This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 732253.

D4.4 Proof-of-Concept

V0.6

Programme	**H2020**		
Funding scheme	RIA - Research and Innovation action		
Topics	ICT-10-2016 - Software Technologies		
Project number	732253		
Project name	Quality-aware rapid software development		
Project duration	November 2016 – October 2019		
Project website	www.q-rapids.eu		
Project WP	WP4 – Q-Rapids Platform		
Project Task	T4.4 Deployment of the scientific results according to the Q-Rapids architectural blueprint		
Deliverable ID & name	D4.4 Proof-of-Concept		
Deliverable type		R	Document, report
	X	DEM	Demonstrator, pilot, prototype
		DEC	Websites, patent fillings, videos, etc.

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 732253.

		OTHER	Material that does not belong to any specified category
		ETHICS	Ethics requirement
Deliverable version	0.6		
Contractual delivery	31/01/2018		
Delivered	24/01/2018		
Responsible beneficiary	ITTI		
Dissemination level	X	PU	Public
		CO	Confidential, only for members of the consortium (including the Commission Services)
		EU-RES	Classified Information: RESTREINT UE (Commission Decision 2005/444/EC)
		EU-CON	Classified Information: CONFIDENTIEL UE (Commission Decision 2005/444/EC)
		EU-SEC	Classified Information: SECRET UE (Commission Decision 2005/444/EC)

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 732253.

Version history			
Version no.	Date	Description	Author
V0.1	11/2017	Table of contents, general structure of the document, introduction section	Michał Choraś, Damian Puchalski, Rafał Renk, Szymon Panfil, Tomasz Springer (ITTI)
V0.2	12/12/2017	Preliminary version of sections content	
V0.3	18/12/2017	Some minor modifications	
V0.4	08/01/2018	Pre-final version of the document, Executive Summary and Conclusions added (ready for review)	
V0.5	12/01/2018	Changes based on FhG and UPC comments	Michał Choraś, Damian Puchalski, Rafał Renk, Szymon Panfil, Tomasz Springer (ITTI), Silverio Martinez, Axel Wickenkamp, Andreas Jedlitschka (Fraunhofer IESE)
V0.6	24/01/2018	After review, final version	

Authors	
Organisation	Name
ITTI	Michał Choraś
ITTI	Damian Puchalski
ITTI	Rafał Renk
ITTI	Szymon Panfil
ITTI	Tomasz Springer
Reviewers	
Organisation	Name
Softeam	Antonin Abherve
Oulu	Pertti Karhapaa

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 732253.

Definition of the key terms and abbreviations	
A term or an abbreviation (alphabetical order)	Explanation of the term or the abbreviation
API	Application Programming Interface
DEM	Demonstrator
DoW	Description of Work
GUI	Graphical User Interface
JSON	JavaScript Object Notation
PoC	Proof-of-Concept
WP	Work Package

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 732253.

Contents

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 732253.

The list of figures

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 732253.

Executive summary

This document has been delivered to report the work done in T4.4 task and provides the overview on the proof of concept version of the Q-Rapids tool. According to the DoW, D4.4 was planned at Month 15 as DEM (Demonstrator) type of deliverable that has to *"demonstrate that the envisaged architecture is actionable and allows integration of bespoke and external components."*

The source code, installation package and manuals for the Proof-of-Concept deployment are available at the project Basecamp internal space.

This document provides overview and description of particular components and is structured according to the system data flow. In general, we can report that the Q-Rapids architecture modules are developed, integrated and operational.

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 732253.

1. Introduction

1.1 Motivation

The goal of this report is to present results of the first phase of task T4.4 "Deployment of the scientific results according to the Q-Rapids architectural blueprint" focusing on the development and early integration of the Q-Rapids proof-of-the concept, ended at M15.

According to the DoW, the project schedule and scope, the minimum output of this T4.4 phase is an up-and-running product with all conceptual and design assumptions valid and ability for separate components (developed by WP1-3) communication.

Formally, this deliverable at M15 is marked as "DEM" (Demonstrator), however we decided to report the work done towards the integrated and operational Q-Rapids platform aside of the running tool prototype.

1.2 Intended audience

The intended audience of this document are the members of the Q-Rapids consortium.

1.3 Scope

The scope of this document is the full Q-Rapids project, in all its Work Packages and along its entire timeline, however WP4, WP1, WP3 (technical work packages producing components to be integrated within the platform) and WP5 (related to the evaluation of tool) are mostly related to the scope of the current report.

1.4 Relation to other deliverables

This deliverable is related to the following deliverables:

- D1.2: Data gathering and analysis proof-of-concept (Month 15) – first version of the data gathering and analysis module, D1.2 report contains more detailed information on data acquisition for the quality analysis purposes,
- D3.2: Dashboard proof-of-concept (Month 15) – first version of the Dashboard prototype, delivered both as the installation package and report/documentation (containing more detailed information on the Strategic Dashboard module),
- D4.3: Architectural Blueprint (Month 9) – technical description of the software architecture that was a basis for the prototype development,
- D4.5: Consolidated version (Month 24) and D4.6: Final release (Month 33) – focused on implementation and integration of the developed WP1-WP3 techniques to enhance operational capabilities of the current Proof-of-Concept,
- D5.3 Evaluation of pilot cases (Month 15 for PoC) – evaluation of the current version of the tool based on the results from case studies.

1.5 Structure of the deliverable

This deliverable is organised into the following sections:

- Section 2 presents the general approach and data flow in the current Proof-of-Concept,
- Section 3 presents data sources used to feed the PoC,
- Section 4 presents the approach to data collection, synchronization and storage for further use,
- Sections 5 and 6 present the approach to data processing, analysis and generation of metrics, quality factors and Key Strategic Indicators,
- Section 7 presents visualization capabilities of the PoC.

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 732253.

2. General approach and data flow in the Proof-of-Concept

The conceptual architecture of the Q-Rapids system is presented in Fig. 1. We use several data sources to measure the statistics (we call those metrics) related to the software code and software development process. In the current prototype, these metrics are retrieved from GitLab[1], Jenkins, JIRA, Redmine and SonarQube[2] project management tools and from SVN repositories. In the future it is possible to extend this list, if needed, since in different organizations and software houses, different tools are used. Obviously In some organizations only a subset of those tools will be used. However, in many cases it is just a matter of having the right connector between a project management tool and our prototype.

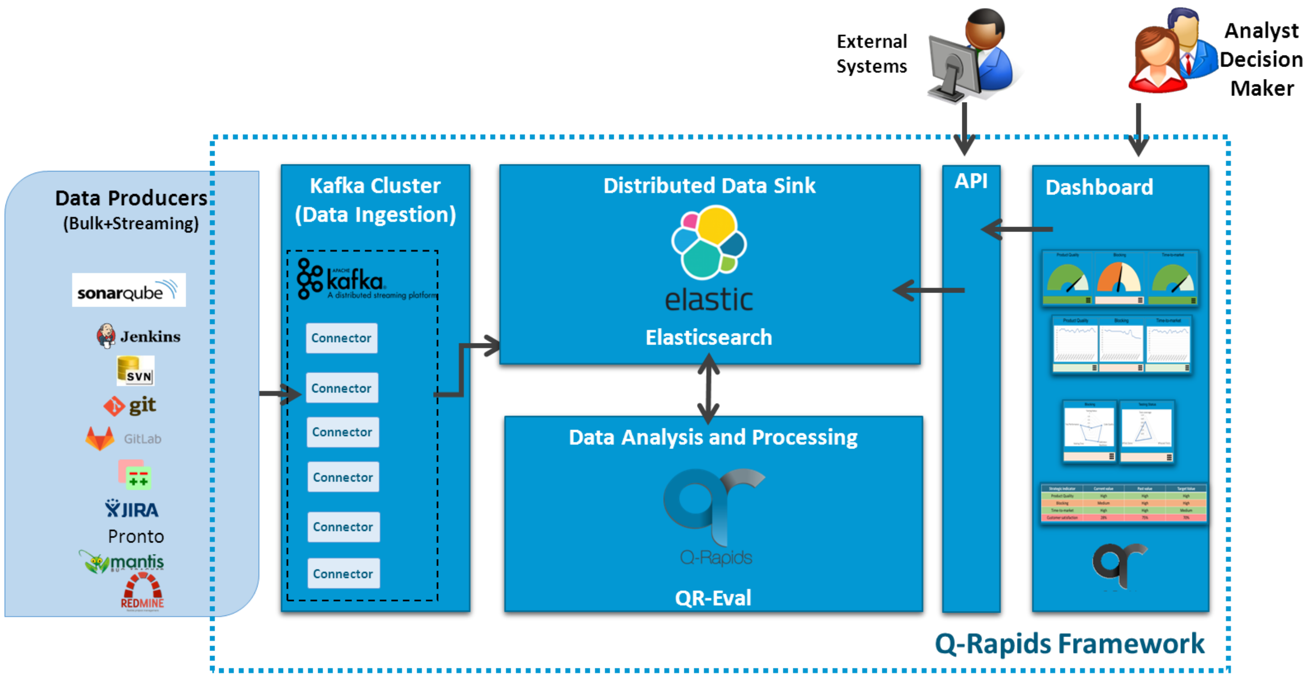


Figure 1: Conceptual architecture of the Q-Rapids system

As shown in the Fig. 1, the raw data maintained in the data producers tools (e.g. GitLab, SonarQube) feed distributed data sink and data analysis and processing modules. For this purpose we developed the connectors enabling acquisition of the data produced by software development tools. The Distributed Data Sink is fed through Kafka Cluster as the Data Ingestion layer. Synchronised data is preliminarily processed, filtered and anonymised. Synchronization can be done with different time intervals (e.g. once a day) both automatically and manually (please refer to section 4).

The main role of particular layers is as follows:

- Data producers – software development tools that maintain and provide raw data to be analysed in Q-Rapids tool,
- Data ingestion layer – provides the link between software development tools and Q-Rapids tool,
- Distributed Data Sink – for data maintenance and indexing,
- Data Analysis and Processing layer – for calculation of factors and indicators,
- Strategic Dashboard – for the presentation of results and interaction with the user.

[1] https://gitlab.com/
[2] https://www.sonarqube.org/

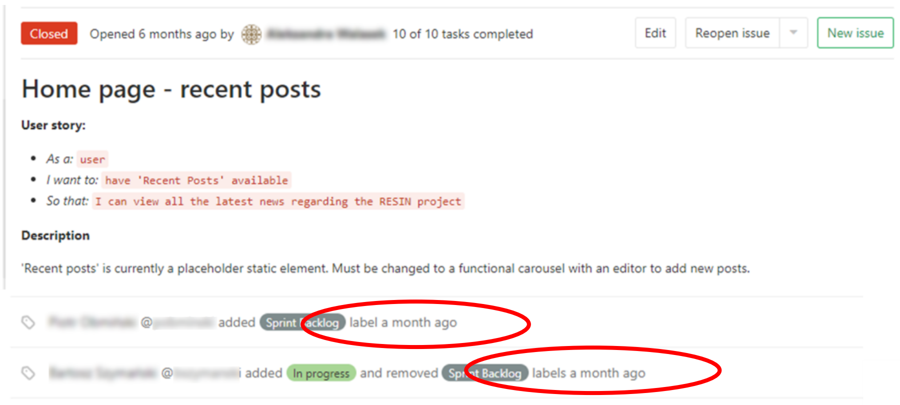
This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 732253.

3. Data producers

For the purpose of Q-Rapids we plan to feed our system with 3 categories of the source data:

- Code-related, including code repository statistics and information from tools used to static code analysis and quality review (e.g. SonarQube),
- Development process -oriented, including information from management tools related to software development projects effort allocation, scheduling, issue tracking, etc.,
- Software behaviour-related, based on the usage of already developed and deployed products, including e.g. aspects of usability, functionalities usage/failures, user-friendliness, etc.

At the current stage of the Q-Rapids and in the current version of the Proof-of-Concept, we have implemented connectors to code-related and process-related data, obtained from GitLab, Jenkins, JIRA, Redmine and SonarQube tools. At the current stage of the project, it is not sure if/how software behaviour-related aspects will be used.



Opened 6 months ago by [redacted] 10 of 10 tasks completed

Home page - recent posts

User story:

- *As a:* `user`
- *I want to:* `have 'Recent Posts' available`
- *So that:* `I can view all the latest news regarding the RESIN project`

Description

'Recent posts' is currently a placeholder static element. Must be changed to a functional carousel with an editor to add new posts.

[redacted] @[redacted] added Sprint Backlog label a month ago

[redacted] @[redacted]i added In progress and removed Sprint Backlog labels a month ago

Figure 2: Example of user story tracking in GitLab

For the ITTI case, we focus mainly on GitLab-based data that is additionally pre-processed due to the fact that structure of these data, scheme for labelling, etc. are highly dependent on the software development methodologies, and definition of development processes used in the organization that can be different for different product owners, project managers or projects even within the same company. For example, different programming teams may use different labels, even in the same project.

As presented in the Fig. 2, all the issues (e.g. user stories, bugs, etc.) maintained in the GitLab allow for extraction of time stamp for each modification, e.g. adding and removing labels, closing, opening and reopening issues, etc. Based on such data, we can track the course of the given project in terms of work intensification and then combine and/or compare these data to the code-related metrics, e.g. extracted from the SonarQube tool, such as code complexity, number of duplicated code lines in relation to total number of lines, etc.

4. Data Ingestion

As mentioned in the previous section, the data collection from the currently connected tools (GitLab, SonarQube) can be synchronised and stored for further processing and analysis both automatically with scheduled intervals, as well as manually if needed. Therefore historical data will be available for the analysis or re-analysis even after the project finalization, e.g. in order to compare different projects or to add a new data gathered from the other sources. Also, evolution of the code-related metrics can be built based on the code changes, tracked in the code repository (e.g. Git repository).

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 732253.

The main role of the Data Ingestion component provide the link between data producers (Software development tools) and the Q-Rapids framework (analytical and visualisation components). Both GitLab and SonarQube provide WebAPIs for external accessing of the data maintained in the tools. For example GitLab default API[3] allows for gathering issues-related data using JSON format GET requests. More information about SonarQube WebAPI can be found in the official technical documentation[4]. Also, both tools provide plugin libraries for most of the popular programming languages. However, for our purposes to fully control the analysed data, and to be able to extend the solution to other tools, we had to develop our own connectors.

5. Data processing and analysis

In the Q-Rapids project we use the hierarchical model of the software (code) quality developed as Quamoco approach (Wagner et al. 2015). We adopted the following hierarchy (from the low level to the high-level of the model):

1) Software quality metrics, derived directly from the source code/data sources,
2) Quality Factors, calculated based on the gathered metrics with the defined weights,
3) Key Strategic Indicators, calculated based on the aggregated and interpreted quality factors.

The hierarchy and basic relations between mentioned quality-related concepts are presented in Fig. 3 and Fig. 4.

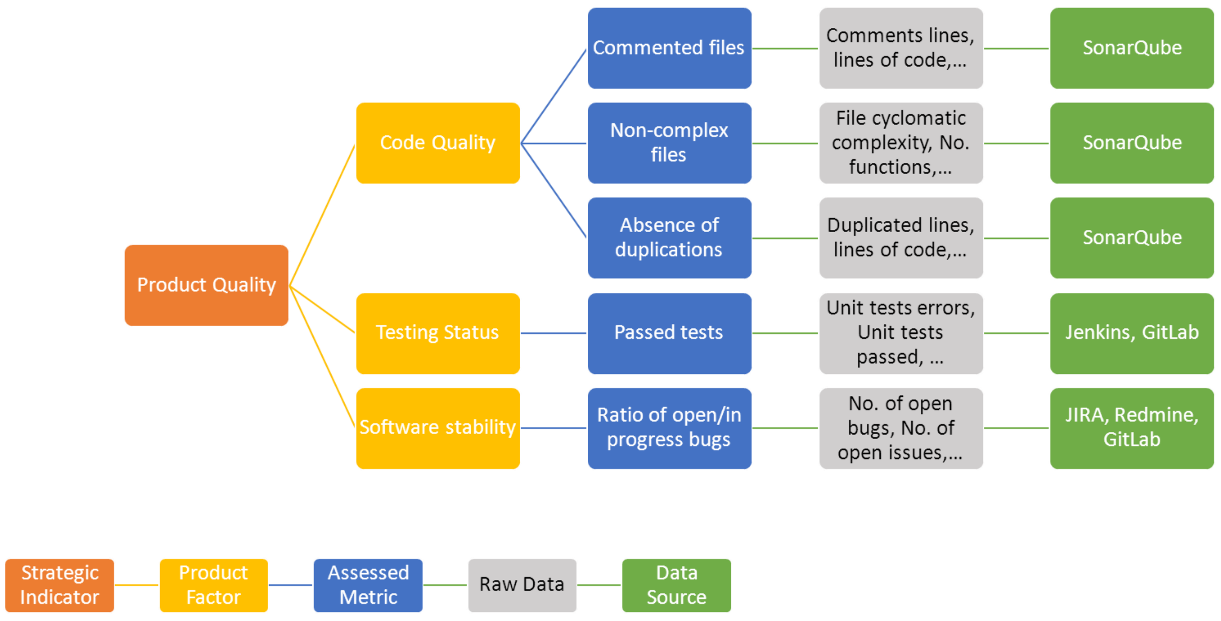


Figure 3: Relations between metrics, factors and indicators chosen for the Proof-of-Concept version of Q-Rapids tool (for Product Quality key strategic indicator)

[3] https://docs.gitlab.com/ee/api/README.html
[4] https://docs.sonarqube.org/pages/viewpage.action?pageId=2392172

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 732253.

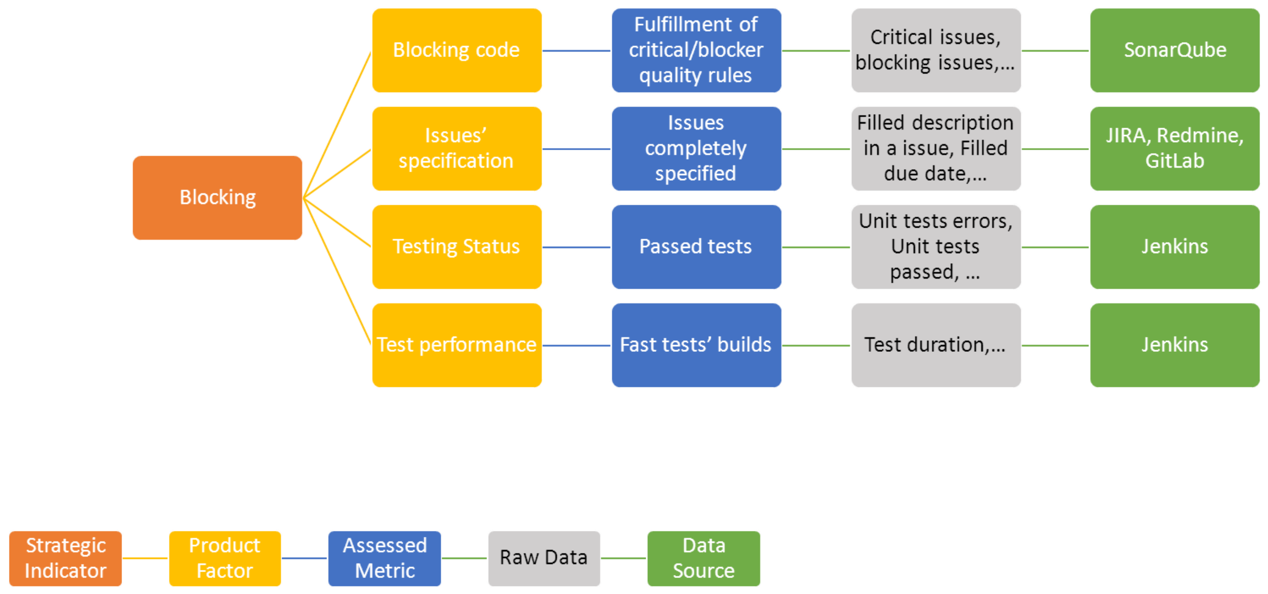
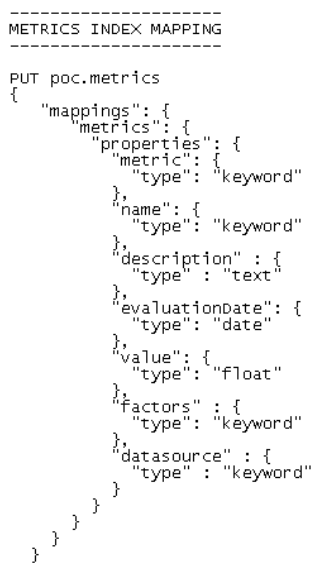


Figure 4: Relations between metrics, factors and indicators chosen for the Proof-of-Concept version of Q-Rapids tool (for Blocking key strategic indicator)

5.1 Data indexing

The next step after the collection of the data is the data indexing. In the Distributed Data Sink we use ElasticSearch cluster. In WP1 work package, we defined four indexed for three different classes of metrics - basic metrics are indexed as the raw data and as the normalised data.



```
---------------------
METRICS INDEX MAPPING
---------------------

PUT poc.metrics
{
   "mappings": {
      "metrics": {
        "properties": {
          "metric": {
            "type": "keyword"
          },
          "name": {
            "type": "keyword"
          },
          "description" : {
            "type" : "text"
          },
          "evaluationDate": {
            "type": "date"
          },
          "value": {
            "type": "float"
          },
          "factors" : {
            "type": "keyword"
          },
          "datasource" : {
            "type" : "keyword"
          }
        }
      }
    }
  }
}
```

Figure 5: Example of the quality metric index

As presented in the Fig. 5, metrics index includes such properties as:

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 732253.

- metric: unique identifier of metric used in Elasticsearch,
- name: defined name of the metric,
- description: description of metric calculation method (e.g. including calculation formula),
- evaluationDate: time of calculation,
- value: current value (normalised metrics can take on values in 0 to 1 range),
- factors: denotes quality factors constituted from given metric,
- datasource: address of the given index (measure) storage.

5.2 Calculation of metrics (ITTI case example)

Basic metrics are retrieved directly from the connected tools, as explained in section presenting data sources (Section 3). In practice, for real life software projects we currently collect the following metrics for the further aggregation:

- SonarQube-based:
  - Comments ratio,
  - Complexity,
  - Duplication density,
  - Non-blocking files,
- GitLab-based:
  - Percentage of passed tests,
  - Non-bug density.

The hierarchy and basic relations between SonarQube and GitLab metrics and calculated for ITTI case Quality Factors and Strategic Indicators are presented in Fig 6.

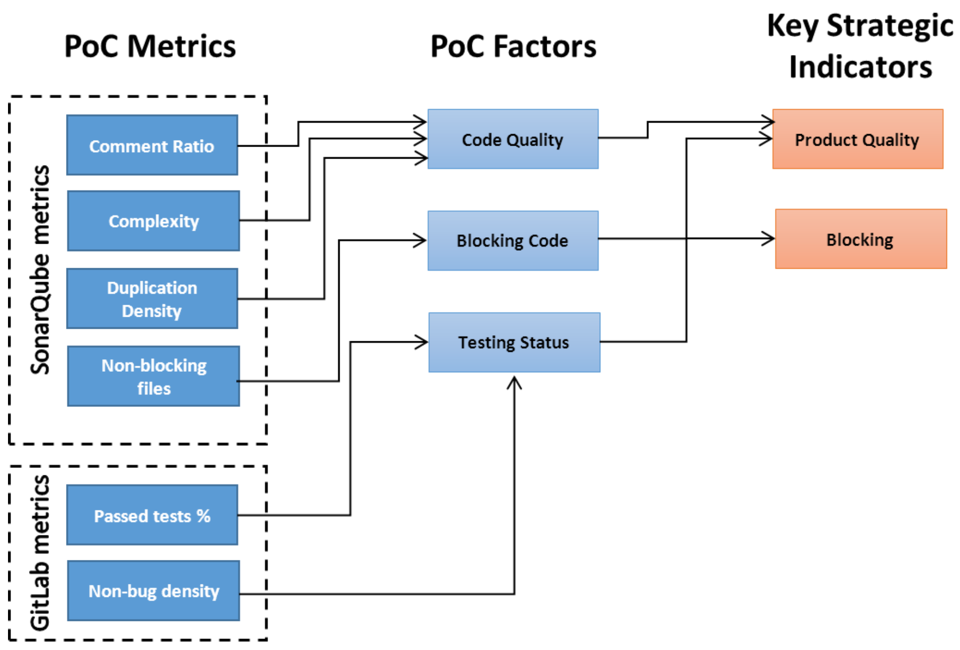


Figure 6: Relations between metrics, factors and indicators chosen for the Proof-of-Concept version of Q-Rapids tool (for ITTI case)

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 732253.

However, a number of basic quality metrics that are not used for calculation of quality factors and strategic indicators can be accessed using our connectors and can be visualised to provide added value for decision makers. Those include for example statistics calculated based on the GitLab data, related to programming sprints (e.g. presenting actual work in progress) and to sprint backlog content (e.g. presenting ratio of opened, closed or delayed issues in the backlog).

Also a huge number of SonarQube ratings is accessible using our connectors, such as project files and directory size, maintainability and reliability scores, etc.

5.3 Calculation of Quality Factors and Key Strategic Indicators (ITTI case example)

As explained in the introduction to the current section, Quality Factors are calculated based on the metrics that are aggregated using the defined weights. Currently we calculate following Quality Factors:

- Code Quality – based on comments density/ratio, code complexity scoring and duplicated code density,
- Blocking Code – based on non-blocking files metric, and
- Testing Status – based on non-bug density and percentage of passed tests.

Key Strategic Indicators that are the highest level of the adopted quality model that are currently calculated are:

- Product Quality – based on Code Quality and Testing Status,
- Blocking– based on Blocking Code factor.

6. Data visualisation (Strategic Dashboard)

For the visualization purposes in the Q-Rapids system we can use:

- Strategic Dashboard module developed in WP3,
- Kibana visualization capabilities,
- Additional GUI, developed for particular partner purposes (e.g. ITTI, Bittium).

In the Proof-of-Concept version of our tool we provide visualised aggregated data to the end-user through the web-based GUI. Calculated code characteristics (from metrics to strategic indicators) can be displayed in two different modes, including textual presentation of the data, and data projected on the charts. In addition, the Dashboard allows the user to display those data calculated for the current stage of the project, as well as charts presenting evolution of the metrics, factors and indicators over the time, from the beginning of the project. This allows for analysis of historical data if provided to the data gathering module. Another configurable property of the data visualization is the possibility to adjust grid of the time-based charts to the current needs and present evolution of data with granularity from days up to months.

In the next figure (Fig. 7), metrics related to the code quality calculated based on the real data from real software development projects. For example Comment Ratio, Complexity and Duplication Density metrics constitute "Code Quality" quality factor. Metric shown in the figure visualise historical data and their evolution.

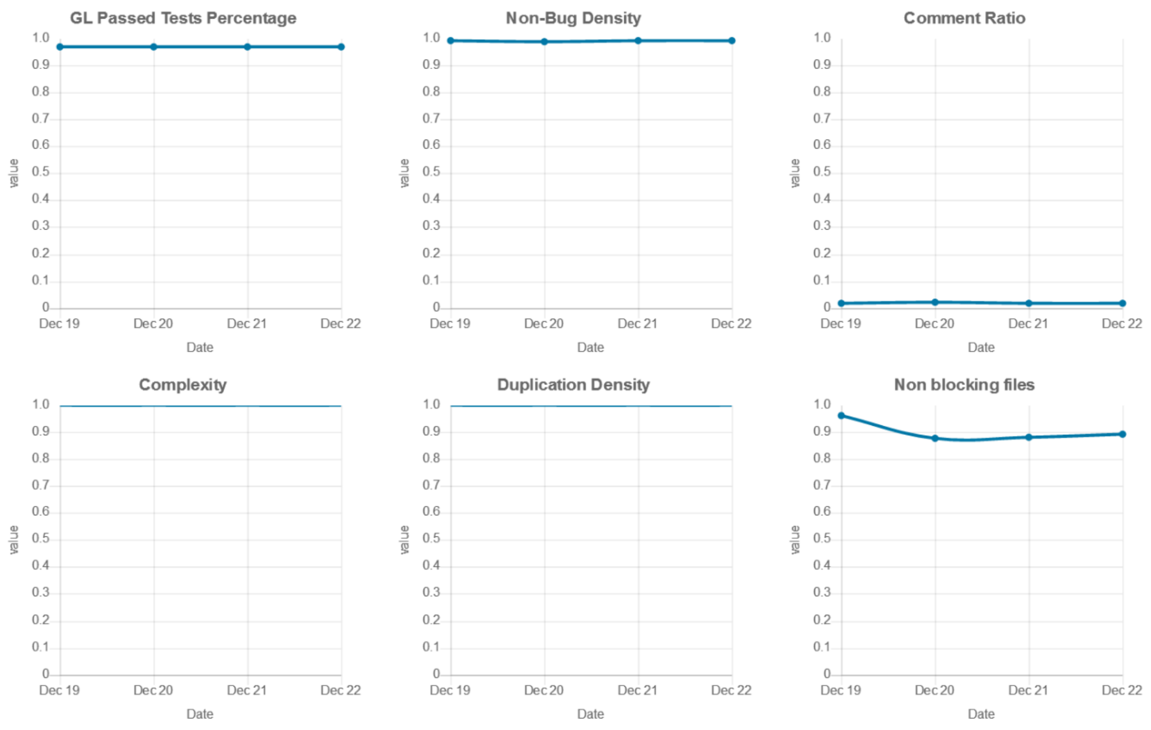


Figure 7: Visualization of quality metrics

Also quality factors can be visualised, including radar charts (Fig. 8) and evolution charts (Fig. 9) allowing decomposition of quality factors into more generic metrics and separate analysis of particular quality factor components.

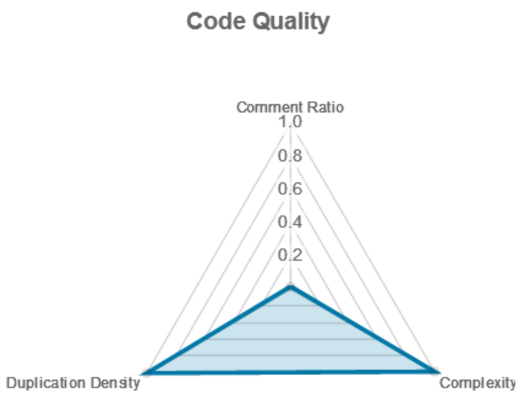


Figure 8: Visualization of quality factors using radar charts

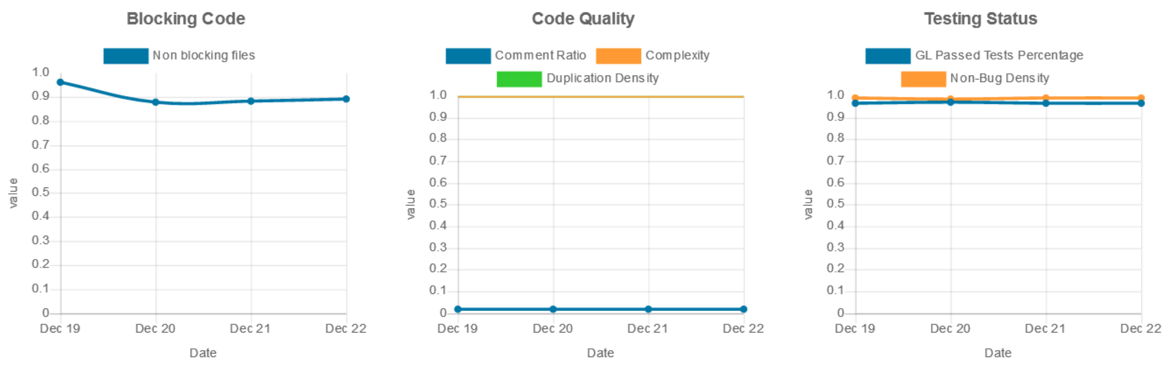


Figure 9: Visualization of quality factors using evolution charts

Finally, the most general, top-view of the quality analysis are Key Strategic Indicators allowing decision maker for quick evaluation of the quality-related aspects in particular software development project. This will allow decision maker to assess the code quality versus the quality requirements, specific in given company or organization. In Fig. 10 evolution (historical data) charts are shown, while Fig. 11 presents the current value of Key Strategic Indicators in the form of gauges.

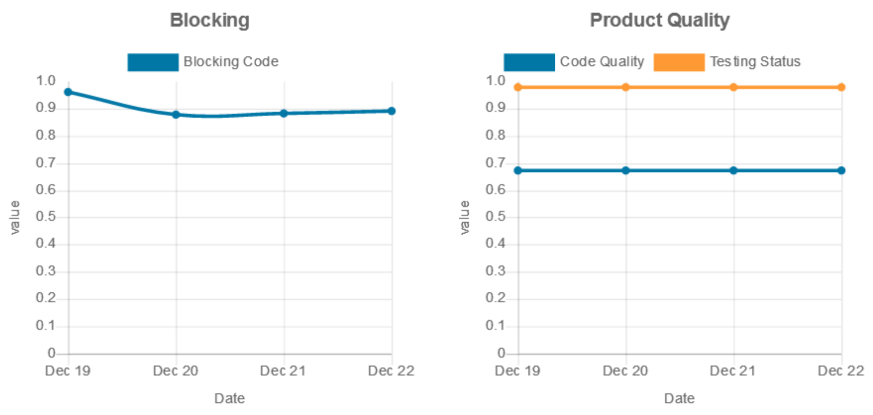


Figure 10: Visualization of Key Strategic Indicators using evolution charts

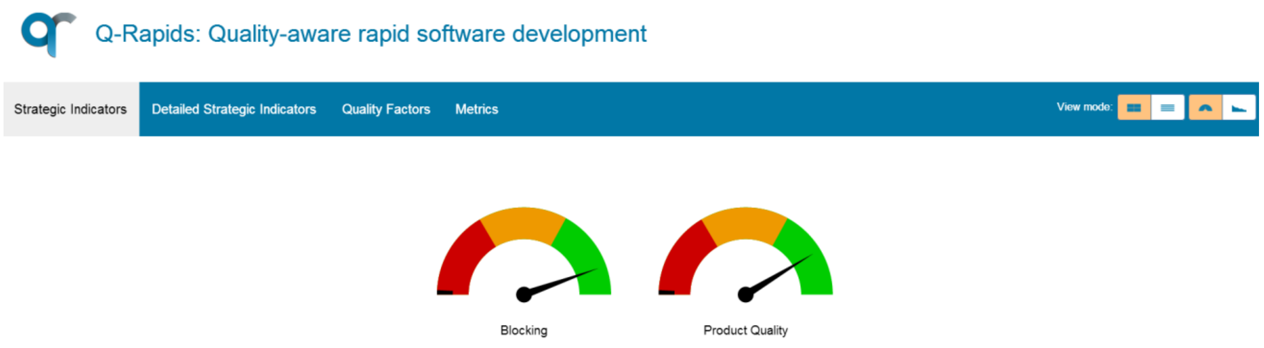

Q-Rapids: Quality-aware rapid software development
Strategic Indicators
Detailed Strategic Indicators
Quality Factors
Metrics
View mode:
Blocking
Product Quality

Figure 11: Visualization of strategic indicators (WP3 Dashboard)

More information about visualisation capabilities, as well as about Strategic Dashboard component design, development and deployment can be found in D3.2 report "Dashboard proof-of-concept".

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 732253.

Conclusions

Current document provides overview on the Proof-of Concept version of the Q-Rapids platform, scheduled for M15. In the document, all the modules comprising the tool have been presented in accordance to the system data flow, starting from the data production and ingestion, up to data analysis and visualisation.

All the functional requirements and architectural assumptions reported in previous WP4 deliverables have been satisfied, all the modules communicate each other and end-to-end flow is ensured, i.e. Proof-of-Concept downloads and analyses the raw data, calculates quality factors and strategic indicators, and provides visualisation capabilities through the WP3 Dashboard. Also, such functionalities as: selection of the visualisation mode (charts/textual data), selection of chart types for different analyses (radar chart for current factor/indicator values and evolution charts for historical data) and possibility to narrow time-span of visualized data have been implemented. Also, user of the tool can easily decompose Key Strategic Indicators into Quality Factors, and then Quality Factors into metrics, by simple clicking on relevant chart or chart area, and thus to preview higher or lower level of the quality analysis.

Proof of the concept is now being evaluated according to WP5 DoW, e.g. evaluation session at ITTI took place at 11/01/18.

Next release of the tool, extending its capabilities based on case studies evaluation is scheduled at M24 (Consolidated version).