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

D3.2 – Dashboard proof-of-concept

V1.0

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	1st November 2016 – 31st October 2019		
Project website	www.q-rapids.eu		
Project WP	WP*3 – Strategic Decision Making Dashboard*		
Project Task	Task *3.1 – Dashboard Specification*		
Deliverable type		R	Document, report
	X	DEM	Demonstrator, pilot, prototype
		DEC	Websites, patent fillings, videos, etc.
		OTHER	Material that does not belong to any specified category
		ETHICS	Ethics requirement
Contractual delivery	None		
Delivered			
Responsible beneficiary	UPC		
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)

Version history			
Version no.	Date	Description	Author
0.0	21/07/2017	Outline & Sprint planning	L. López
0.1	20/12/2017	Sections reviewed	L. López
0.2	01/01/2018	Subsections included	L. López
0.3	10/01/2017	Added download subsection to the outline	L. López
0.4	10/01/2018	Initial content	L. López, C. Gómez
0.5	12/01/2018	Structure reorganized, sections reordered and some content moved to an annex.	L. López
0.6	18/01/2018	Internal review	M. Oriol
0.7	19/01/2018	Screenshots and links to the installation package and source code included and general review. Version sent to the project internal review.	L. López
0.8	30/01/2018	General review considering internal reviewer's comments	L. López
1.0	31/01/2018	General review, version to be sent to the PO	L. López

Authors	
Organization	Name
UPC	Lidia López
UPC	Cristina Gómez
Reviewers	
NOKIA	Sanja Aaramaa
Fraunhofer IESE	Silverio Martinez

Disclaimer

The work associated with this report has been carried out in accordance with the highest technical standards and the Q-Rapids partners have endeavoured to achieve the degree of accuracy and reliability appropriate to the work in question. However, since the partners have not control over the use to which the information contained within the report is to be put by any other party, any other such party shall be deemed to have satisfied itself as to the suitability and reliability of the information in relation to any use, purpose or application.

Under no circumstances will any of the partners, their servants, employees or agents accept any liability whatsoever arising out of any error or inaccuracy contained in this report (or any further consolidation, summary, publication or dissemination of the information contained within this report) and/or the connected work and disclaim all liability for any loss, damage, expenses, claims or infringement of third party rights.

Definition of the key terms	
Elasticsearch	Elasticsearch is a search engine based on Lucene. It provides a distributed, multitenant-capable full-text search
Factor (related to the quality model element)	Attributes of parts of the product or the process. They need to be concrete enough to be measured.
qma-api-elastic	A module of the Q-Rapids Dashboard to read data from the analysis module (WP1)
Quality model	The main purpose of a quality model is to link the data gathered from some data sources to the strategic indicators.
Quality model assessment	The operationalization and execution of a quality model with specific values/measures for all its elements in a timestamp: strategic indicators, product factors, and assessed metrics
qrapids-dashboard	A module of the Q-Rapids Tool for visualizing the analysis module (WP1)
Strategic indicator	An aspect that a company considers relevant for the decision-making process.

Abbreviations	
API	Application Programming Interface
GUI	Graphical User Interface
PO	Project Officer
PoC	Proof-of-concept
SIs	Strategic Indicators

Contents

The list of tables 7
The list of figures 8
Executive summary 9
1 Introduction 10
1.1 Motivation 10
1.2 Intended audience 10
1.3 Scope 10
1.4 Relation to other deliverables 10
1.5 Structure of the deliverable 11
2 Component Overview 12
2.1 Role in the Q-Rapids Framework 12
2.2 Functional Description 12
2.2.1 Data Views 12
2.2.2 View Modes 13
3 Q-Rapids Dashboard Architecture 13
3.1 Quality Requirements 13
3.2 Components Description 14
3.2.1 Front end (GUI) 15
3.2.2 Back-end 15
3.2.3 SDM REST API 15
4 Strategic Decision Making Dashboard Implementation 16
4.1 Progress on Strategic Indicators 16
4.2 Progress on User Stories for Q-Rapids Dashboard 16
4.3 Development Process 20
4.4 Evaluation of Technical Aspects 21
5 Demonstration of Q-Rapids Dashboard 22
5.1 Demonstration of the Role-dependent Views and Project Current Status User Stories 22
5.2 Demonstration of the Role-dependent Views and Navigation User Stories 24
5.3 Demonstration of the Trend Analysis User Stories 26
6 Q-Rapids Dashboard Software and Installation 28
6.1 Q-Rapids Dashboard Software 28
6.1.1 Technical Specification 28
6.1.2 Q-Rapids Dashboard Component 28
6.1.3 QMA-API-Elastic Component 29

6.2 Q-Rapids Dashboard Installation 30
6.3 Download 31
Conclusion 31
Annex A. Quality Model Assessment (QMA) API Documentation 32
Annex B. Strategic Decision Making (SDM) REST API Documentation 35
Annex C. Development Roadmap 43
Annex D. Q-Rapids Dashboard Quick Guide 46
Annex E. Q-Rapids Dashboard Release Notes 48
Annex F. QMA-API-Elastic Release Notes 49
Annex G. How to Deploy Q-Rapids Dashboard 50

The list of tables

Table 1. Proof-of-concept User Stories Candidate 17
Table 2. Progress of User Story related to Role-dependent View (#56) 19
Table 3. Progress of User Story related to Trend Analysis (#85) 20
Table 4. Progress of User Story related to Red/Yellow/Green (#83) 20
Table 5 Q-Rapids Dashboard Roadmap 20
Table 6. qrapids-dashboard technical specification 28
Table 7. qrapids-qma-api-elastic tectnical specification 28
Table 8. class StrategicIndicator (QMA API) 32
Table 9 class Factor (QMA API) 33
Table 10 class Metric (QMA API) 33
Table 11 EvaluationDTO (QMA API) 33
Table 12 StrategicIndicatorEvaluationDTO (QMA API) 33
Table 13 FactorEvaluationDTO (QMA API) 34
Table 14 MetricEvaluationDTO (QMA API) 34
Table 15 StrategicIndicatorFactorEvaluationDTO (QMA API) 34
Table 16 FactorMetricEvaluationDTO (QMA API) 34
Table 17 Get Current Strategic Indicators Evaluations (SDM REST API) 35
Table 18 Get Historic Strategic Indicators Evaluations (SDM REST API) 36
Table 19 Get Current Detailed Strategic Indicators Evaluations (SDM REST API) 37
Table 20 Get Historical Detailed Strategic Indicators Evaluations (SDM REST API) 38
Table 21 Get Current Quality Factors Evaluations (SDM REST API) 39
Table 22 Get Historical Quality Factors Evaluations (SDM REST API) 40
Table 23 Get Current Quality Factors Evaluations (SDM REST API) 41
Table 24 Get Historical Metrics Evaluations (SDM REST API) 42

The list of figures

Figure 1.Simplified Q-Rapids Framework 12
Figure 2. Q-Rapids Tools Architecture (source deliverable D4.4) 13
Figure 3. qr-dashboard component architecture 14
Figure 4. Q-Rapids Dashboard layout 22
Figure 5. Strategic Indicators Data View – Current & Graphical (gauge charts) 23
Figure 6. Detailed Strategic Indicators Data View – Current & Graphical (radar charts) 23
Figure 7. Metrics Data View – Current & Textual (Table) 24
Figure 8. Detailed Strategic Indicator for Product Quality Strategic Indicator 25
Figure 9. Detailed Strategic Indicator for Testing Status Quality Factor 25
Figure 10. Strategic Indicators Data View - Historical & Graphical 26
Figure 11. Detailed Strategic Indicators Data View - Historical & Graphical 27
Figure 12. Metrics data View - Historical & Textual 27
Figure 13. REL17.09.15 (v 0.0.1) - presented to potential partner users 43
Figure 14. REL17.09.30 - internal release 43
Figure 15. REL17.10.15 (v 0.0.2) - presented to partner representatives 43
Figure 16. REL17.11.15 (v 0.1.0) - first deployment at use case premises 44
Figure 17. REL17.11.30 (v 0.1.2) - some improvements 44
Figure 18. REL17.12.15 (v0.1.3) - proof-of-concept evaluation 44
Figure 19. REL18.01.15 - internal release (user management) 45
Figure 20. REL18.01.31 - internal release (refactoring) 45

Executive summary

This document is the D3.2 proof-of-concept demonstrator, an operational prototype including the first set-up of the dashboard. The main aim of this document is to demonstrate the overall concept of the dashboard to the use case providers in the consortium. Focusing on demonstrating the correct computation of some strategic indicators from real data coming from data gathering and analysis modules (WP1).

The document includes:

- the dashboard software component design derived from the dashboard specification documented in the previous deliverable D3.1,
- a report on the implementation progress made of epics and user stories included in the dashboard backlog (also reported in D3.1),
- a demonstration of the deployment of the dashboard module of the Q-Rapids tool, and
- the software solution (source code) together with technical and user oriented documentation.

1 Introduction

The overall goal of this document is to provide a demonstration of the first set-up of the architecture and running modules for the strategic decision making during the proof-of-concept phase. Proving the suitability of computing some strategic indicators from real data coming from the data gathering and analysis modules.

1.1 Motivation

Q-Rapids is a data-driven project, which should provide this data to the end-user in an attractive and user-friendly way. This data visualization should support decision makers in their decisions related to the quality requirements management. In order to provide a high-level information to be used by decision makers, the data gathered is aggregated in strategic indicators..

The motivation of this document is to demonstrate that Q-Rapids Dashboard has been integrated into the Q-Rapids Tool, and it can be used to support decision makers visualizing strategic indicators and the rationale behind their assessment as a basis for this support. More advanced features increasing the support provided by Q-Rapids Dashboard will be included in future versions of the tool.

1.2 Intended audience

The main audience of this document are all the industrial partners (i.e. Bittium, Softeam, iTTi, and Nokia), who deployed the system at their premises to perform a first assessment of the Q-Rapids tool. Some of the documentation included in this document has been already shared with the industrial partners to support the deployment of the tool.

Additionally, any project member can use this document to be informed in the progress of the development and the instructions to install or use the dashboard. As the dashboard development will be iterative, by having access to the current status of the implemented functionalities (functional requirements) and behaviour (quality requirements), they can contribute to the next iterations of the development process. This document will be send to the PO and the reviewers assigned to the Q-Rapids. Finally, as this report is a public document, it will be accessible by any person interested on the strategic dashboard.

1.3 Scope

The scope of this document is the following months to proof-of-concept milestone (M15), until the release of the consolidated prototype that will be reported at month M24 in the deliverable D3.3. The information included is the result of the second phase of WP3 tasks (from month M7 to M15), and it will change as new functionality will be added to Q-Rapids Dashboard. This deliverable purpose is to demonstrate the technical feasibility of the Q-Rapids Tool, including the strategic indicator concept to support decision maker's decisions. Next phases of Q-Rapids project will have their own demonstration and reports: M24 (prototype), M33 (consolidation), and M36 (final release).

1.4 Relation to other deliverables

This deliverable is strongly related to the previous deliverable D3.1, which contains the dashboard specification used for the design and subsequent implementation. It is also related to deliverables D4.2 (product backlog) and D4.3 (Q-Rapids tool architectural blueprint).

It is also related to some contemporary deliverables. Deliverable D1.2 that reports the demonstration of the data gathering and analysis modules, directly responsible of providing the data visualized in the dashboard. And deliverable D4.4 that summarizes the proof-of-concept of the Q-Rapids Tool.

Finally, it is worth to mention that deliverable D5.3 contains the evaluation of Q-Rapids Dashboard as part for Q-Rapids Tool evaluation. This evaluation also includes a light assessment related to the strategic indicators computation included in this version.

1.5 Structure of the deliverable

This document is structured as follows. Section 2 includes an overview of the component, including the functional description and the role in the Q-Rapids Framework. Section 3 includes a short description of the Q-Rapids Dashboard software component, including the quality requirements that guided the architecture design. Section 4 reports the status of the work done related to the dashboard. This work is divided into two tasks, the first task referent to the demonstration of the feasibility of the assessment of strategic indicators using real data from the use cases, and the second task the development of the tool. Section 5 includes several screenshots of the different views in the tool as a demonstrator that the operationalized quality model assessment can be visualized using the Q-Rapids Dashboard. Finally, Section 6 includes some technical documentation relate to the dashboard component and its installation.

The main goal of this deliverable is giving a general view of the dashboard component, so the main body of the document contains a general description. Annexes include details related to specific issues. Annex A and Annex B includes the APIs technical documentation, complementing Section 3. Annex C includes the detailed roadmap followed for the component development, complementing the general description included in Section 4. Annex D includes Q-Rapids Dashboard Quick Guide supporting tool screenshots included in Section 5. Finally, complementing Section 6, Annex E and Annex F contain the release notes for the both software components related to the dashboard (qr-dashboard and qr-qma-elastic); and Annex G includes the detailed instructions to deploy Q-Rapids Dashboard tool.

2 Component Overview

2.1 Role in the Q-Rapids Framework

Q-Rapids project has three general strategic objectives, which focus on the impact of the project: *improve software quality*, *increase software life cycle productivity*, and *shorten software time to market*. In order to succeed, four scientific objectives have been derived from the general strategic objectives:

1. Collect and analyse runtime and design time data
2. Define a software life cycle of integration quality requirements and functional requirements
3. Elaborate strategic key indicators for decision makers to manage the development process
4. Provide tool support to a quality-aware software life cycle

Q-Rapids Dashboard is part of the third and fourth objective of providing tool support.

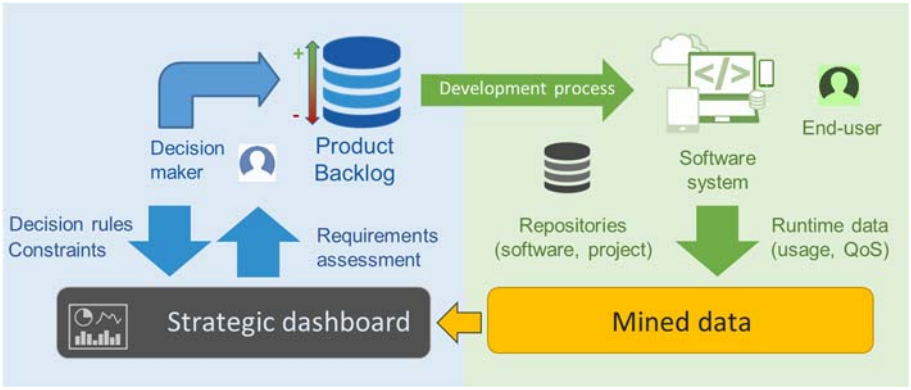


Figure 1.Simplified Q-Rapids Framework

Figure 1 shows a simplified version of the cycle described by the project objectives. Q-Rapids Dashboard is the bridge between the assessed data coming from the different data sources (repositories, software system and runtime data) and the decision makers. Decision makers will use the information visualized in the dashboard to feed their product backlog.

2.2 Functional Description

The main functionality of the current version of Q-Rapids Dashboard is providing several ways to visualize and explore the data. It includes *15 different views* corresponding to the combination of four *Data Views* and two V*iew Modes excluding one of the combinations*. The user can navigate among these views to explore the rationale behind the visualized results.

2.2.1 Data Views

The data views correspond to the different elements to be explored in the dashboard with different level of detail.

- *Strategic Indicators*. General view for the strategic indicators. For each indicator, the view includes its assessment values.
- *Detailed Strategic Indicators*. For each strategic indicator, the view provides information about the assessment of the factors impacting the strategic indicator.
- *Quality Factors*. For each quality factor, the view provides information about the assessment of the metrics used to calculate the factor.
- *Metrics*. General view for the metrics. For each metric the view includes its assessment values.

2.2.2 View Modes

The view modes allow the users to visualize the data in different ways.

- *Visualization Mode*: Graphical (using charts) or Textual (using tables)
- *Timing*: Current Value or Historical Data

Note: The tool does not provides a view for visualizing the current value of the metrics graphically.

3 Q-Rapids Dashboard Architecture

Figure 2 describes the conceptual architecture of the Q-Rapids Tool. 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, and
- Strategic Dashboard – for the presentation of results and interaction with the user.

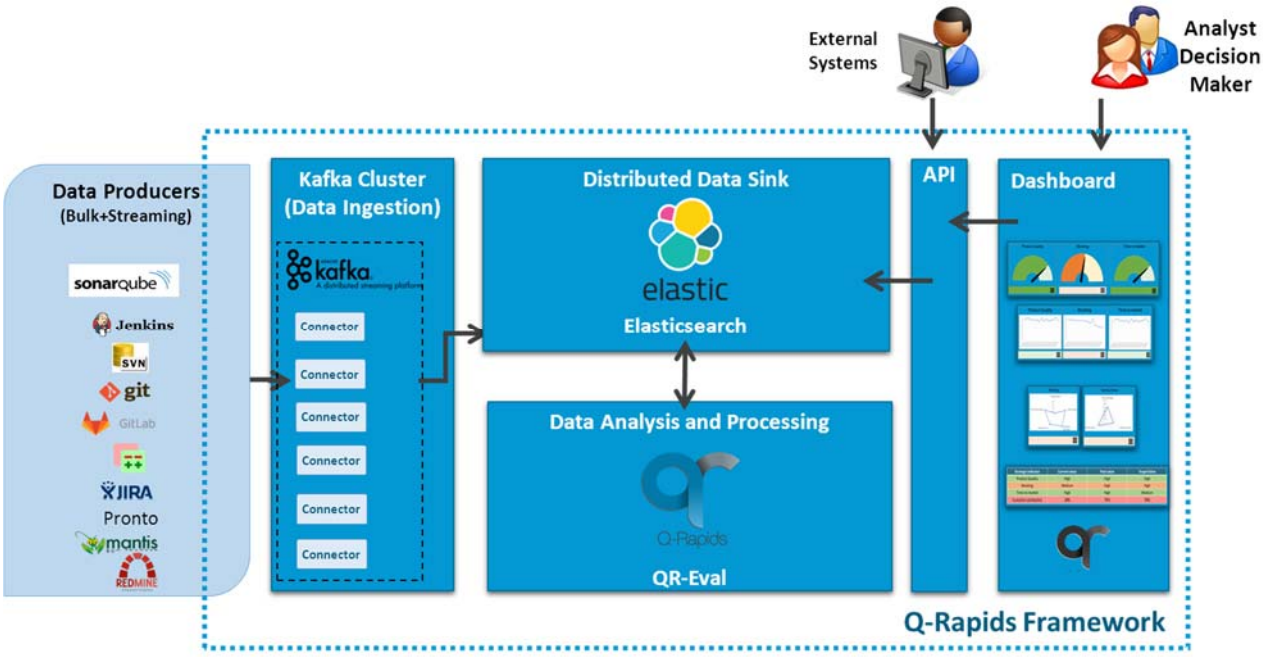


Figure 2. Q-Rapids Tools Architecture (source deliverable D4.4)

The scope of this section is the architecture for the Dashboard layer. The subsequent subsections will include the quality requirements guiding the design of this layer and the description of its internal modules.

3.1 Quality Requirements

Quality requirements guiding the design of this software components are *maintainability, portability* and *compatibly*.

Maintainability is an internal quality requirement for facilitating the software maintenance and evolution for the development team.

As Q-Rapids Tool is developed by several development teams, responsible of the different components of the tool. Because of the heterogonous need of the different components, the tool includes several technologies. We consider that we need to maximise the components' *portability* in order to facilitate the integration among them.

We also consider *interoperability* (compatibility) quality requirement responding to some use cases demands of connecting the Q-Rapids Tool to existing systems.

3.2 Components Description

Q-Rapids Dashboard software component follows the typical 3-tier architectural pattern (presentation, domain, and data) that improves the *maintainability* quality requirements. We also try to minimize the tier coupling addressing the *portability* quality requirement. Figure 3 shows the internal modules corresponding to the 3-tier architectural design. The tool is a web application, so we have the presentation tier in the front-end and the domain and data in the back-end, the communication between front-end and back-end is throw the SDM (Strategic Decision Making) REST API.

In order to increase the portability of the component, we developed an external component responsible for the communication with the Distributed Data Sink (Figure 2) layer. This allows that the technology used in the Distributed Data Sink layer (Elastic Search Engine) is transparent for the dashboard layer. This decision gave us to implement two separate components:

- *qrapids-dashboard*: Q-Rapids Dashboard tool.
- *qrapids-qma-api-elastic* (QMA API): external component that is included in qrapids-dashboard as an external library

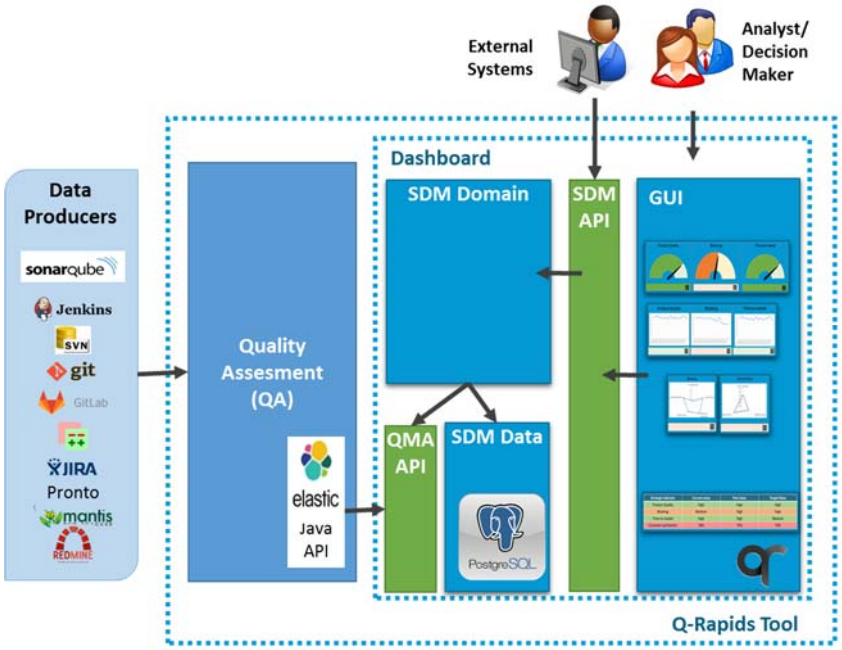


Figure 3. qr-dashboard component architecture

Following subsections include the details for the modules conforming the *qr-dashboard* component. Q-Rapids Dashboard is a web application, we grouped the components depending if they are part of the front-end (*GUI* module) or the back-end (*SDM Domain*, *SMD Data*, and *QMA API*). Finally, we included details about the SDM REST API used for the communication between them.

3.2.1 Front end (GUI)

This component corresponds to the presentation tier, in a web application is the component run in the client machine. We applied model-view-controller pattern in this model having in separated files views (html files rendering the results) and the controllers (responsible of the communication to the Domain layer).

We plan to follow the next rules and patterns for the user interface design (not all of them have been applied for the current version of the dashboard):

- The user is in control: Interaction modes, flexibility, cancelations, rectifications, customization, transparency…
- Minimize memorization: Reduce short term memory, default options, shortcuts, metaphors, hierarchies …
- Maintain a consistent interface: Inform about the context, consistency between windows, consistency between product families, respect conventions …
- Protection pattern: In the case of the result of executing some functionality concurs in a difficult undo status, the system will ask for confirmation to the user, e.g. remove a Strategic Indicator.

3.2.2 Back-end

As it is previously mentioned, the back end is composed by two modules: *SDM Domain* and *SDM Data*.

In order to minimize the coupling we applied the Adapter pattern, having a set of classes to gather information from the external component implementing the Distributed Data Sink layer and a set of classes accessing to the data managed by the dashboard, implemented in SMD module.

The adapters that gather information from the Distributed Data Sink module use the methods provided by the QMA API. Methods exposed by this API and the DTOs used for the data transfer are detailed in Annex A.

3.2.3 SDM REST API

The REST API allows to external systems consume the dashboard functionalities impacting positively in the *interoperability* quality requirement that we have for the system.

The REST API services are available in the URL <Q-Rapids dashboard deployment URL>/api/

The current version of the dashboard provide services related to gather the data exposed by the different views of the dashboard (see Section 2.2) grouped by the different data views:

- `<dashboard deployment URL>/api/StrategicIndicators`
- `<dashboard deployment URL>/api/DetailedStrategicIndicators`
- `<dashboard deployment URL>/api/Factors`
- `<dashboard deployment URL>/api/Metrics`

For each group, the dashboard provides a service generating the current assessment and other for the historical data:

- `<dashboard deployment URL>/api/StrategicIndicators/CurrentEvaluation`
- `<dashboard deployment URL>/api/StrategicIndicators/HistoricalData`

Detailed documentation about the services interface provided by this REST API are detailed in Annex B.

4 Strategic Decision Making Dashboard Implementation

4.1 Progress on Strategic Indicators

Part of the demonstration for this deliverable is the correct computation of some strategic indicators from real data. For the proof-of-concept, the consortium chose to include two strategic indicators:

Product Quality referring to the maintainability, reliability, and functional suitability of your software product. The factors used for its computation are:

- *Code Quality*: It measures the impact of code changes in source code quality.
- *Testing Status*: It measures the quality and stability level of executed tests during development.
- *Software stability*: It measures the most critical issues.

Blocking referring to conditions that negatively affects the progress of your project workflow. Blocking situations are e.g. stop-the-line, postpone a feature, etc. The factors used for its computation are:

- *Blocking code*: It measures the technical debt of software code, in terms of quality rule violations.
- *Quality Issues' specification*: It measures the state in which final information of a feature is included in the backlog, and hence it is ready to be developed.
- *Test performance*: It measures the time consumed for the execution of tests.

The current version computes the strategic indicators applying an arithmetic average among all the factors impacting on it. This computation is done in the Quality Assessment module. The details about the computation of quality factors and strategic indicators are included in deliverable D1.2.

4.2 Progress on User Stories for Q-Rapids Dashboard

The main goal of the Q-Rapids Tool proof-of-concept was to prove that we will be able to implement Q-Rapids project goal of aggregating data from heterogeneous data sources into strategic indicators. Providing a graphical visualization through a dashboard. All of this using real data from current tools used by the project use cases for collecting data. In order to fulfil this goal, we selected some features covering three of the seven user scenarios described in the dashboard specification (deliverable D3.1, Section 5.2):

- *View strategic indicators*. The current status of a product in terms of the strategic indicators defined for it will be shown at the beginning of the session (e.g., beginning of the day) and on user demand (user story #63, #83).
- *Trend analysis*. The trends of the Strategic Indicators in a given period of time will be shown on user demand (#67 and #85).
- *Role-dependent view*. The information shown (e.g., Strategic Indicators selected, level of detail) will depend on the role played by the user entering in the session (#56 and #72). The system should keep the traceability between the different levels of detail (#112) in order to visualise the rationale of the results of each level.

The numbers referred correspond to the issue number in the internal project management tool Redmine[1], reported in two deliverables produced by WP4 (D4.1 - Development environment and methodology and D4.2 – Product Backlog).

Table 1 includes all the user stories referent to the previous tree user scenarios. The proof-of-concept is limited to a single product, assessing the product once a day, independently of the step in its life-cycle.

[1] http://193.142.112.120/redmine

Therefore, user stories #67 (product comparison) and #72 (assessment at different life-cycle step) are out of the scope of the proof-of-concept. The user stories considered in the proof-of-concept are related to the epic *Visualisation* (#121).

Table 1. Proof-of-concept User Stories Candidate

User Scenario	ID	User story	PoC
View SIs	#63	As product manager, I want (online) quality visible over the products	Yes
View SIs	#83	As product manager/owner, I want to have red/yellow/green in terms of general quality, so that the problems are quickly visualised	Yes
Trend Analysis	#67	As product manager, I want to generate trend charts for QRs at level of product portfolio, so that we have comparable results over the project.	No
Trend Analysis	#85	As product owner, I want to analyse trend charts to see the evolution of an specific QR over the time, so that estimations can be produced	Yes
Role-dependent View	#56	As product manager, I want to include different viewpoints from user role/expertise, so that we see different information from roles in a different way	Yes
Role-dependent View	#72	As product manager, I want to include different viewpoints depending on the user role/expertise and the different product life-cycle step, so that different users can see different information	No
Role-dependent View	#122	As a product manager/owner, I want to understand the rationale behind the assessment reported by the dashboard	Yes

As an agile project, we cannot assess the progress of the user stories. New features related to the user story could appear in later stages of the project. Therefore, following tables include details about the progress of the features identified for user stories implemented from Table 1. The third column reports the progress, which could feel into one of the following categories:

- Open — This feature is in the initial 'Open' state.
- In Progress — This feature is being actively worked on at the moment.
- Resolved — The development team consider the development of the feature ended, the feature is ready to be included in a release. From here, issues are either 'Reopened' or are 'Closed'.
- Reopened — This feature was once 'Resolved' or 'Closed', but is now being re-examined. From here, issues are either marked In Progress, Resolved or Closed.
- Closed — This feature is complete and included in a release. From here, it could be eventually 'Reopened'.

In order to better understand the features included in this version and the relationship among them, we start with the user the user story referent to the role-dependent view (#56). We operationalised this user story as one feature of "Visualise different level of data" (#148). This feature has been divided in the different data views, for each data view (Strategic Indicators, Detailed Strategic Indicators, Factors, and Metrics), we needed to include a view corresponding to the current value of the assessed quality model in two view modes (graphical and textual).

Table 2 includes the progress of the features identified to support this user story. We included in the proof-of-concept different views depending on the quality model elements, we didn't introduce the concept of role yet. Therefore, when the concept of role is considered, we are going to reconsider whether we need more features to this user story.

Table 2. Progress of User Story related to Role-dependent View (#56)

Feature ID	Description	Progress
#142	View Strategic Indicators - Current value & Charts	Closed
#146	View Strategic Indicators - Current Value & Table	Closed
#165	View Detailed Strategic Indicators - Current value & Table	Closed
#143	View Detailed Strategic Indicators - Current value & Charts	Closed
#144	View Factors - Current Value & Charts	Closed
#147	View Factors - Current value & Table	Closed
#145	View Metrics - Current value & Table	Closed
#152	Navigation through different level of data	Closed

On one hand, we have the features related to visualise the current project status (

Table 2). On the other hand we have the user story related to analyse trends at level of product (#85). This user story has been operationalised as two features: view historical data (#210) and adding trend lines in the historical data charts (#160). Table 3 includes the features related to these user stories, the feature of viewing historical data has been divided in the different data views, and for each view in two views mode.

Table 3. Progress of User Story related to Trend Analysis (#85)

Feature ID	Description	Progress
#157 (#210)	View Strategic Indicators - Historical data & Chart	Closed
#161 (#210)	View Strategic Indicators - Historical data & Table	Closed
#241 (#210)	View Detailed Strategic Indicators - Historical data & Chart	Closed
#242 (#210)	View Detailed Strategic Indicators - Historical data & Table	Closed
#158 (#210)	View Factors - Historical data & Chart	Closed
#162 (#210)	View Factors - Historical data & Table	Closed
#159 (#210)	View Metrics - Historical data & Chart	Closed
#163 (#210)	View Metrics - Historical data & Table	Closed
#160	Add trend line to the historical data charts	Open

Finally, we considered the user story related to online quality visualisation (#63) covered by the other the other two (#56 and #85) for the quality visualisation. Because Q-Rapids Dashboard is a web application component, we are covering the "online" property.

Table 4 includes the progress on the features operationalising the user story related to the red/yellow/green colours.

Table 4. Progress of User Story related to Red/Yellow/Green (#83)

Feature ID	Description	Progress
#199	Add target value line (blue) to the historical data charts	Closed
#200	Add lower (red) and upper (green) threshold lines to the historical data charts	Closed
#204	Adding colour (red/green) to the Strategic Indicators View (table)	Closed
#231	Using red/yellow/green colours in the gauge charts	Closed

4.3 Development Process

Before starting the development, we run workshops with all the use cases to present them the mock-ups we envisage for the visualization of the quality model. These mock-ups included the four data views and two view modes. The feedback was positive, so we used these mock-ups as the basis for the development. More sophisticated visualizations will be considered for future versions. From these workshops we included some feature to be considered for the future versions. For the proof-of-concept, we included the use of "breadcrumbs" for navigation, when the user examines one element from one view (e.g. in the quality factors data view), in the resulting view (e.g. metrics data view), the user can see the path of the current data (Strategic Indicators > Factor).

Development started on mid July 2017, we had 3 development iterations, and 2-week internal releases. The development team had got weekly meetings, every Monday. Table 5 includes the roadmap of the releases shared with the consortium.

Table 5 Q-Rapids Dashboard Roadmap

Release	Description	Purpose
REL17.09.15	First implementation	Show initial results to potential users
REL17.10.15	Check point	Show results to use case partner representatives
REL17.11.15	PoC first deployment	Deployment purposes, the tool needed to be installed at use case premises

REL17.12.15	PoC evaluation	Version to be evaluated as proof-of-concept at M15

During this deliverable writing process (January 2017), we are in a refactoring iteration planned for end of January (REL18.01.31). In this internal release, we are also integrating the user's management implemented in parallel of the functionality to be evaluated at M15 (January 2018). Annex C includes the detailed roadmap followed for the component development.

During the proof-of-concept, we have kept updated the source code in our GitLab repository[2]. The source code is available to some of the stakeholders in the Q-Rapids project (i.e., Q-Rapids developers from several partners). In GitLab, different tags with the modules' versions together with release notes are available. Besides, we used the e-mail list mq-rapids-pilots-technical@essi.upc.edu to communicate the releases of the dashbaord together with release notes. In this e-mail list, there are responsible people for the deployment of the Q-Rapids tool in all partners.

4.4 Evaluation of Technical Aspects

The strategic indicators and the dashboard for the proof-of-concept has been evaluated in the use cases jointly to the quality model. The details about general quality of the tool and the reliability of the results is reported in the parallel deliverable D5.3.

For the technical issues that appear during the development and installation of this component, it is worth to mention that one security issue raised in one of the use cases. Qma-api-elastic API uses a java API provided by Elasticsearch engine that can come with some security issues for specific security configurations. We faced this security issue temporally with extra configuration in the server where the dashboard was installed. We will solve this issue in future versions using the REST API, already provided by Elasticsearch engine, or adding some specific configuration to the current API. We also have some problems of accessing the web application using Windows Internet Explorer browser.

[2] Q-Rapids GitLab repository: http://193.142.112.102/

5 Demonstration of Q-Rapids Dashboard

This section exposes the features included in the dashboard through screenshots. Figure 4 shows the dashboard layout, we can identify three important areas:

- *Data views menu*: menu options to select what kind of information should be shown in the assessment area. Strategic indicators, Detailed Strategic indicators, Quality Factors and Metrics.
- *View mode menu:* menu options to select how the data should be shown, the options allows graphical and textual visualization (two options on the left) of the current status or the historical data (two options on the right).
- *Assessment area: area where the assessment is visualized.*

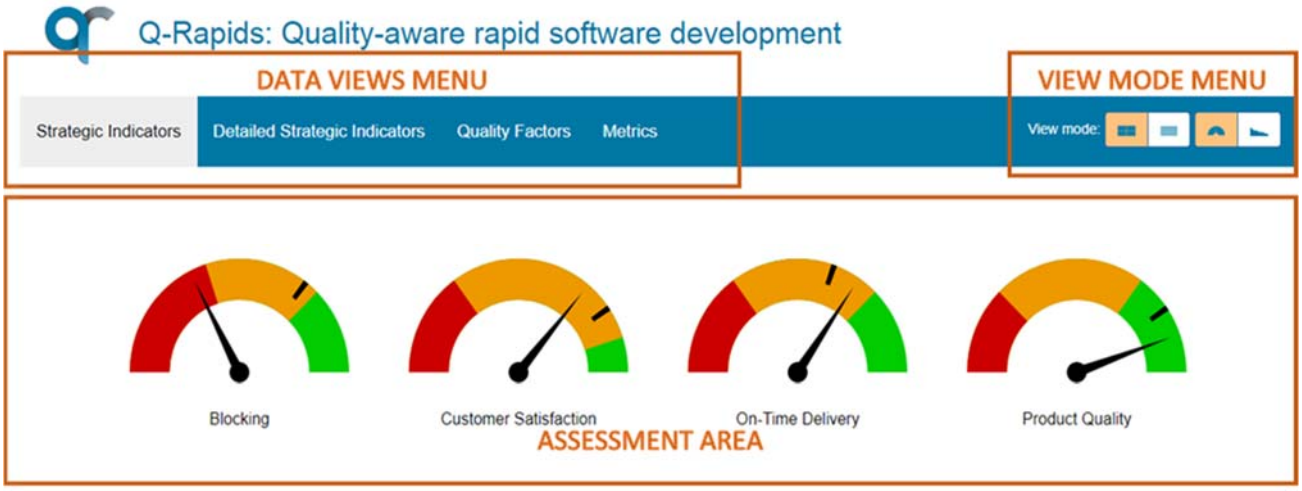


Figure 4. Q-Rapids Dashboard layout

Following subsections will include some details about the features related to the current status visualization (#56 and #63 user stories), the project evolution visualisation (#85 – trend analysis), and navigation through the different elements.

Annex D includes the Q-Rapids Dashboard Quick Guide including a short explanation for all the options accessible in the different areas of the tool. The Quick Guide and the User's Guide including all the details about the complete functionality are public at the project web site, download section (http://q-rapids.eu/). In the same page, there is also available a video tutorial.

5.1 Demonstration of the Role-dependent Views and Project Current Status User Stories

In order to include role-dependents views, the systems includes 4 different data views. These views allow the user inspect different level of abstraction that could be more adequate to his/her role. In order to inspect the current project status the dashboard provide the "*Current*" mode view all the data views. There is two different visualizations:

- Using gauge charts: showing the current value, when we only have one number to show. These charts are used in the *Strategic Indicators Data View* (Figure 5)
- Using radar charts: showing the current value for all the elements used to calculate the assessed element. These charts are used in the *Detailed Strategic Indicators Data View* (Figure 6), showing

the assessed value for the factors used to calculate the SI, and in the *Quality Factors Data View*, showing the assessed value for the metrics used to calculate the factor.

Figure 5. Strategic Indicators Data View – Current & Graphical (gauge charts)

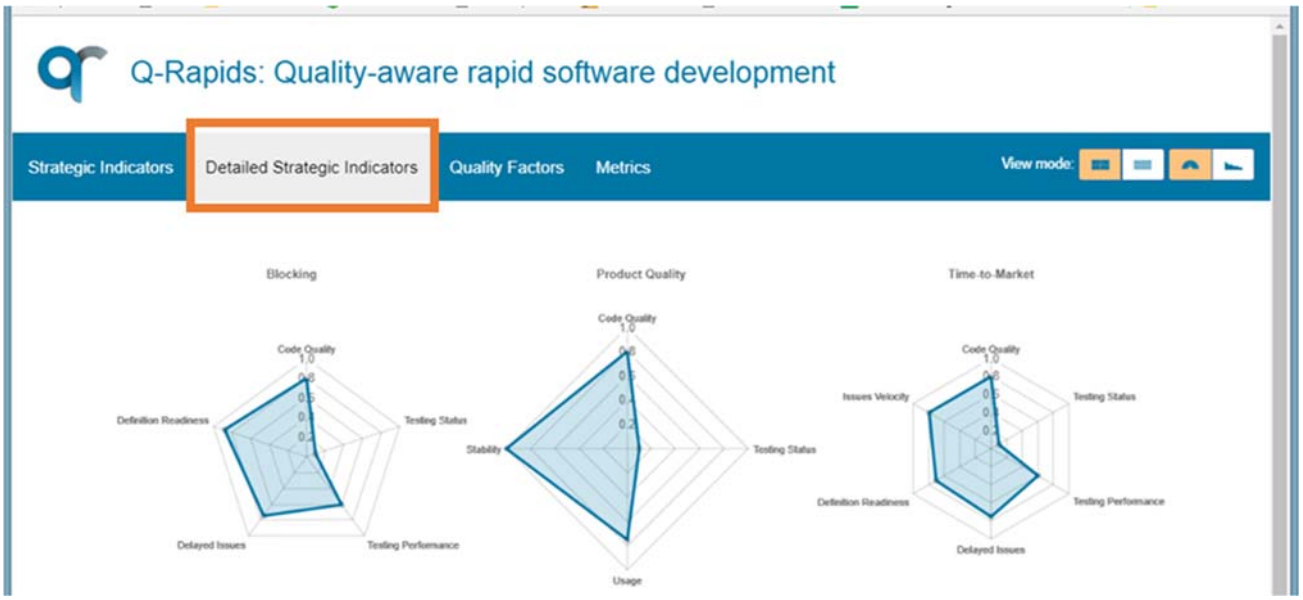


Figure 6. Detailed Strategic Indicators Data View – Current & Graphical (radar charts)

For the data views, the "*Textua*l" option in the view mode menu allows the user to visualise the information in a textual way. Figure 7 shows the textual mode view for *Metrics Data View*.

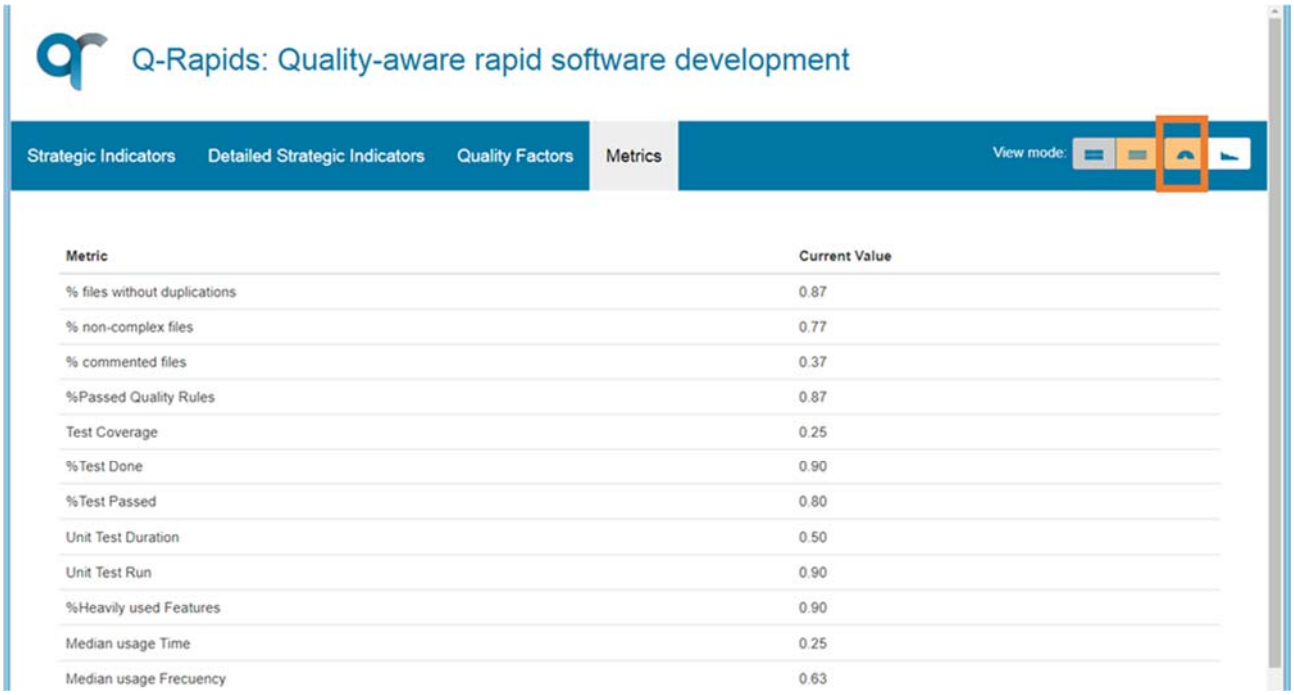


Figure 7. Metrics Data View – Current & Textual (Table)

There is no view for showing graphically the current assessed value for metrics. The current value for metrics can be only visualised textually.

5.2 Demonstration of the Role-dependent Views and Navigation User Stories

The navigation through the elements helps the user to understand the assessed values. Helping also to understand the meaning of the different elements. The user can navigate from high level elements (Strategic Indicators) to low level elements (Metrics) clicking on the element name.

For example, if the user clicks in the name of strategic indicator named *Product Quality* in the *Strategic Indicators Data View*, the system will move to the *Detailed Strategic Indicators Data View* showing only the strategic indicator Product Quality (Figure 8). If the user clicks again in the indicator's name, the system will move to the *Quality Factors Data View* showing only the factors related to this indicator (*Product Quality*). Finally, in the *Quality Factors Data View* the user can click in one of the factor to see the metric used to calculate this factor.

Figure 8. Detailed Strategic Indicator for Product Quality Strategic Indicator

When the user is navigating through the quality model elements, the name of the element is used for forward navigation (Strategic Indicator → Detailed Strategic Indicators → Quality Factors → Metrics), and the system includes the path that the user followed at the top of the assessment area (see orange square in Figure 9). The links in the navigation path can be used to navigate backwards to the origin elements. For example, Figure 9 shows the metrics related to the *Testing Status* quality factor, the user arrived to the Testing Status inspecting the strategic indicator *Product Quality*. At this point, the user could return to the *Quality Factors Data View* (viewing the Testing Status metrics) or to the *Strategic Indicators Data View*.

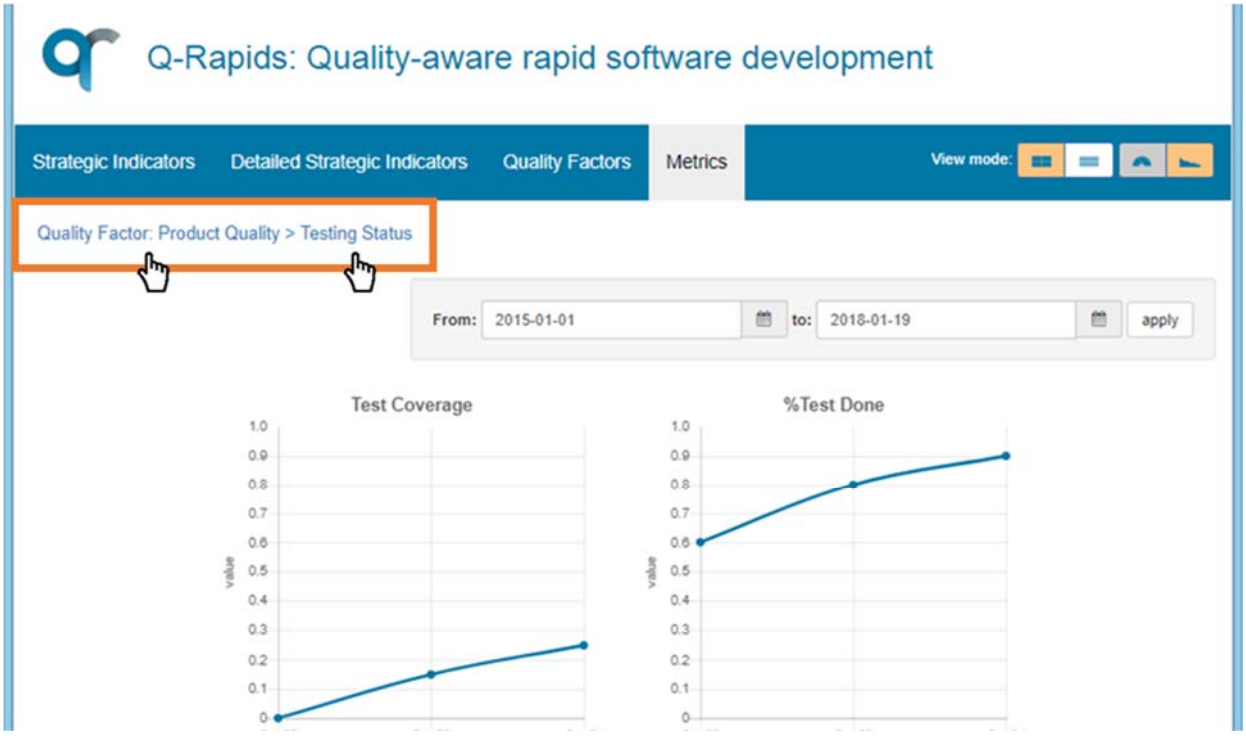


Figure 9. Detailed Strategic Indicator for Testing Status Quality Factor

5.3 Demonstration of the Trend Analysis User Stories

In order to visualise the data in order to do trend analysis, the dashboard provides the view mode option "*Historical Data*" (last option in the *view mode* menu). There is two different kind of visualizations:

- Lines charts including one series that shows the evolution of the assessed value, when we only have one number to show. These charts are used in the *Strategic Indicators Data View* (Figure 10) and Metrics Data View.
- Lines charts including several series that show the evolution of the assessed value for all the elements used to calculate the assessed element. These charts are used in the *Detailed Strategic Indicators Data View*, showing the assessed value for the factors used to calculate the strategic indicators, and in the *Quality Factors Data View*, showing the assessed value for the metrics used to calculate the factor (Figure 11).

In the historical data view, the user can select the inspected period using the date filters at the top of the assessment area. In the case of the Strategic Indicators View (Figure 10), the chart includes the target value (blue horizontal line) and the thresholds (green and red horizontal lines).

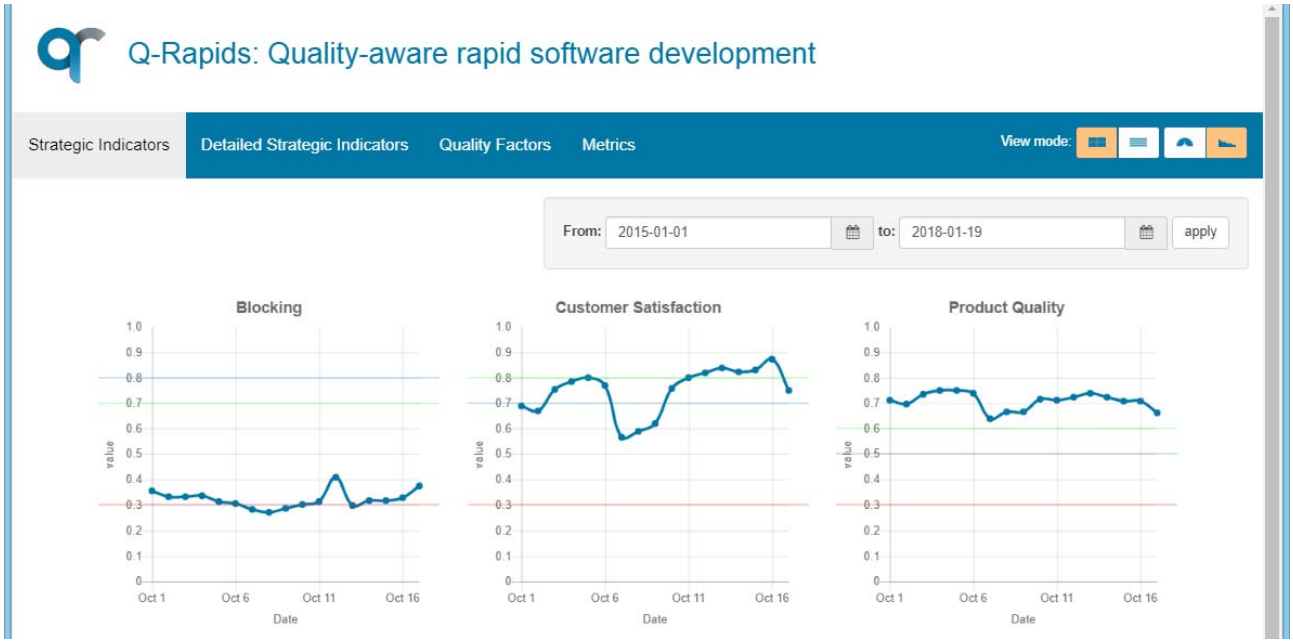


Figure 10. Strategic Indicators Data View - Historical & Graphical

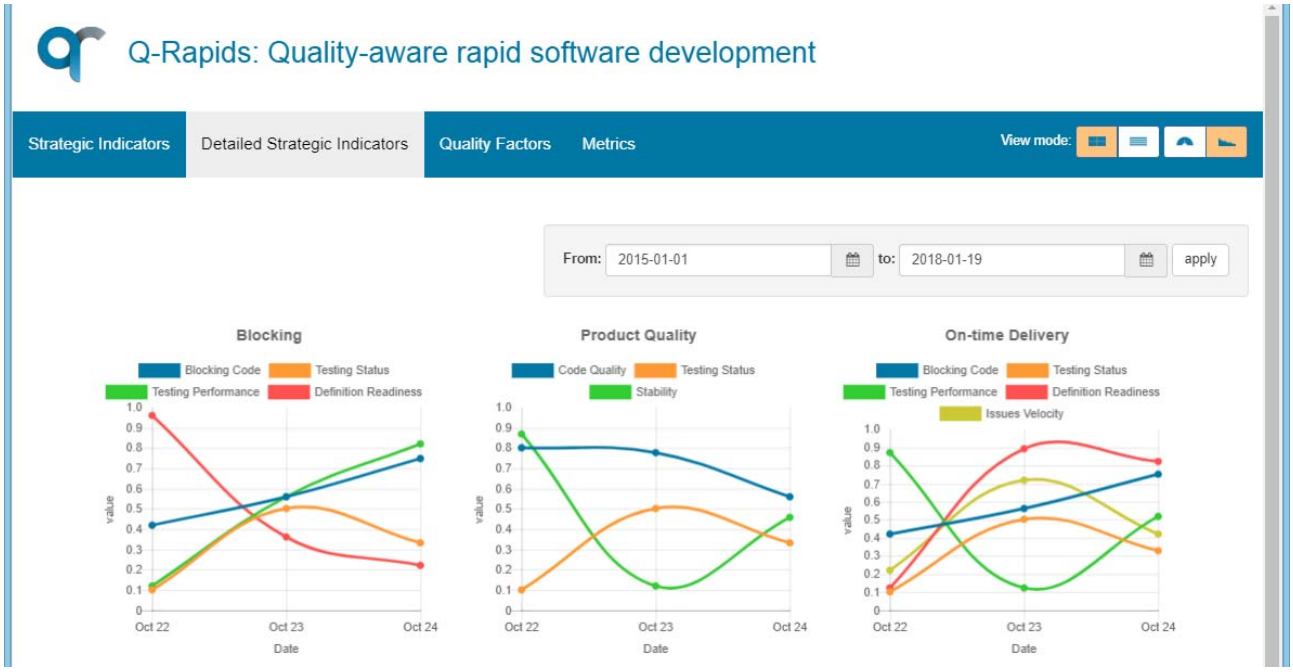
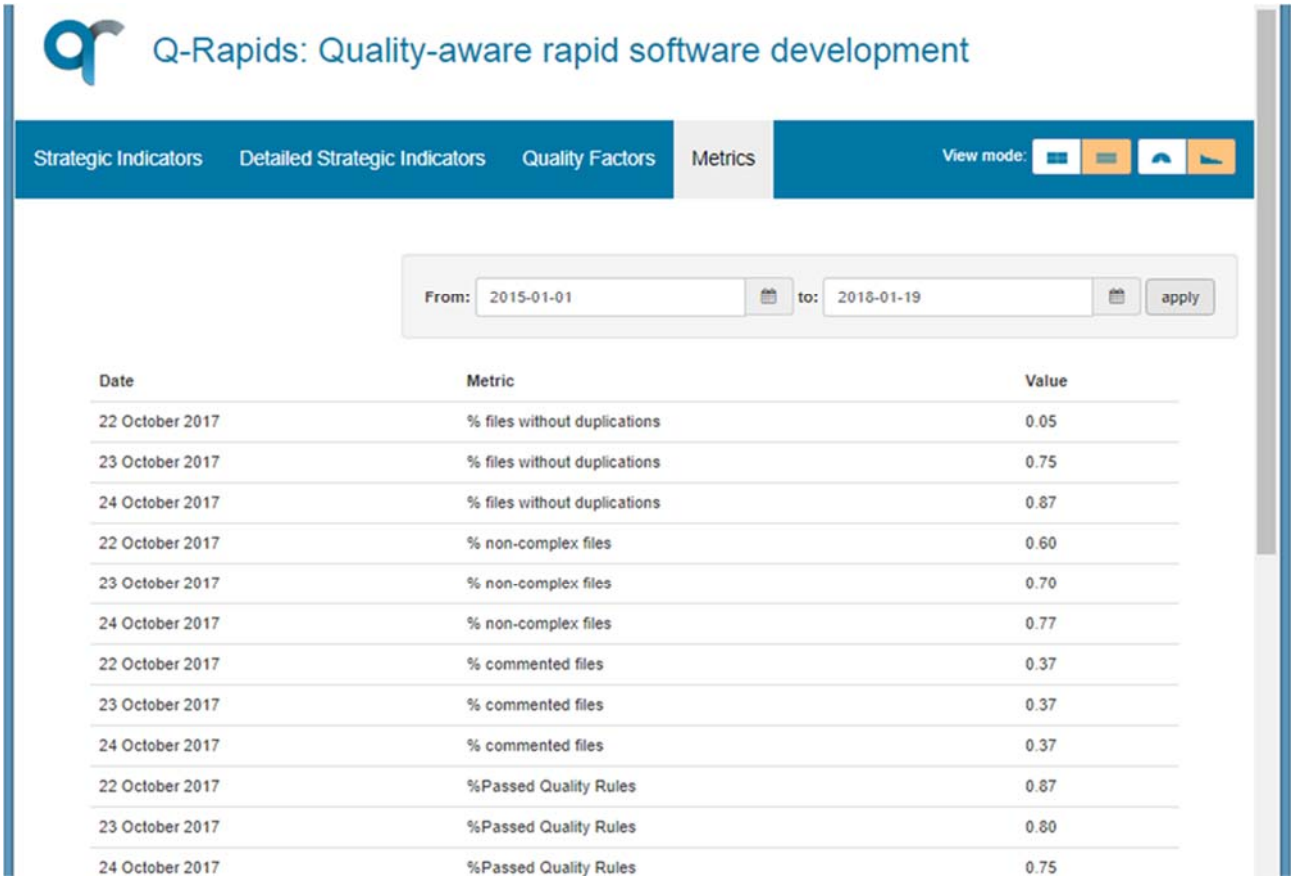


Figure 11. Detailed Strategic Indicators Data View - Historical & Graphical

The textual option shows the data in a table, where each row corresponds to the assessment of the element for a specific day (Figure 12).



Date	Metric	Value
22 October 2017	% files without duplications	0.05
23 October 2017	% files without duplications	0.75
24 October 2017	% files without duplications	0.87
22 October 2017	% non-complex files	0.60
23 October 2017	% non-complex files	0.70
24 October 2017	% non-complex files	0.77
22 October 2017	% commented files	0.37
23 October 2017	% commented files	0.37
24 October 2017	% commented files	0.37
22 October 2017	%Passed Quality Rules	0.87
23 October 2017	%Passed Quality Rules	0.80
24 October 2017	%Passed Quality Rules	0.75

Figure 12. Metrics data View - Historical & Textual

6 Q-Rapids Dashboard Software and Installation

6.1 Q-Rapids Dashboard Software

6.1.1 Technical Specification

Q-Rapids Dashboard package include two software components (qr-dashboard and qr-qma-elastic), both software components has been developed using the programming language Java. The specific details for each component are described in the following tables.

Table 6. qrapids-dashboard technical specification

Type of component	Web Application
Build	.war
Programming language	Java
DBMS	PostgreSQL
Frameworks	Spring Boot, AngularJS, Graddle
External libraries	Chart.js, Elasticsearch java API

Table 7. qrapids-qma-api-elastic tectnical specification

Type of component	Library
Build	.jar
Programming language	Java
Frameworks	Maven
External libraries	Elasticsearch java API

6.1.2 Q-Rapids Dashboard Component

According to the technical specification detailed in Table 6, the needed tools are needed:

- Java Development Kid (JDK)
- A Java IDE, e.g. Eclipse or InteliJ
- The Gradle build tool

For the configuration of the development environment, the developer needs to follow the next steps:

- *Step1*: Copy the source code in a folder, e.g `qrapids-dashboard`
- *Step 2*: Import the Gradle project by selecting the project file in the corresponding folder. Gradle will download and build all the needed dependencies (this process can take some minutes).
- *Step 3*: Generate the .war file (Gradle project window/Tasks/build/war). The .war file will be stored in the project folder, subfolder "\build\libs"

Note: the name and version of the generated file must be declared in the build.gradle configuration file:

```
war {
      baseName = 'QRapids'
      version =  '0.0.1'
}
```

The code has been structured following the packages defined in the architecture (see Section 3):

- front-end: `qrapids-dashboard\scr\main\java\com\upc\resources`
- mapping from the application URLs to the corresponding view (html file): `qrapids-dashboard\scr\main\java\com\upc\mapping`
- back-end: `qrapids-dashboard\scr\main\java\com\upc\app`

Front-end package is composed by two packages:

- `resources\templates`: html files defining the views, these views execute java script to get the data from the presentation controllers. The views are grouped by kind of information to show into four packages: `Strategic Indicators`, `Detailed Strategic Indicators`, `Quality Factors`, and `Metrics`.
- `resources\static`: The presentation controllers that call the RestServices from the back-end. The controllers are the java script files that are invoked in the html files.

Back-end package is composed by three packages:

- `app\domain`: contains the `services` package including the REST services to be invoked from the front-end on an external system, and the `adapters` package to connect to the data layer (`adapters\database`) and the external services (`adapters\QMA`).
- `app\data`: contains the classes to access to the internal database, these classes must implement some interface from the `domain\adapters\database` package.
- `app\dto`: contains the data transfer objects used by the different classes in the different packages to interchange information.
- app\config: contains the classes responsible of the connection configuration to the internal database and the elastic search engine.

Annex E details the qr-dashboard component release notes.

6.1.3 QMA-API-Elastic Component

According to the technical specification detailed in Table 7, the needed tools are needed:

- The Java Development Kit (JDK)
- A Java IDE, e.g. Eclipse, IntelliJ
- The Maven build tool

Maven will identify/download the Elastic search libraries for the connection to the Elastic Search Engine. The dependencies are defined in the `pom.xml` configuration file.

For the configuration of the development environment, the developer needs to follow the next steps:

- *Step 1:* Copy the source code in a folder: qrapids-qma-elastic
- *Step 2:* Import the Maven project by selecting the project file in the corresponding folder
- *Step 3:* Compile the project (Maven Project window/Livecycle/compile).
- *Step 4:* Generate the jar file (Maven Project window/Plugins/jar/jar:jar). The generated file will be located in the project folder, subfolder "target".

The .jar file can be generated including or not all the dependencies. If the jar file is generated without dependencies, the project that imports this jar will need to include explicitly the library dependencies (the developer should check the tag <dependencies> in project file `pom.xml`).

The developer can choose between both options including some parameters in `pom.xml` file. For the generation of the jar with all the dependencies, the `pom.xml` should include the following:

```
<plugin>
   <groupId>org.apache.maven.plugins</groupId>
   <artifactId>maven-shade-plugin</artifactId>
   <version>3.1.0</version>
   <executions>
      <execution>
         <phase>package</phase>
         <goals><goal>shade</goal></goals>
         <configuration>
           <transformers>
              <transformer
implementation="org.apache.maven.plugins.shade.resource.ServicesResourceTransfor
mer"/>
           </transformers>
         </configuration>
      </execution>
   </executions>
</plugin>
```

Note: The name and version for the .jar file should be specified in the pom.xml configuration file before step 3.

The code has been structured as:

- `java\evaluation`: contains the DTOs used for the data transfer.
- `java\qrapids.qm`: contains the four classes to be used by the external systems. Connection package for the elastic search connection configuration and Factor, Metric, and StrategicIndicators packages for accessing to the evaluation of these kind of elements.
- `java\util`: contains some classes that allow parametrization and code reusability.

Annex F details the qr-qma-elastic component release notes.

6.2 Q-Rapids Dashboard Installation

The installation process is quite different if it is a version upgrade or an initial installation. For the upgrades, the system administrator needs to unpack the war file included in the installation package and deployed in the web server. For the initial installation the installation process is a bit more complicated, the system manager needs to follow the next steps:

- *Step 1:* Deploy war file from the installation package in the web server
- *Step 2:* Create the local database, adding manually some records
- *Step 3:* Configure the database and elastic search engine connections

Detailed instructions for the installation are included in Annex G. Q-Rapids Dashboard installation is part of the Q-Rapid Tool installation, concretely they are included as step 5:

- *Step 1:* Download and unzip the installation package files
- *Step 2:* Follow the instructions in the document "Connector - Deployment_v0.1.docx" from the data gathering package. You will need the files included in "qr-connect.zip"
- *Step 3*: Configure the Elasticsearch indexes to compute the quality model. You will find the instructions in the file "poc.indexes.txt"

- *Step 4*: Follow the instructions of the "README.txt" of the qr-eval folder in the data gathering package.
- *Step 5*: Install and configure Q-Rapids Dashboard component. You will find the instructions in the document "Dashboard - Deployment.pdf". You will need the the files included in WP3.zip (QRapids-0.1.0.war, dashboardDB.sql, dashboardDB_tables.sql, and dashboardDB_PoC.sql).

One of the requisites for Q-Rapids Dashboard installation is to have the data gathering and analysis software modules installed, specific instructions for their installation can be found in deliverable D1.2. Additionally, the technical requisites to deploy Q-Rapids Dashboard are: a web server, e.g. Tomcat, Oracle Java JRE, and PostgreSQL DBSM.

6.3 Download

The qrapids-dashboard installation package and source code are provided as a zip archive files in public links in the Q-Rapids basecamp (internal project management tool):

- qr-dashboard installation package:
  https://public.3.basecamp.com/p/b5z9DoqztPEN6ZUVLB9ATVBh
- qr-dashboard and qr-qma-api-elastic components source code:
  https://public.3.basecamp.com/p/XRZ7B9vBxsA5iKvPD6TrF4oW

Conclusion

To demonstrate the feasibility of the solution proposed by Q-Rapids project, this deliverable describes Q-Rapids Dashboard software component, in terms of functionality and the role as part of the Q-Rapids Tool. It also includes a report of the current status of the tasks from work package 3 (Strategic Decision Making Dashboard) that the results are reflected on the tool.

The main goal of the proof-of-concept, reported in this deliverable, was to demonstrate that the solution proposed by the project allows the assessment of strategic indicators using real data of the use cases, and that the project can provide a strategic dashboard tool to visualize them.

Finally, the deliverable contains some technical documentation related to the software component and its download and installation.

Annex A. Quality Model Assessment (QMA) API Documentation

The Q-Rapids quality model is characterised by three types of entities: Strategic Indicators, Factors and Metrics. This library) includes three classes to access to these types of entities.

Following tables contain the method summary for these classes and the DTOs used to transfer the data.

Table 8. class StrategicIndicator (QMA API)

Method and result	Description
Current Status (only one evaluation per strategic indicator)	
getEvaluations() returns StrategicIndicatorEvaluationDTO []	This method returns the list of the strategic indicators and the "last" evaluation. The evaluation contains the evaluation date and value
getFactorsEvaluations() returns StrategicIndicatorFactorEvaluationDTO []	This method returns the list of the strategic indicators. For each strategic indicator, it returns the list of factors associated to it and their "last" evaluation. The evaluation contains the evaluation date and value.
getFactorsEvaluations(id: string) returns FactorEvaluationDTO []	This method returns the list of factors associated to the *strategic indicator evaluation* passed as a parameter and their "last" evaluation. The evaluation contains the evaluation date and value.
Historical Data (more than one evaluation per strategic indicator)	
getEvaluation(Date from, Date to) returns StrategicIndicatorEvaluationDTO []	This method returns the list of the strategic indicators and the evaluations belonging to the specific period defined by the parameters from and to. The evaluation contains the evaluation date and value.
getFactorsEvaluations(Date from, Date to) returns StrategicIndicatorFactorEvaluationDTO[]	This method returns the list of the strategic indicators. For each strategic indicator, it returns the list of factors associated to it and their evaluations belonging to the period defined by the parameters *from* and *to*. The evaluation contains the evaluation date and value.
getFactorsEvaluations(id: string, Date from, Date to) returns FactorEvaluationDTO[]	This method returns the list of factors associated to the *strategic indicator evaluation* passed as parameter and their evaluations belonging to the period defined by the parameters *from* and *to*. The evaluation contains the evaluation date and value.

Table 9 class Factor (QMA API)

Method and result	Description
Current Status (only one evaluation per factor)	
getEvaluations() returns FactorEvaluationDTO[]	This method returns the list of the factors and the "last" evaluation. The evaluation contains the evaluation date and value.
getMetricsEvaluations() returns FactorMetricEvaluationDTO []	This method returns the list of the factors, for each factor it returns the list of metrics associated to this factor and the "last" metric evaluation. The evaluation contains the evaluation date and value.
getMetricsEvaluations(id: string) returns MetricEvaluationDTO[]	This method returns the list of metrics associated to the *factor evaluation* passed as parameter and the "last" metric evaluation. The evaluation contains the evaluation date and value.
Historical Data (more than one evaluation per factor)	
getEvaluations(Date from, Date to)	This method returns the list of the factors and the evaluations belonging to a specific period defined by the parameters *from* and *to*. The evaluation contains the evaluation date and value.
getMetricsEvaluations(Date from, Date to) returns FactorEvaluationDTO []	This method returns the list of the factors, for each factor it returns the list of metrics associated to this factor. For each metric, it returns the evaluations belonging to the period defined by the parameters *from* and *to*. The evaluation contains the evaluation date and value.
getMetricsEvaluations(id: string, Date from, Date to) returns MetricEvaluationDTO[]	This method returns the list of metrics associated to the *factor evaluation* passed as parameter. For each metric, it returns the evaluations belonging to the period defined by the parameters *from* and *to*. The evaluation contains the evaluation date and value.

Table 10 class Metric (QMA API)

Method and result	Description
getEvaluations() returns MetricEvaluationDTO []	This method returns the list of the metrics and their "last" evaluation. The evaluation contains the evaluation date and value.
getEvaluations(Date from, Date to) returns MetricEvaluationDTO []	This method returns the list of the metrics and the evaluations belonging to a specific period defined by the parameters *from* and *to*. The evaluation contains the evaluation date and value.

Following tables include the data included in the DTOs used to return the results.

Table 11 EvaluationDTO (QMA API)

Attribute	Type
Id	Integer
date	Date
value	Float
source	String

Table 12 StrategicIndicatorEvaluationDTO (QMA API)

Attribute	Type
strategicIndicatorID	String
strategicIndicatorName	String
strategicIndicatorDescription	String
evaluations	EvaluationDTO[]

Table 13 FactorEvaluationDTO (QMA API)

Attribute	Type
factorID	String
factorName	String
factorDescription	String
evaluations	EvaluationDTO[]

Table 14 MetricEvaluationDTO (QMA API)

Attribute	Type
metricID	String
metricName	String
metricDescription	String
evaluations	EvaluationDTO[]

Table 15 StrategicIndicatorFactorEvaluationDTO (QMA API)

Attribute	Type
strategicIndicatorID	String
strategicIndicatorName	String
factors	FactorEvaluationDTO[]

Table 16 FactorMetricEvaluationDTO (QMA API)

Attribute	Type
factorID	String
factorName	String
metrics	MetricEvaluationDTO[]

Annex B. Strategic Decision Making (SDM) REST API Documentation

This annex contains the documentation about the RestServices provided by the dashboard.

Detailed Strategic Indicators, Factors and Metrics can also be called including an identifier (id) in the URI, which will filter the results only returning the corresponding Detailed Strategic Indicator identifier, the ones matching the identifier of the parent Strategic Indicator (for Factors) or parent Factor (for Metrics).

Following tables contain the service interfaces.

Table 17 Get Current Strategic Indicators Evaluations (SDM REST API)

Title	**Get Current Strategic Indicators Evaluations**
URI	… /api/StrategicIndicators/CurrentEvaluation
Method	GET
URI Params	None

Success Response	200 ok
	[{
	"id": "productquality",
	"name": "Product Quality",
	"target": 0.2,
	"lowerThreshold": 0.1,
	"upperThreshold": 0.7,
	"evaluations": [
	{
	"value": 0.76090264,
	"date": {
	"year": 2017,
	"month": "DECEMBER",
	"dayOfYear": 349,
	"leapYear": false,
	"dayOfMonth": 15,
	"dayOfWeek": "FRIDAY",
	"era": "CE",
	"chronology": {
	"id": "ISO",
	"calendarType": "iso8601"
	},
	"monthValue": 12
	},
	"datasource": "http://www.example.com"
	}
	]
	}]

Error Response	404 Not Found

Table 18 Get Historic Strategic Indicators Evaluations (SDM REST API)

Title	**Get Historic Strategic Indicators Evaluations**			
URI	…/api/StrategicIndicators/HistoricalData			
Method	GET			
URI Params	**Attribute**	**Type**	**Required**	**Description**
	from	Date	yes	From date (yyyy-mm-dd) for the requested the period
	to	Date	yes	To date (yyyy-mm-dd) for the requested period

Success Response	200 ok
	[{
	"id": "productquality",
	"name": "Product Quality",
	"target": 0.2,
	"lowerThreshold": 0.1,
	"upperThreshold": 0.7,
	"evaluations": [
	{
	"value": 0.94,
	"date": {
	"year": 2017,
	"month": "NOVEMBER",
	"dayOfYear": 318,
	"leapYear": false,
	"dayOfMonth": 14,
	"dayOfWeek": "TUESDAY",
	"era": "CE",
	"chronology": {
	"id": "ISO",
	"calendarType": "iso8601"
	},
	"monthValue": 11
	},
	"datasource": "http://www.example.com"
	},
	{
	"value": 0.76090264,
	"date": {
	"year": 2017,
	"month": "NOVEMBER",
	"dayOfYear": 326,
	"leapYear": false,
	"dayOfMonth": 22,
	"dayOfWeek": "WEDNESDAY",
	"era": "CE",
	"chronology": {
	"id": "ISO",
	"calendarType": "iso8601"
	},
	"monthValue": 11
	},
	"datasource": "http://www.example.com"
	}
	]
	}			

Error Response	404 Not Found			

Table 19 Get Current Detailed Strategic Indicators Evaluations (SDM REST API)

Title	**Get Current Detailed Strategic Indicators Evaluations**
URI	…/api/DetailedStrategicIndicators/CurrentEvaluation …/api/DetailedStrategicIndicators/CurrentEvaluation/<id>
Method	GET
URI Params	None

Success Response	200 ok
	[{
	"id": "blocking",
	"name": "Blocking",
	"factors": [
	{
	"id": "blockingcode",
	"name": "Blocking Code",
	"evaluations": [
	{
	"value": 0.6692857,
	"date": {
	"year": 2017,
	"month": "DECEMBER",
	"dayOfYear": 349,
	"leapYear": false,
	"dayOfMonth": 15,
	"dayOfWeek": "FRIDAY",
	"era": "CE",
	"chronology": {
	"id": "ISO",
	"calendarType": "iso8601"
	},
	"monthValue": 12
	},
	"datasource": "http://www.example.com"
	}]
	},
	{
	"id": "qualityissuespecification",
	"name": "Quality Issue Specification",
	"evaluations": [
	{
	"value": 0,
	"date": {
	"year": 2017,
	"month": "DECEMBER",
	"dayOfYear": 349,
	"leapYear": false,
	"dayOfMonth": 15,
	"dayOfWeek": "FRIDAY",
	"era": "CE",
	"chronology": {
	"id": "ISO",
	"calendarType": "iso8601"
	},
	"monthValue": 12
	},
	"datasource": "http://www.example.com"
	}]
	}]
	}]

Error Response	404 Not Found

Table 20 Get Historical Detailed Strategic Indicators Evaluations (SDM REST API)

Title	**Get Historical Detailed Strategic Indicators Evaluations**			
URI	…/api/DetailedStrategicIndicators/HistoricalData …/api/DetailedStrategicIndicators/HistoricalData/<id>			
Method	GET			
URI Params	**Attribute**	**Type**	**Required**	**Description**
	from	Date	yes	From date (yyyy-mm-dd) for the requested the period
	to	Date	yes	To date (yyyy-mm-dd) for the requested period

Success Response	200 ok
	[{
	"id": "blocking",
	"name": "Blocking",
	"factors": [{
	"id": "blockingcode",
	"name": "Blocking Code",
	"evaluations": [{
	"value": 0.6692857,
	"date": {
	"year": 2017,
	"month": "NOVEMBER",
	"dayOfYear": 331,
	"leapYear": false,
	"dayOfMonth": 27,
	"dayOfWeek": "MONDAY",
	"era": "CE",
	"monthValue": 11,
	"chronology": {
	"id": "ISO",
	"calendarType": "iso8601"
	}
	},
	"datasource": "null"
	},{
	"value": 0.6692857,
	"date": {
	"year": 2017,
	"month": "DECEMBER",
	"dayOfYear": 338,
	"leapYear": false,
	"dayOfMonth": 4,
	"dayOfWeek": "MONDAY",
	"era": "CE",
	"monthValue": 12,
	"chronology": {
	"id": "ISO",
	"calendarType": "iso8601"
	}
	},
	"datasource": "null"
	}]
	},{
	"id": "qualityissuespecification",
	"name": "Quality Issue Specification",
	…
	}]}			

Error Response	404 Not Found			

Table 21 Get Current Quality Factors Evaluations (SDM REST API)

Title	**Get Current Quality Factors Evaluations**
URI	…/api/QualityFactors/CurrentEvaluation …/api/QualityFactors/CurrentEvaluation/<id>
Method	GET
URI Params	None

Success Response	200 ok
	[
	{
	"id": "blockingcode",
	"name": "Blocking Code",
	"metrics": [
	{
	"id": "nonblockingfiles",
	"name": "Non blocking files",
	"evaluations": [
	{
	"value": 0.6692857,
	"date": {
	"year": 2017,
	"month": "DECEMBER",
	"dayOfYear": 355,
	"leapYear": false,
	"dayOfMonth": 21,
	"dayOfWeek": "THURSDAY",
	"era": "CE",
	"chronology": {
	"id": "ISO",
	"calendarType": "iso8601"
	},
	"monthValue": 12
	},
	"datasource": "http://www.example.com"
	}]
	}]
	}
	]

Error Response	404 Not Found

Table 22 Get Historical Quality Factors Evaluations (SDM REST API)

Title	**Get Historical Quality Factors Evaluations**			
URI	…/api/QualityFactors/HistoricalData …/api/QualityFactors/HistoricalData/<id>			
Method	GET			
URI Params	**Attribute**	**Type**	**Required**	**Description**
	from	Date	yes	From date (yyyy-mm-dd) for the requested the period
	to	Date	yes	To date (yyyy-mm-dd) for the requested period

Success Response	200 ok
	[
	{
	"id": "blockingcode",
	"name": "Blocking Code",
	"metrics": [
	{
	"id": "nonblockingfiles",
	"name": "Non blocking files",
	"evaluations": [
	{
	"value": 0.6692857,
	"date": {
	"year": 2017,
	"month": "NOVEMBER",
	"dayOfYear": 331,
	"leapYear": false,
	"dayOfMonth": 27,
	"dayOfWeek": "MONDAY",
	"era": "CE",
	"chronology": {
	"id": "ISO",
	"calendarType": "iso8601"
	},
	"monthValue": 11
	},
	"datasource": "null"
	},{
	"value": 0.6692857,
	"date": {
	"year": 2017,
	"month": "DECEMBER",
	"dayOfYear": 338,
	"leapYear": false,
	"dayOfMonth": 4,
	"dayOfWeek": "MONDAY",
	"era": "CE",
	"chronology": {
	"id": "ISO",
	"calendarType": "iso8601"
	},
	"monthValue": 12
	},
	"datasource": "http://www.example.com"
	}]
	}]
	}
	]			

Error Response	404 Not Found			

Table 23 Get Current Quality Factors Evaluations (SDM REST API)

Title	**Get Current Quality Factors Evaluations**
URI	…/api/Metrics/CurrentEvaluation …/api/Metrics/CurrentEvaluation/<id>
Method	GET
URI Params	None

Success Response	200 ok
	[
	{
	"id": "complexity",
	"name": "Complexity",
	"evaluations": [
	{
	"value": 0.9939302,
	"date": {
	"year": 2017,
	"month": "DECEMBER",
	"dayOfYear": 355,
	"leapYear": false,
	"dayOfMonth": 21,
	"dayOfWeek": "THURSDAY",
	"era": "CE",
	"monthValue": 12,
	"chronology": {
	"id": "ISO",
	"calendarType": "iso8601"
	}
	},
	"datasource": "http://www.example.com"
	}]
	}
	]

Error Response	404 Not Found

Table 24 Get Historical Metrics Evaluations (SDM REST API)

Title	**Get Historical Metrics Evaluations**			
URI	…/api/Metrics/HistoricalData …/api/Metrics/HistoricalData/<id>			
Method	GET			
URI Params	**Attribute**	**Type**	**Required**	**Description**
	from	Date	yes	From date (yyyy-mm-dd) for the requested the period
	to	Date	yes	To date (yyyy-mm-dd) for the requested period

Success Response	200 ok
	[
	{
	"id": "complexity",
	"name": "Complexity",
	"evaluations": [
	{
	"value": 0.99,
	"date": {
	"year": 2017,
	"month": "NOVEMBER",
	"dayOfYear": 318,
	"leapYear": false,
	"dayOfMonth": 14,
	"dayOfWeek": "TUESDAY",
	"era": "CE",
	"monthValue": 11,
	"chronology": {
	"id": "ISO",
	"calendarType": "iso8601"
	}
	},
	"datasource": "http://www.example.com"
	},{
	"value": 0.9939302,
	"date": {
	"year": 2017,
	"month": "NOVEMBER",
	"dayOfYear": 324,
	"leapYear": false,
	"dayOfMonth": 20,
	"dayOfWeek": "MONDAY",
	"era": "CE",
	"monthValue": 11,
	"chronology": {
	"id": "ISO",
	"calendarType": "iso8601"
	}
	},
	"datasource": "http://www.example.com"
	}]
	}
	]			

Error Response	404 Not Found			

Annex C. Development Roadmap

Following figures show the features included in all the development releases (internal and public to the consortium). The releases public to the consortium are the ones that contains a version number.

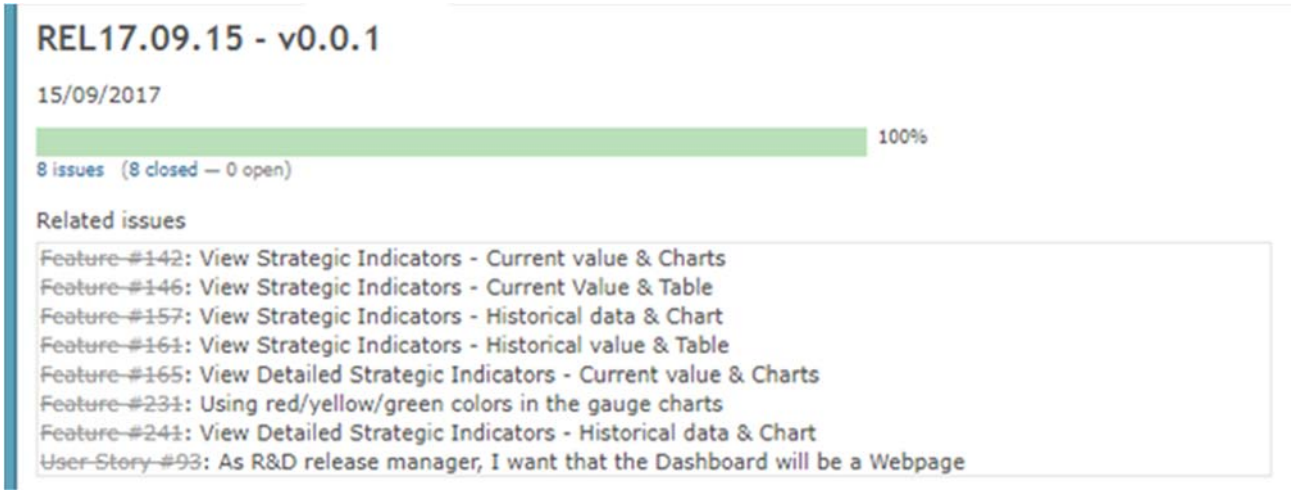


Figure 13. REL17.09.15 (v 0.0.1) - presented to potential partner users

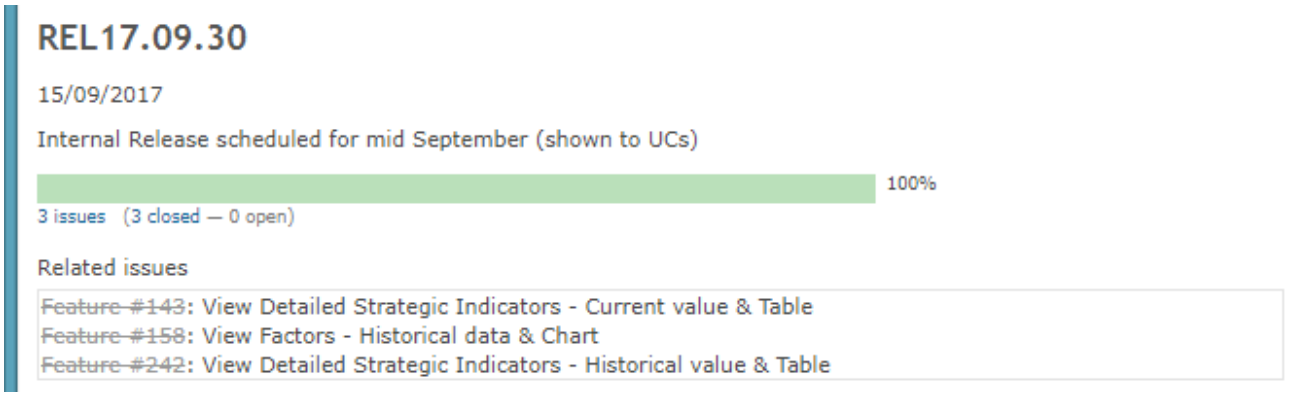


Figure 14. REL17.09.30 - internal release

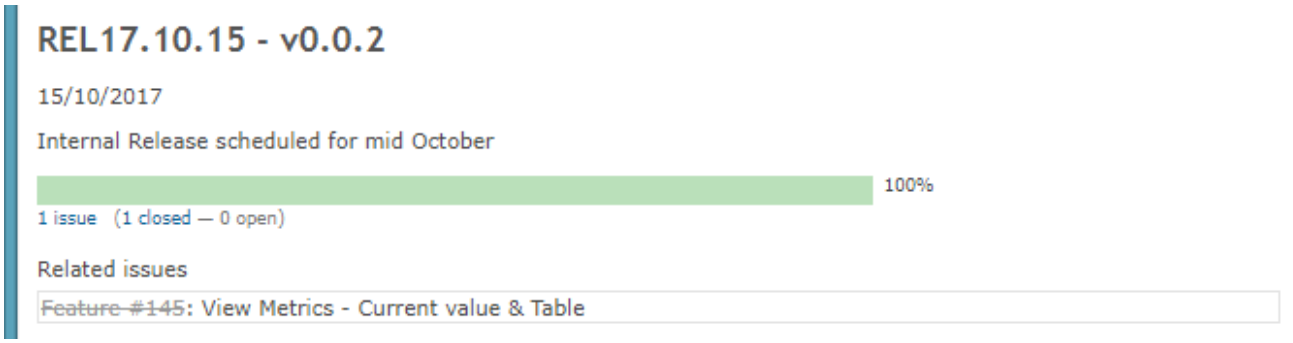


Figure 15. REL17.10.15 (v 0.0.2) - presented to partner representatives

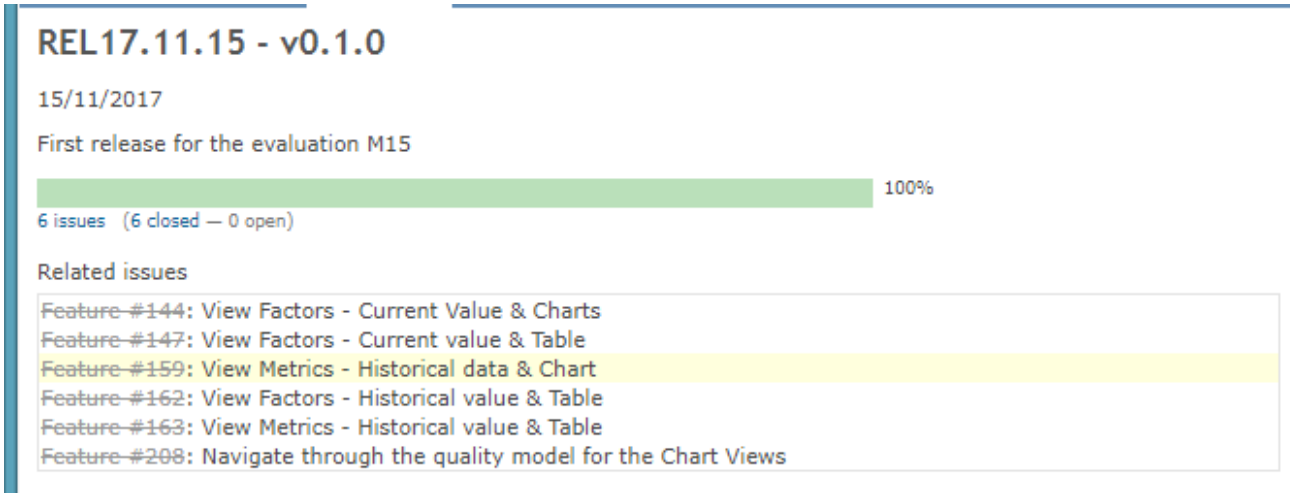


Figure 16. REL17.11.15 (v 0.1.0) - first deployment at use case premises

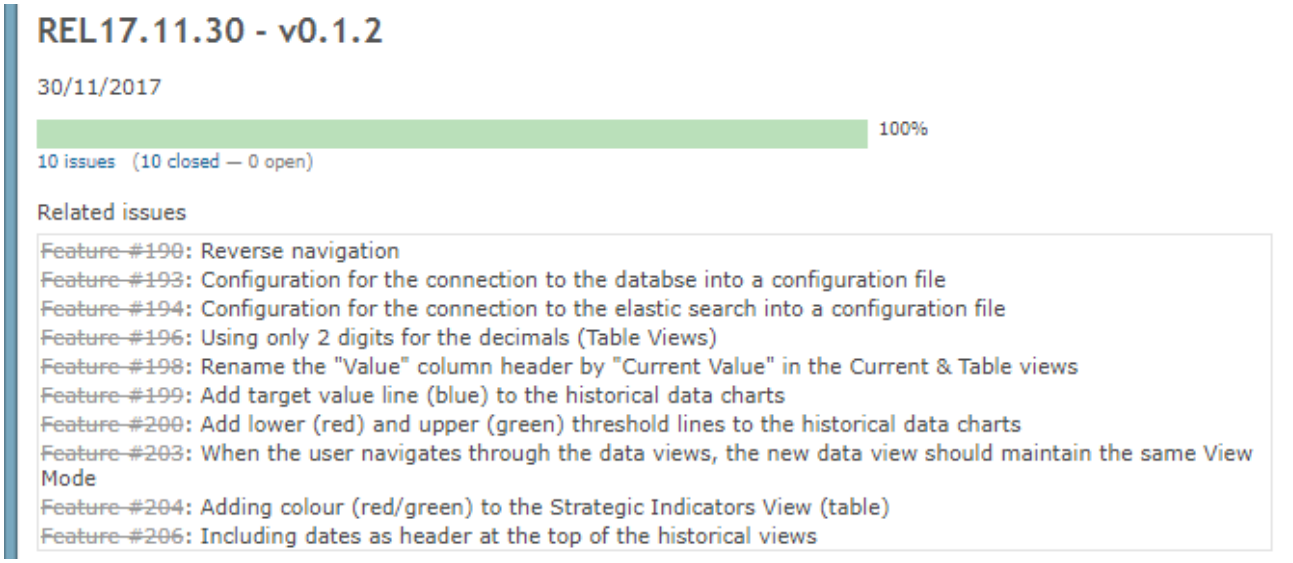


Figure 17. REL17.11.30 (v 0.1.2) - some improvements

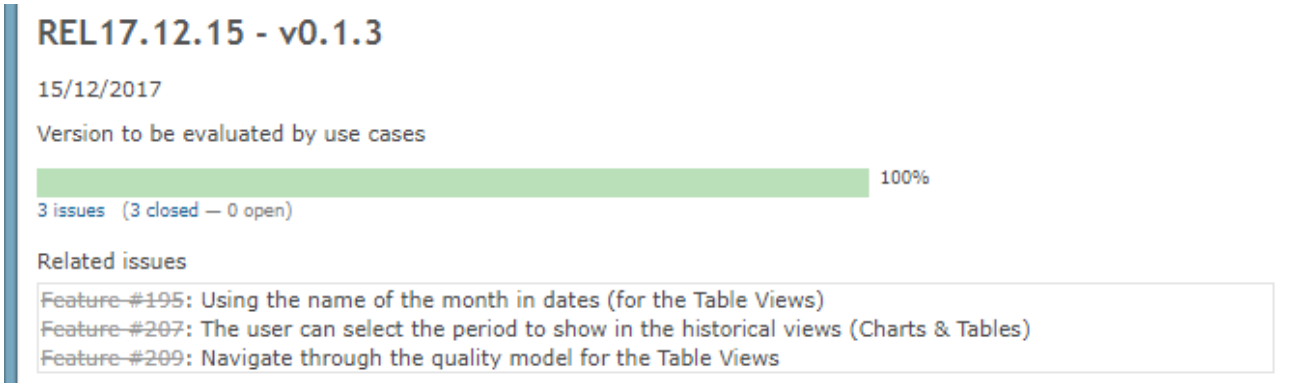


Figure 18. REL17.12.15 (v0.1.3) - proof-of-concept evaluation

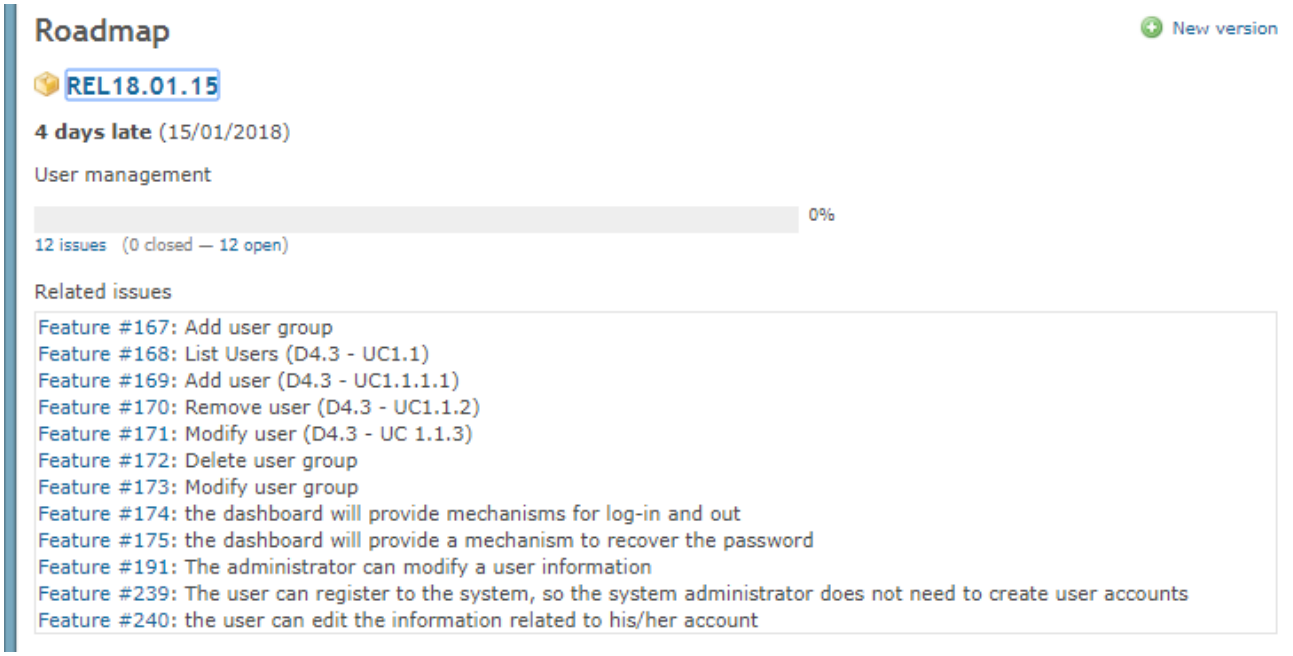


Figure 19. REL18.01.15 - internal release (user management)

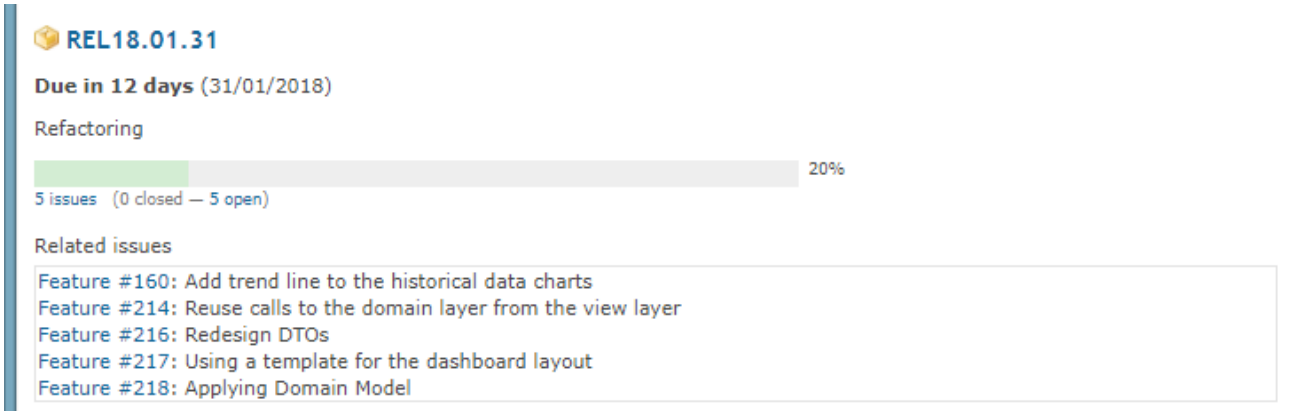
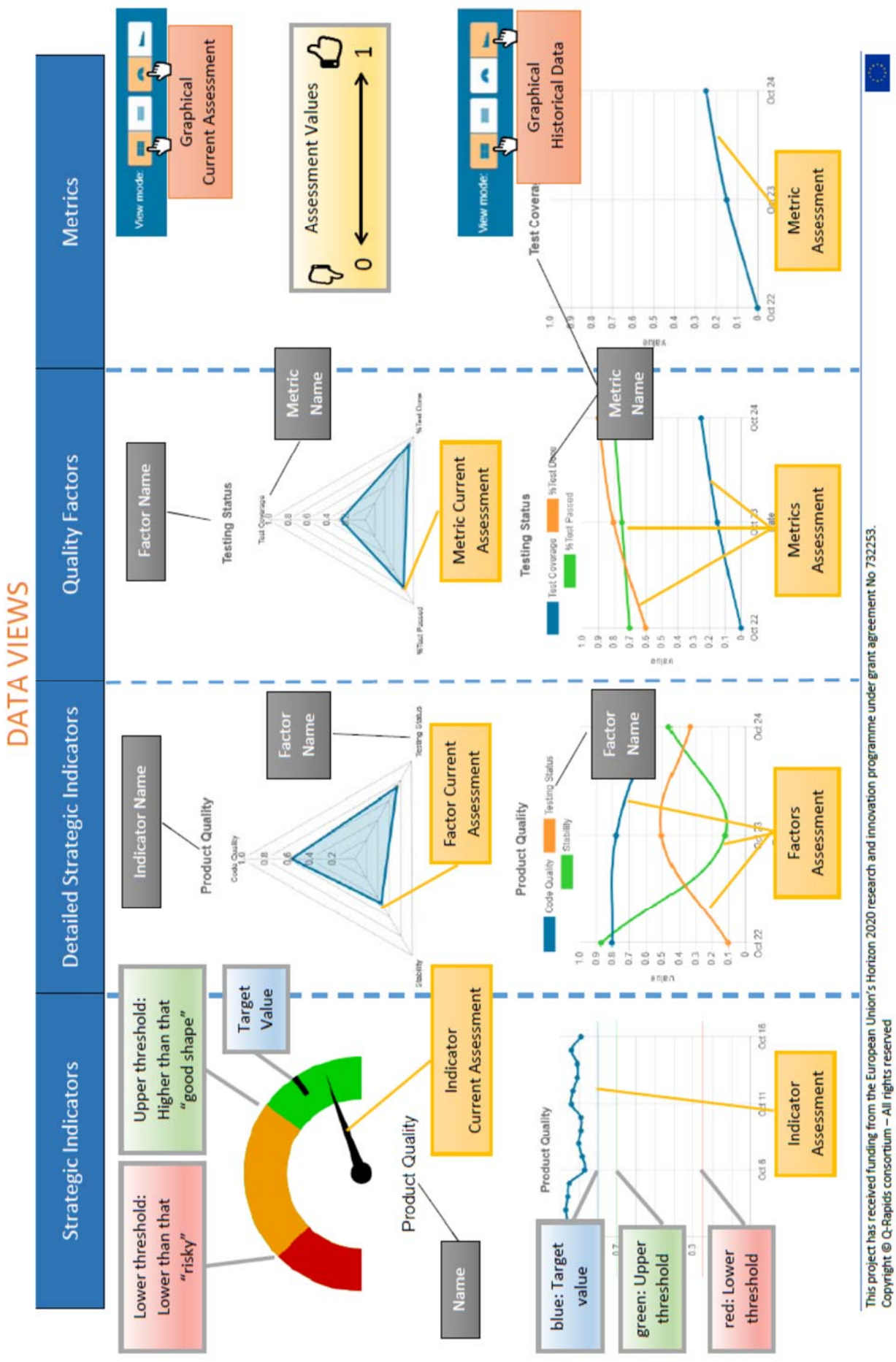


Figure 20. REL18.01.31 - internal release (refactoring)

Annex D. Q-Rapids Dashboard Quick Guide


DATA VIEWS
Strategic Indicators
Detailed Strategic Indicators
Quality Factors
Metrics
Graphical Current Assessment
Assessment Values
0
1
Graphical Historical Data
Factor Name
Testing Status
Metric Name
Metric Current Assessment
Metric Assessment
Indicator Name
Product Quality
Factor Current Assessment
Factors Assessment
Upper threshold: Higher than that "good shape"
Target Value
Lower threshold: Lower than that "risky"
Indicator Current Assessment
Indicator Assessment
Name
blue: Target value
green: Upper threshold
red: Lower threshold
This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 732253.
Copyright © Q-Rapids consortium – All rights reserved

Annex E. Q-Rapids Dashboard Release Notes

Q-Rapids v0.1.3 (December, 19th 2017)

- Including the release notes (README.md file) in the deployment
- Filter for dates period in the historical data views (only informative in previous version)
- Element navigation included for the table views (only in graphical views in previous version)
- Dates format as "Day Month Year", months in words
- Fixed a bug in the historical views for not valid evaluation values

Q-Rapids v0.1.2 (December, 5th 2017)

- Strategic Indicators (Current & Table) View:
  - The evaluation value is coloured like charts (red if value lower than lower threshold and green if higher that the upper threshold)
- Strategic Indicators (Chart & Historical) View: added the target value (blue) and the lower (red) and upper (green) thresholds
- Historical Data views: Included the from and to dates at the top (only informative fields)
- Navigation, when the user uses the menu, the View mode is kept. In previous versions, the menu options open Chart & Current
- Table views: Float numbers show 2 decimal digits in tables

Q-Rapids v0.1.1 (November, 22nd 2017)

Connection to the database and Elastic Search Engine configured in the application.properties file

Q-Rapids v0.1.0 (November, 15th 2017)

This version includes the functionality related to the visualization of the four data views of data. For the graphical views, the user can navigate forward among this views using the element names.

Annex F. QMA-API-Elastic Release Notes

qma-elastic v0.0.6 (November, 17th 2017)

Previous version where internal versions, so we include the complete list features of in this version.

This API provides three classes grouping the functionality related to the different quality model assessment elements:

- Strategic Indicators
- Factor
- Metric

For each element, it provides a set of methods retrieving data related to this element assessment. For example for the strategic element, the API provides methods to get:

- the list of the strategic indicators and their "last" evaluation
- the list of the strategic indicators and the evaluations belonging to the specific period
- the list of the strategic indicators. For each strategic indicator, it returns the list of factors associated to it and their "last" evaluation
- the list of the strategic indicators. For each strategic indicator, it return the list of factors associated to it and their evaluations belonging to the period
- the list of factors associated to the strategic indicator evaluation passed as a parameter and their "last" evaluation
- the list of factors associated to the strategic indicator evaluation passed as parameter and their evaluations belonging to the period
- the list of metrics associated to every factor of a strategic indicator passed as parameter and the "last" metric evaluation

Other features:

- Indexes prefix can be configured when the connection is initialised
- Elastic search connection can be configured when the connection is initialised

Annex G. How to Deploy Q-Rapids Dashboard

This deployment guidelines includes:

- Instructions to deploy a new version of the dashboard in a current installation
- Instructions to install and configure the dashboard for the first time
- Some information about the Elastic Search indexes used to provide information to the dashboard

Deploying a new version of Q-Rapids Dashboard

In the case of Q-Rapids Dashboard version upgrade, you need to follow the next steps.

STEP 1: UNPACKING Q-RAPIDS DASHBOARD

The installation package includes the `.war` file to be deployed in the web server. You can replace your version with the new one having both in the same web server using different names.

Details about deploying a web application in a web server in next section Step 1 of this document. Remember to keep your configuration in the new deployment (details innext section - Step 1).

STEP 2: CONFIGURING Q-RAPIDS DASHBOARD

You need to edit the application configuration file to maintain the values for your installation. Details about the parameters in Section 0 (Step 3) of this document.

Installing Q-Rapids Dashboard (first time)

Instructions to install Q-Rapids Dashboard in a new location.

PREREQUISITES

Q-Rapids Dashboard is a web application developed with Java. Some data is stored in a PostgreSQL database and some in an Elasticsearch Cluster.

The requisites to deploy Q-Rapids Dashboard are:

- A web server, e.g. Tomcat (please refer to `http://tomcat.apache.org` for installation instructions).
- Oracle Java JRE (please refer to `https://java.com/en/download/` for installation instructions)
- PostgreSQL, an open source object-relational database system (please refer to `https://www.postgresql.org/download/`). The development has been tested using version 9.5.9 for Windows. The information that you need to configure the connection to the database is:
    - URL and port where PostgreSQL is accessible
    - username
    - password
- Q-Rapids Source Data Connectors (please refer to the specific deployment documentation).

OPTIONAL TOMCAT CONFIGURATION

If the selected web server is Tomcat, you need to check the configuration for the allocated memory, Q-Rapids Dashboard requires at least 512M. The CATALINA_OPTS environment variable can be configured as follows:

```
CATALINA_OPTS=-Xms512M -Xmx1024M –server
```

Tomcat server manager interface have a limit for the file size to be uploaded when a web application is deployed. The default limit is 50MB (52428800 bytes) and the `.war` file to deploy Q-Rapids Dashboard is

bigger than 50MB. This limit, which is expressed in bytes, should be configured in the tomcat manager configuration file (<Tomcat folder>\webapps\manager\WEB-INF\web.xml):

```
<multipart-config>
  <!-- 50MB max: 52428800-->
  <max-file-size>52428800</max-file-size>
  <max-request-size>52428800</max-request-size>
  <file-size-threshold>0</file-size-threshold>
</multipart-config>
```

STEP 1: DEPLOYING Q-RAPIDS DASHBOARD

The installation package includes the `.war` file to be deployed in the web server.

If you use Tomcat as web server, you have two options:

- Manual installation:
  - Stop Tomcat
  - Copy the .war file into the folder named "`webapps`" inside the folder where Tomcat is installed
  - Restart Tomcat
- Use the Tomcat server manager interface (e.g. `http://<Tomcat IP>/manager`)

Note: The name of the war file will be the name used to access to the dashboard in the browser, if you want to use a different name, you only need to rename the file just before deploying it in the web server.

STEP 2: CREATING THE DATABASE

In the installation package there are some scripts to create the database and add some data.

- `dashboardDB.sql`: the script creates an empty database (no tables) named `postgres`, the owner user is "`potsgres`". If you want to have a different database name or username, these values should be changed before the script is run
- `dashboardDB_tables.sql`: the script creates the tables.
- `dashboardDB_PoC.sql`: the script inserts in the corresponding tables the information for following strategic indicators: Blocking (id: blocking), Product Quality (id: productquality), Customer Satisfaction (id: customersatisfaction), and On-Time Delivery (id: ontimedelivery)

ADDING STRATEGIC INDICATORS TO THE DATABASE MANUALLY

In order to create a new strategic indicator, information should be inserted in two tables: `strategic indicator` and `kpi`. Following SQL sentences create a new strategic indicator named `Productivity`.

```
insert into public.strategicindicator(si_ID, description, name) values ('productivity','Productivity','Description for productivity');

insert into public.kpi (KPI_ID, target, lowerthreshold, upperthreshold) values ('productivity ',0.5,0.25,0.75);
```

In both tables the value for the ID fields should be the same (si_ID and KPI_ID). This value should be also the same that is used in the Elastic Search Engine to identify the same strategic indicator evaluations.

STEP 3: CONFIGURING Q-RAPIDS DASHBOARD

You need to configure the connection to the PostgreSQL database and the connection to the Elastic Search Engine. These connection should be configured editing the application.properties file following the next steps:

- Stop the webserver
- Open the application.properties file. It is a text file is located in `<Tomcat folder>\webapps\QRapids-X.Y.Z\ WEB-INF\classes.`
- Write the correct values for the following variables:

```
# Local connection
spring.datasource.url=jdbc:postgresql://localhost:5433/postgres
spring.datasource.username=postgres
spring.datasource.password=QRapids

# QMA connection
qma.ip=AAA.BB.CC.CC
qma.port=XXX
qma.prefix=poc.
```

The variable `qma.prefix` allows gathering data from more than one product. If you want to have evaluations for multiple product, you need to have the same indexes (strategic_indicators, factors, and metrics) with different prefixes (e.g. prd1.stratetgic_indicators and prd2.strategic_indicators). You need to change the value for the qma.prefix variable and restart the web application to be sure that the new value will be used.

Elastic Search Indexes

The current version of Q-Rapids Dashboard accesses to the following indexes, defined as follows:

- Strategic indicators: <prefix>strategic_indicators (e.g. poc.strategic_indicators):
  - `strategic_indicator` identifying the strategic indicator, this is the field that is used to link the information about the strategic indicator with the information in the database), `description, evaluationDate, datasource.`
- Product factors and process factors: <prefix>factors (e.g. poc.factors)
  - `factor (identifying the factor), description, strategic_indicators (array of strategic indicators IDs), evaluationDate, datasource`
- Normalized metrics: <prefix>metrics (e.g. poc.metrics)
  - `metric, description, factors (array of factors IDs), evaluationDate, datasource`

Fields `description` and `datasource` are not shown in the current version of Q-Rapids Dashboard. In order to identify the strategic indicators, the value used to identify them in the strategic indicator index should be the same that the value that identifies them in the corresponding tables in the database (`strategicindicator`, `kpi`). The fields that identify an evaluation are `ID` and `evaluationDate`.

Note: `<prefix>` is configured in the configuration file, variable qma.prefix (see previous section).

