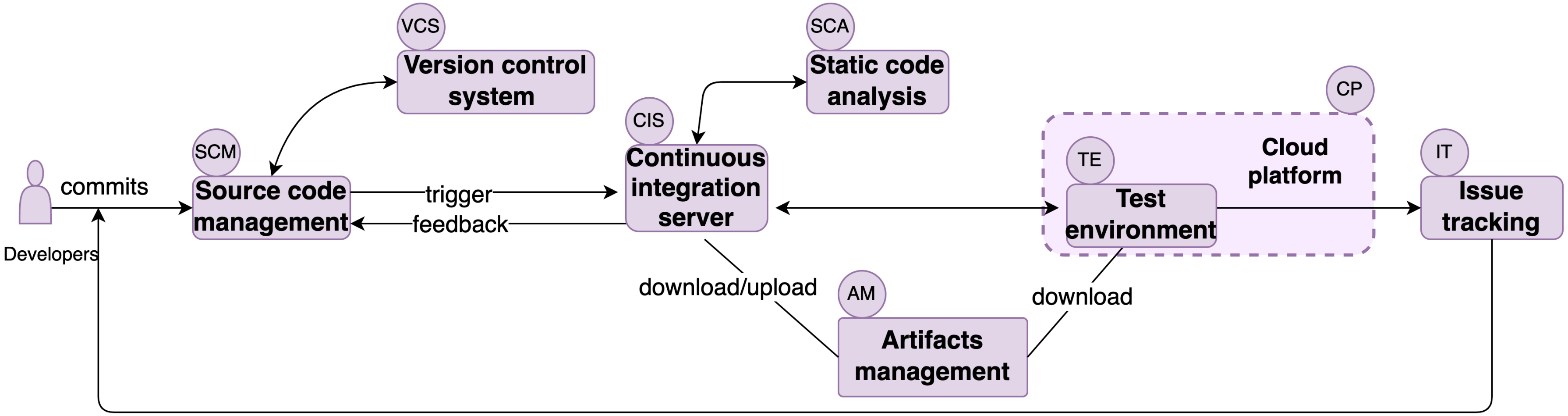


SOTA: ISO mapping

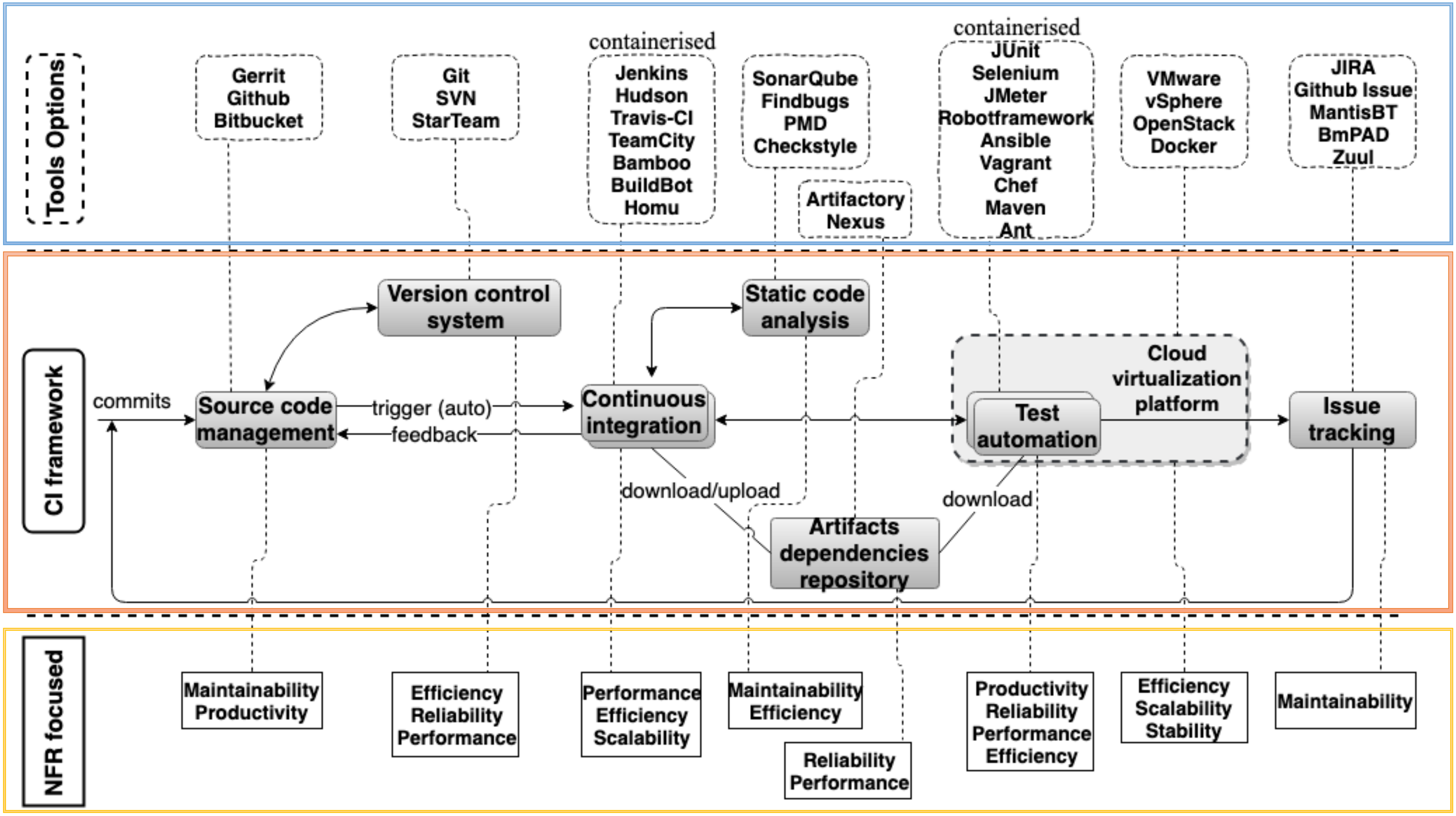


SOTA: CI model

- From component view[1]

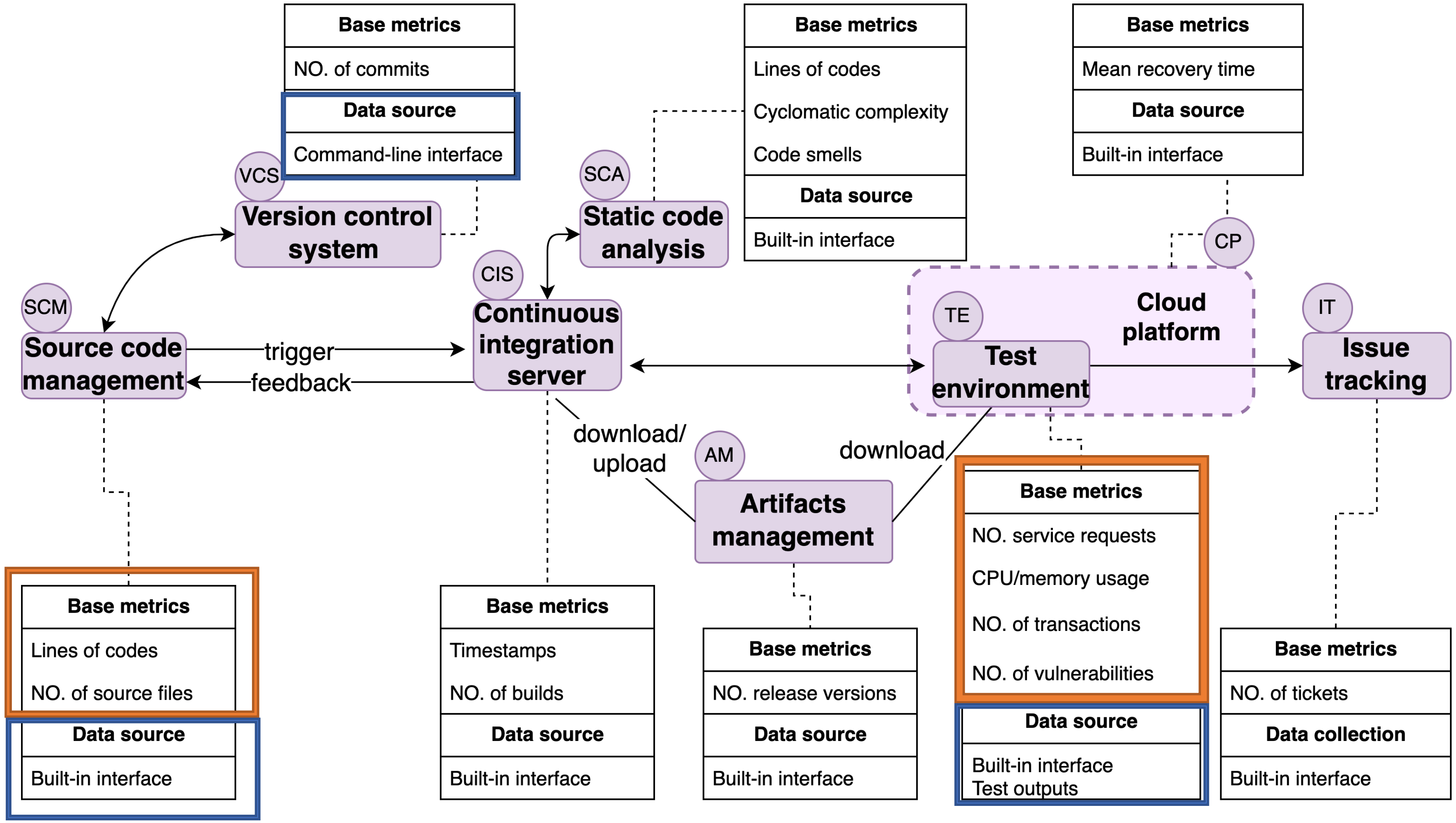


[1] Yu, L., Alégroth, E., Chatzipetrou, P., & Gorschek, T. (2020). Utilising CI environment for efficient and effective testing of NFRs. *Information and Software Technology*, *117*, 106199.


Tools Options
Gerrit
Github
Bitbucket
Git
SVN
StarTeam
containerised
Jenkins
Hudson
Travis-CI
TeamCity
Bamboo
BuildBot
Homu
SonarQube
Findbugs
PMD
Checkstyle
Artifactory
Nexus
containerised
JUnit
Selenium
JMeter
Robotframework
Ansible
Vagrant
Chef
Maven
Ant
VMware
vSphere
OpenStack
Docker
JIRA
Github Issue
MantisBT
BmPAD
Zuul
CI framework
commits
Version control system
Source code management
trigger (auto)
feedback
Continuous integration
Static code analysis
Cloud virtualization platform
Test automation
Issue tracking
download/upload
download
Artifacts dependencies repository
NFR focused
Maintainability
Productivity
Efficiency
Reliability
Performance
Performance
Efficiency
Scalability
Maintainability
Efficiency
Reliability
Performance
Productivity
Reliability
Performance
Efficiency
Efficiency
Scalability
Stability
Maintainability

A multi-case study

Case study: CI model – metric view



Base metrics
NO. of commits
Data source
Command-line interface

Base metrics
Lines of codes
Cyclomatic complexity
Code smells
Data source
Built-in interface

Base metrics
Mean recovery time
Data source
Built-in interface

Base metrics
Lines of codes
NO. of source files
Data source
Built-in interface

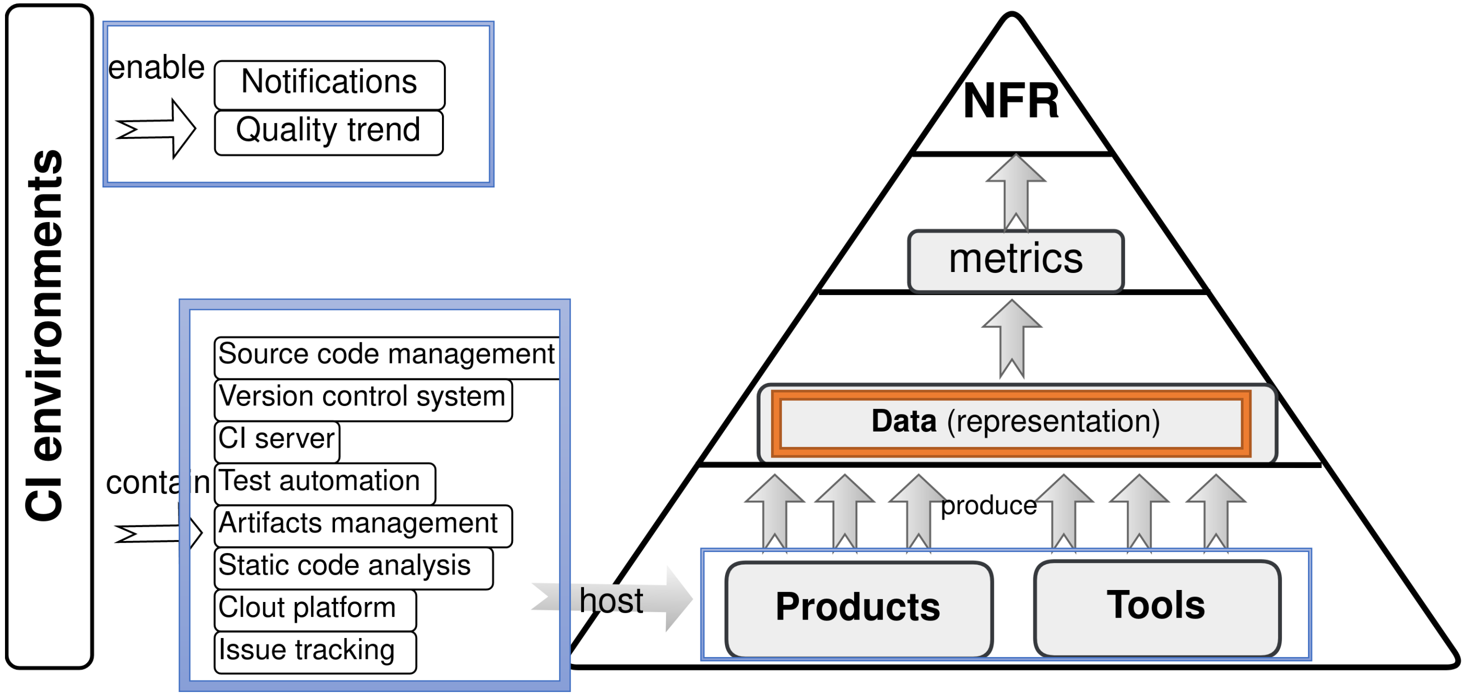
Base metrics
Timestamps
NO. of builds
Data source
Built-in interface

Base metrics
NO. release versions
Data source
Built-in interface

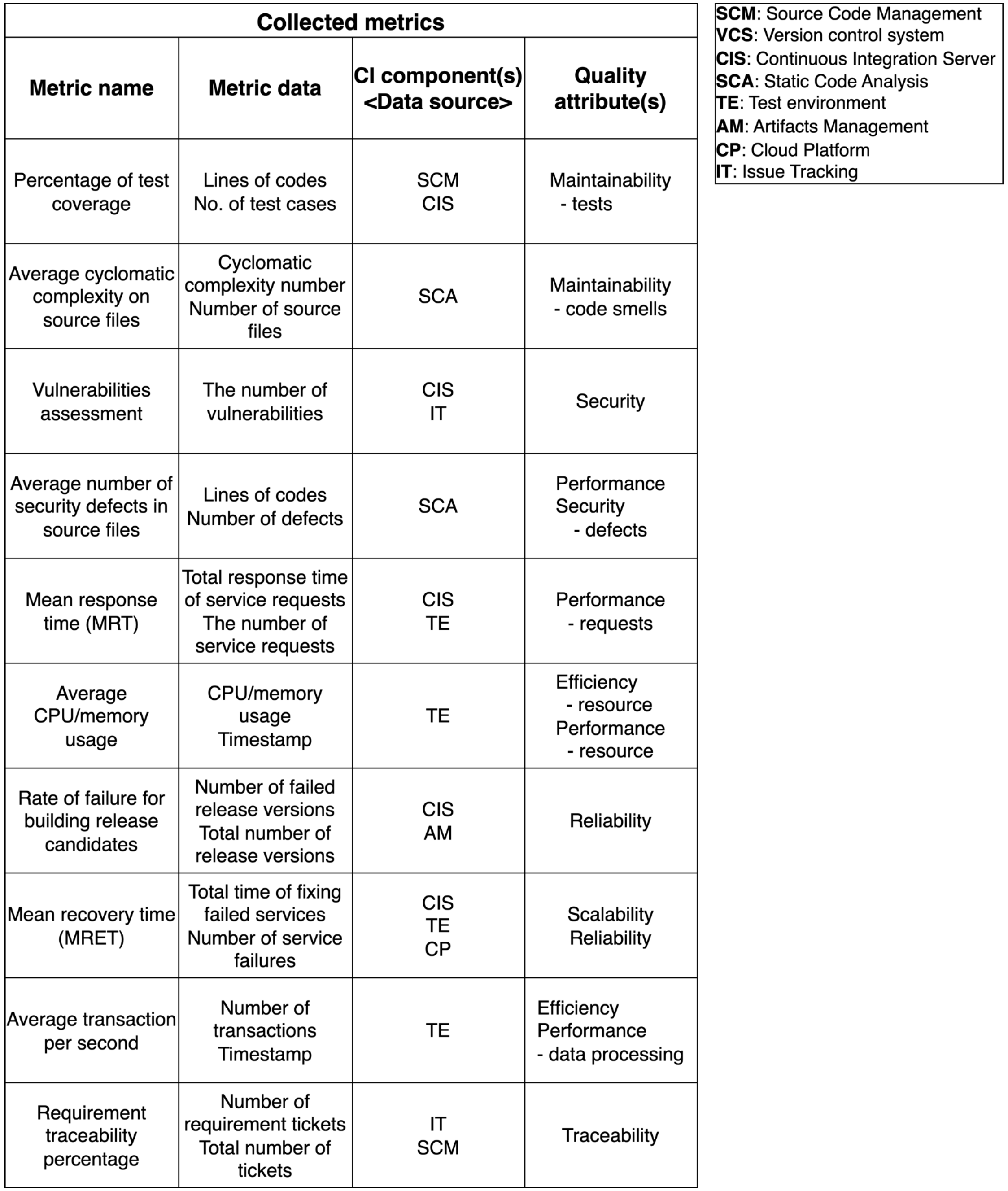
Base metrics
NO. service requests
CPU/memory usage
NO. of transactions
NO. of vulnerabilities
Data source
Built-in interface
Test outputs

Base metrics
NO. of tickets
Data collection
Built-in interface

Case study: CI environment data



Case study: metrics

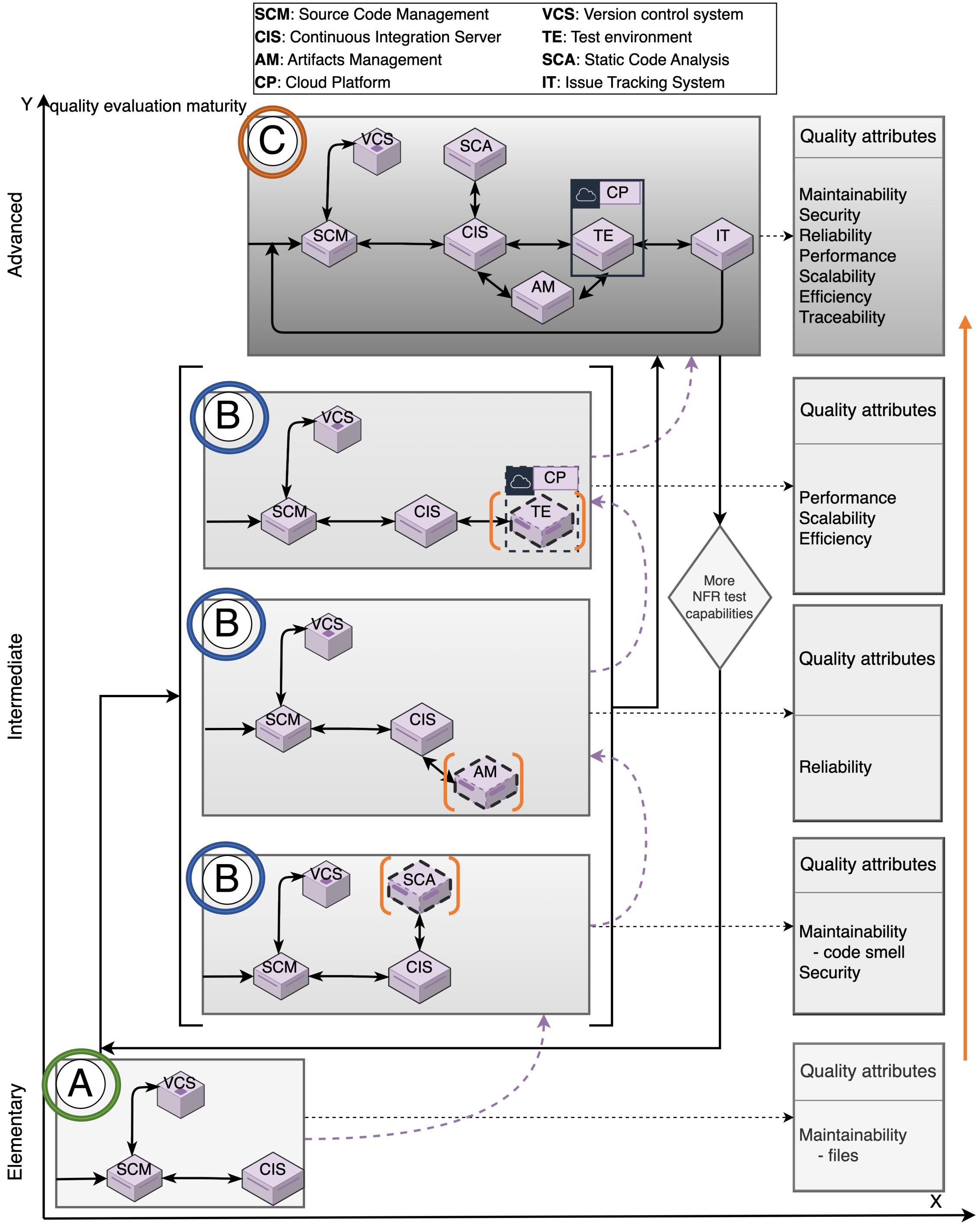


Collected metrics			
Metric name	**Metric data**	**CI component(s) <Data source>**	**Quality attribute(s)**
Percentage of test coverage	Lines of codes No. of test cases	SCM CIS	Maintainability - tests
Average cyclomatic complexity on source files	Cyclomatic complexity number Number of source files	SCA	Maintainability - code smells
Vulnerabilities assessment	The number of vulnerabilities	CIS IT	Security
Average number of security defects in source files	Lines of codes Number of defects	SCA	Performance Security - defects
Mean response time (MRT)	Total response time of service requests The number of service requests	CIS TE	Performance - requests
Average CPU/memory usage	CPU/memory usage Timestamp	TE	Efficiency - resource Performance - resource
Rate of failure for building release candidates	Number of failed release versions Total number of release versions	CIS AM	Reliability
Mean recovery time (MRET)	Total time of fixing failed services Number of service failures	CIS TE CP	Scalability Reliability
Average transaction per second	Number of transactions Timestamp	TE	Efficiency Performance - data processing
Requirement traceability percentage	Number of requirement tickets Total number of tickets	IT SCM	Traceability

SCM: Source Code Management
VCS: Version control system
CIS: Continuous Integration Server
SCA: Static Code Analysis
TE: Test environment
AM: Artifacts Management
CP: Cloud Platform
IT: Issue Tracking

Case study: a roadmap

- Evolve CI environments to meet different needs of quality attribute evaluation



Challenges (1/2)

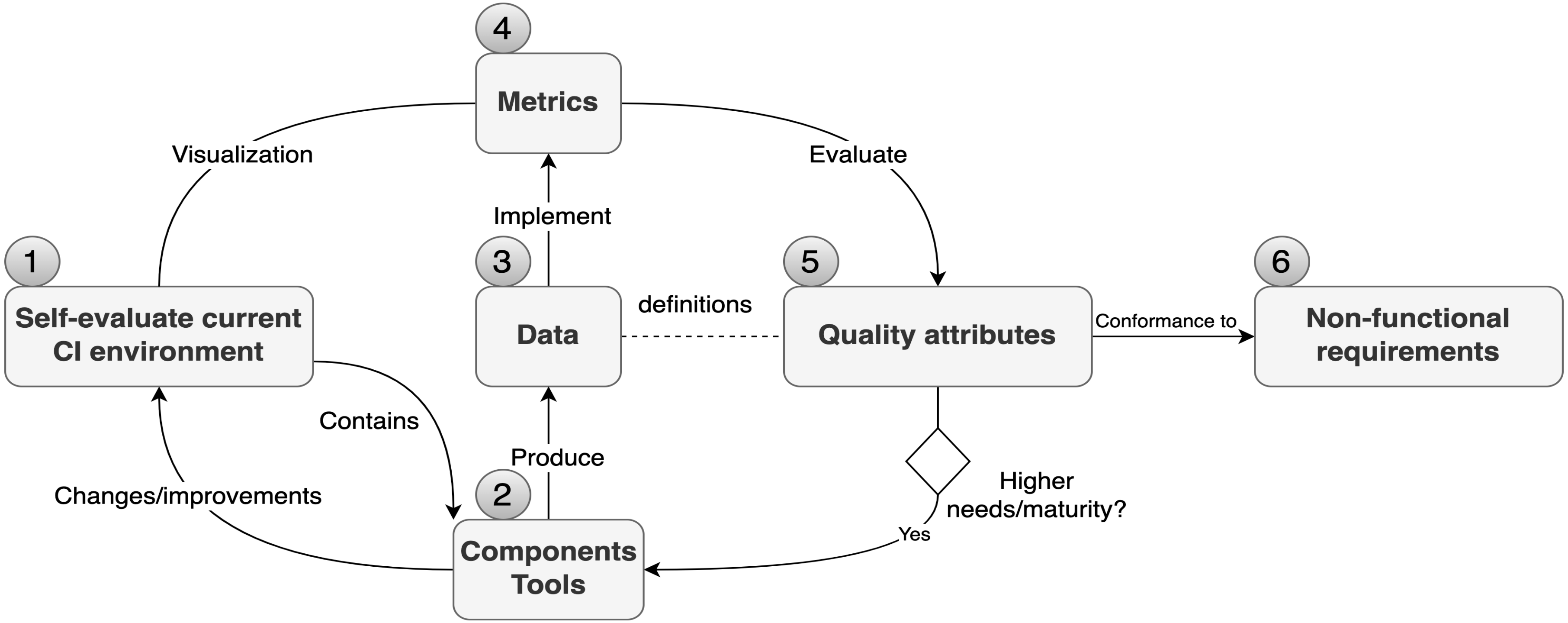
- Project scope
  - Measurable quality level
  - Insufficient data
  - Complexity of software dependencies

Challenges (2/2)

- Technique scope
  - Unstable CI environment
  - Lack of tools to evaluate specific quality attribute (e.g., usability)
- Organizational scope
  - Continuous quality monitoring

Lesson learned

Lesson learned - process level



Lesson learned - project level

- Maintain "*right level*" of quality
  - Quality attributes can be evaluated from multiple aspects
  - Metric definition
- Multiple metrics to evaluate on metrics
  - Consider limited resources
- International standards[1]
  - Comparable results

[1] ISO/IEC-25023: Systems and Software Engineering: Systems and Software Quality Requirements and Evaluation (SQuaRE): Measurement of System and Software Product Quality. ISO (2016)

Lesson learned - technique level

- Unstable CI environments
  - Feedback time
  - Flaky tests
- Tools for the evaluation
  - One tool does not fit all evaluation
  - Caution before introducing new tools
- Monitoring
  - Trade-offs between quality attributes

Get in touch!

Liang Yu

liang.yu@bth.se

SERL Sweden
LEADING SOFTWARE ENGINEERING