



## D1.2 Data gathering and analysis proof-of-concept

### V1.0

<b>Programme</b>	H2020		
<b>Funding scheme</b>	RIA - Research and Innovation action		
<b>Topics</b>	ICT-10-2016 - Software Technologies		
<b>Project number</b>	732253		
<b>Project name</b>	Quality-aware rapid software development		
<b>Project duration</b>	1 <sup>st</sup> November 2016 – 31 <sup>st</sup> October 2019		
<b>Project website</b>	<a href="http://www.q-rapids.eu">www.q-rapids.eu</a>		
<b>Project WP</b>	WP1 – <i>Data Gathering and Analysis</i>		
<b>Project Task</b>	Task 1.2 – <i>Data gathering concept and implementation</i> Task 1.4 – <i>Data Analysis concept and implementation</i>		
<b>Deliverable type</b>		R	Document, report
	X	DEM	Demonstrator, pilot, prototype
		DEC	Websites, patent filings, videos, etc.
		OTHER	Material that does not belong to any specified category
		ETHICS	Ethics requirement
<b>Contractual delivery</b>	31/01/2018		
<b>Delivered</b>	31/01/2018		
<b>Responsible beneficiary</b>	Organisation		
<b>Dissemination level</b>	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
V0.1	18/12/2017	Initial structure of the document	Silverio Martinez (Fraunhofer IESE)
V0.2	10/01/2018	Updated structure of the document	Andreas Jedlitschka, Silverio Martinez (Fraunhofer IESE)
V0.3	12/01/2018	Updates based on internal reviewers feedback	Silverio Martinez (Fraunhofer IESE)
V0.4	19/01/2018	Pre-final version for internal review	Silverio Martinez, Axel Wickenkamp (Fraunhofer IESE)
V0.41	24/01/2018	Adding internal reviews and demonstration of GitLab	Silverio Martinez (Fraunhofer IESE), Michal Choras (ITTI)
V0.5	26/01/2018	Version addressing the internal reviews	Silverio Martinez (Fraunhofer IESE)
V1.0	31/01/2018	Final version (v1.0)	Silverio Martinez (Fraunhofer IESE)



Authors	
Organisation	Name
Fraunhofer IESE	Silverio Martinez
Fraunhofer IESE	Andreas Jedlitschka
Fraunhofer IESE	Axel Wickenkamp
Reviewers	
UPC	Lidia Lopez
Bittium	Jari Partanen

## 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 and abbreviations	
A term or an abbreviation (alphabetical order)	Explanation of the term or the abbreviation
<b>Apache Kafka</b>	Apache Kafka is an open-source stream processing platform developed by the Apache Software Foundation written in Scala and Java. The project aims to provide a unified, high-throughput, low-latency platform for handling real-time data feeds. Its storage layer is essentially a massively scalable pub/sub message queue architected as a distributed transaction log, making it highly valuable for enterprise infrastructures to process streaming data
<b>Assessed metric</b>	A concrete description of how a specific product factor should be quantified for a specific context.
<b>Elasticsearch</b>	Elasticsearch is a search engine based on Lucene. It provides a distributed, multitenant-capable full-text search engine with an HTTP web interface and schema-free JSON documents.
<b>ELK stack or Elastic Stack</b>	Elasticsearch is developed alongside a data-collection and log-parsing engine called Logstash, and an analytics and visualisation platform called Kibana. The three products are designed for use as an integrated solution, referred to as the "Elastic Stack" (formerly the "ELK stack").
<b>Product factor</b>	Attributes of parts of the product. They need to be concrete enough to be measured.
<b>qr-connect</b>	A module of the Q-Rapids tool for data gathering.
<b>qr-eval</b>	A module of the Q-Rapids tool for data analysis.
<b>Quality model</b>	The main purpose of a quality model is to link the data gathered from some data sources to the strategic indicators.
<b>Quality assessment</b>	The operationalisation and execution of a quality model with specific values/measures for all its elements in a timestamp: strategic indicators, product factors, and assessed metrics.
<b>Raw data</b>	Raw data are the data as it comes from the different data sources (without any modification). Typically it cannot be broken down into simpler or more granular forms of data.
<b>Strategic indicator</b>	An aspect that a company considers relevant for the decision-making process.



## Contents

The list of tables .....	6
The list of figures .....	7
Executive summary.....	8
1. Introduction.....	9
1.1 Motivation .....	9
1.2 Intended audience.....	9
1.3 Scope .....	9
1.4 Relation to other deliverables .....	9
1.5 Structure of the deliverable .....	10
2. Q-Rapids Tool: Data Gathering and Analysis.....	10
2.1 Overview: Q-Rapids Tool Architecture .....	10
2.2 Data Gathering and Analysis Modules .....	10
Data gathering: the qr-connect module.....	12
Data analysis: the qr-eval module .....	12
3. Data Gathering and Analysis Implementation .....	15
3.1 Progress on User Stories for Data Gathering .....	15
3.2 Progress on User Stories for Data Analysis.....	17
3.3 Evaluation of Technical Aspects .....	18
Evaluation of technical aspects .....	18
Challenges faced during the Q-Rapids tool deployment in the use cases .....	18
4. Demonstration of Data Gathering and Analysis.....	20
4.1 Demonstration of Data Gathering User Stories .....	20
Start Apache Kafka .....	20
SVN Connector.....	21
Jenkins Connector.....	23
Sonarqube Connector.....	24
Jira Connector.....	26
Redmine Connector .....	27
GitLab connector .....	28
4.2 Demonstration of Data Analysis User Stories .....	29
Assessed Metrics .....	29
Product Factors.....	30
Strategic Indicators.....	32
Performing the quality model assessment.....	33
Elasticsearch API: Access .....	33
5. Data Gathering and Analysis Software and Installation .....	35
5.1 Q-Rapids Data Gathering and Analysis Modules: Download .....	35
5.2 Development, Communication and Support among Q-Rapids Partners .....	35
Conclusion .....	36
Annex A – Quality Model for the Proof-of-Concept.....	37
Blocking .....	37
Product Quality.....	38
Annex B – Data Gathering and Analysis Modules Installation and Use (Axel) .....	40
How to deploy the Q-Rapids Source Data Connectors: the qr-connect module .....	40
Prerequisites.....	40
QR-Connect Framework .....	40
SVN Source Connector.....	41
Jenkins Source Connector.....	43
Jira Source Connector.....	46
Sonarqube Source Connector.....	48



Redmine Source Connector .....	53
GitLab connector (alternative in the ITTI use case).....	55
How to define the quality model indexes in Elastic. ....	63
How to customise the quality model and perform the quality model assessment: the qr-eval-module...	66
Setup.....	66
Run.....	66

## The list of tables

Table 1. Structure of qr-connect and qr-eval modules. ....	11
Table 2. Connectors of qr-connect module.....	12
Table 3. Progress on the user stories for data gathering from D1.1. ....	16
Table 4. Progress on the user stories for data analysis from D1.1.....	17
Table 5. Example of configuring the connection of a connector to a data producer: SVN repository. ....	21
Table 6. Example of a connection to an Elasticsearch server. ....	21
Table 7. An excerpt of the metrics.properties file. An example for “non-complex files” and “commented files” .....	29
Table 8. The factors.properties file, containing the product factors. ....	30
Table 9. An excerpt of the indicator.properties file, containing the strategic indicators. ....	32
Table 10. Quality model for the proof-of-concept: blocking strategic indicator. ....	37
Table 11. Quality model for the proof-of-concept: product quality strategic indicator. ....	38
Table 12. Data gathered about commits with the svn connector: attributes, descriptions, and examples... 41	
Table 13. Data gathered about builds of projects with the Jenkins connector: attributes, descriptions, and examples.....	43
Table 14. Data gathered about issues with the Jira connector: attributes, descriptions, and examples. ....	46
Table 15. Data gathered about measures with the SonarQube connector: attributes, descriptions, and examples.....	48
Table 16. Data gathered about quality rules issues with the SonarQube connector: attributes, descriptions, and examples.....	50
Table 17. Data gathered about issues with the Redmine connector: attributes, descriptions, and examples. ....	53
Table 18. Metrics index mapping. ....	63
Table 19. Factors index mapping.....	64
Table 20. Strategic indicators index mapping. ....	64



## The list of figures

Figure 1. Conceptual architecture of the Q-Rapids system.....	10
Figure 2. Data gathering and analysis modules.....	11
Figure 3. Quality model for the proof-of-concept: the “product quality” strategic indicator. ....	13
Figure 4. Quality model for the proof-of-concept: the “blocking” strategic indicator. ....	14
Figure 5. Examples with the computation of the assessed metrics: “commented files” and “non-complex files” of code quality (product quality strategic indicator). ....	15
Figure 6. Start Zookeeper. ....	20
Figure 7. Start Kafka. ....	20
Figure 8. Example of starting a connector: connect-svn. ....	22
Figure 9. Commit data gathered by the connect-svn connector and stored in the svn elastic search index. ....	22
Figure 10. Test data gathered by the connect-jenkins connector and stored in the jenkins elastic search index. ....	23
Figure 11. Issues (i.e., quality rules violations) data gathered by the connect-sonarqube connector and stored in the sonarqube.issues elastic search index. ....	24
Figure 12. Static code analysis measures data gathered by the connect-sonarqube connector and stored in the sonarqube.measures elastic search index. ....	25
Figure 13. Issues (i.e., tasks) data gathered by the connect-jira connector and stored in the jira elastic search index.....	26
Figure 14. Issues (i.e., tasks) data gathered by the connect-redmine connector and stored in the redmine elastic search index. ....	27
Figure 15. The schema of gitlab.stats.<project name> index.....	28
Figure 16. The assessed metrics from quality model assessments: the poc.metrics elastic search index. ....	30
Figure 17. The product factors from quality model assessments: the poc.factors elastic search index. ....	31
Figure 18. The strategic indicators from quality model assessments: the poc.strategic indicators elastic search index.....	32
Figure 19. Example of an execution of a quality model assessment for the day 19.01.2018.....	33
Figure 20. Accessing the quality model assessment via Elasticsearch Search API. ....	34
Figure 21. GitLab connector configuration file.....	56
Figure 22. Relation between issue and its notes.....	57
Figure 23. Example of Issue note object.....	57
Figure 24. Example of issue object .....	58
Figure 25. Example conversation (on GitLab) composed of issue notes.....	59
Figure 26. Example of FMS of an issue. Created is an initial state, while the Closed is a terminal state.....	60
Figure 27. Example shown how we calculate daily snapshot statistics for the entire project. ....	61
Figure 28. Metrics assessing the development part: number of tasks in backlog, number of task being under development, and number of tasks waiting for testing (the values are shown as average calculated for a period of a sprint).....	61
Figure 29. Metrics assessing the performance of testing. ....	62
Figure 30. Example of global metrics characterizing general advancement of the project.....	62
Figure 31. Defining quality model indexes in ELK.....	65



## Executive summary

The D1.2 is an output of the “Data gathering concept and implementation” (T1.2) and “Data Analysis concept and implementation” (T1.4). It is a *demonstrator* to show the current status of the Q-Rapids tool at the end of the proof-of-concept at month 15. The **objective** of this document is to provide the **description of a first set-up of the architecture for data gathering and analysis as a proof of concept**. Bearing this goal in mind, this document mainly provides:

- An update of the data gathering and analysis architecture (from the previous D1.1 and synchronised with D3.2, D4.4).
- A report on the implementation progress made on epics and user stories for the Q-Rapids data gathering and analysis tool.
- A demo of the deployment of the data gathering and analysis modules of the Q-Rapids tool.
- The software solution, including:
  - source code,
  - data gathering and analysis installation package and documentation.





## 1. Introduction

The overall goal of this document is to provide a demonstration of a first set-up of the architecture and running modules for data gathering and analysis during the proof-of-concept phase.

### 1.1 Motivation

Q-Rapids is a data-driven project. This means that in order to assess the level of software quality during development and at runtime, we firstly need to gather and analyse data about that software. This deliverable explains the implemented software to perform data gathering and analysis with a Q-Rapids tool. It includes:

- An update of the data gathering and analysis architecture (from the previous D1.1 and synchronised with D3.2, D4.4).
- A report on the implementation progress made on epics and user stories for the Q-Rapids data gathering and analysis tool.
- A demonstration of the deployment of the data gathering and analysis modules of the Q-Rapids tool.
- The software solution, including:
  - source code,
  - data gathering and analysis installation package and documentation.

### 1.2 Intended audience

This deliverable is a report produced for all the members of the Q-Rapids project. Specifically, the results of this report are interesting and useful for the following stakeholders:

- The industry partners (i.e., Bittium, Softeam, iTTI, NOKIA), who deployed the data gathering and analysis modules to perform an initial assessment of the level of software quality of their use cases.
- The WP2-WP3-WP4-WP5 researchers, who are provided with the quality model, and the data gathering and analysis implementation. These artefacts become necessary to operationalise the quality model assessments in the use cases.

### 1.3 Scope

The scope of this document is the entire Q-Rapids project. This version of the document is a result of the second phase of WP1: the proof-of-concept (from month 7 to month 15). It is used only for demonstration purposes in this phase. Next phases of the Q-Rapids project will have their own demonstration: M24 (prototype), M33 (consolidation), M36 (final release).

### 1.4 Relation to other deliverables

This deliverable strongly relates to previously produced deliverables:

- Deliverable 1.1, describing the data gathering and analysis specification (the Deliverable 1.1 will be updated in M18 and M33).
- Deliverable 3.1, including an ontology containing terms used in WP1 and this document.
- Deliverables 4.2 and 4.3, describing the integration of the entire Q-Rapids tool.
- Deliverables 5.1 and 5.2, reporting the evaluation strategy of the Q-Rapids tool.

Also, the present document relates to contemporary deliverables:

- Deliverable 3.2, including a demonstration of the dashboard highly connected with the results from this document.
- Deliverable 4.4, summarizing the proof-of-concept of the entire Q-Rapids solution.
- Deliverables 5.3, describing the use cases deploying the Q-Rapids tool (whose data gathering and analysis modules are reported here).

## 1.5 Structure of the deliverable

This document is structured as follows. Section 2 describes the Q-Rapids tool architecture, and deeps into the data gathering and analysis modules implementing the quality model for the proof-of-concept and its operationalisation. Section 3 describes the progress made and reports technical aspects. Section 4 shows the flow of the data gathering and analysis modules through screenshots. Section 5 contains the links to download the delivered software as well as installation and use documentation. Finally, the conclusions are reported.

## 2. Q-Rapids Tool: Data Gathering and Analysis

This section describes the Q-Rapids tool architecture and deeps into its data gathering and analysis modules.

### 2.1 Overview: Q-Rapids Tool Architecture

The conceptual architecture of the Q-Rapids tool is shown in Figure 1. It shows the data flow from the data producers, to its analysis and visualisation. The raw data from the data producers feed the distributed data sink and data analysis and processing layers. For this purpose, we developed the connectors enabling acquisition from the data producers. The Distributed Data Sink is fed through Kafka Cluster as the Data Ingestion layer. Synchronized data is preliminarily processed, filtered and anonymized. Synchronization can be done with different time intervals (e.g., once a day) both automatically and manually.

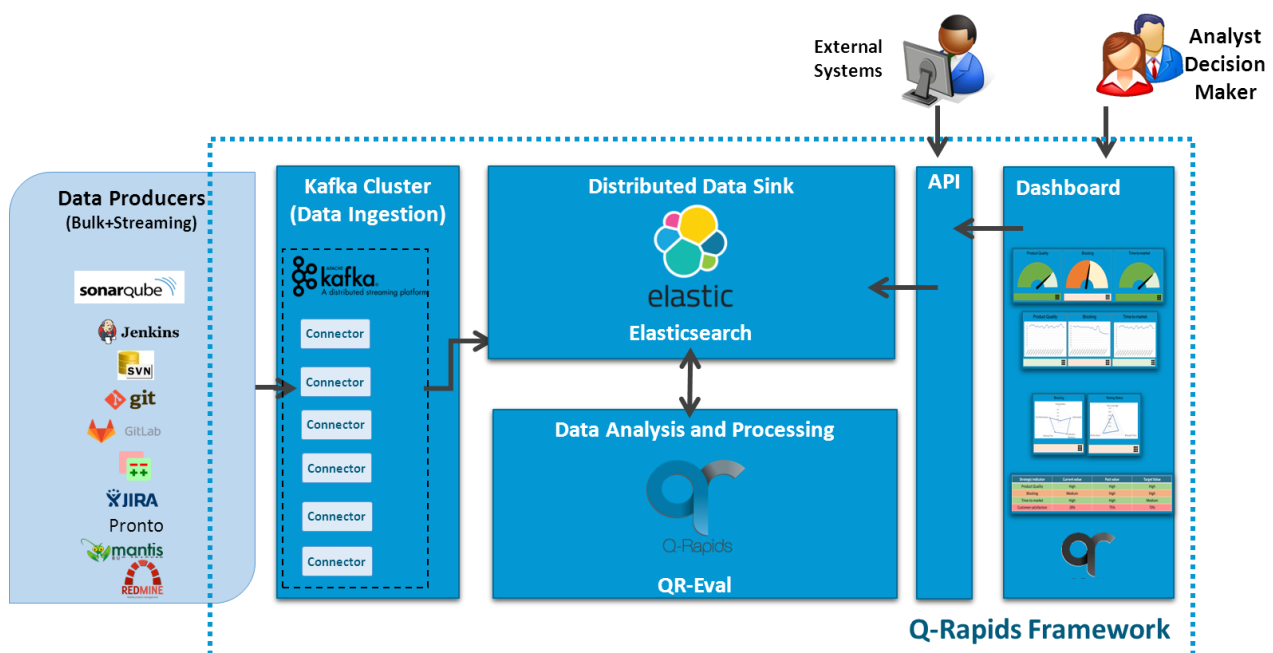


Figure 1. Conceptual architecture of the Q-Rapids system.

### 2.2 Data Gathering and Analysis Modules

In this subsection, we focus on the data gathering and analysis modules of the whole Q-Rapids tool.

The main goals of the data gathering and analysis modules are:

- to gather real data from several data sources, and,
- to analyse the collected data and operationalise quality model assessments.

Therefore, in the proof-of-concept stage, the **main outputs** of the data gathering and analysis modules are the daily **executed and operationalised quality model assessments**.

To achieve these goals, the data gathering and analysis has two modules, which are implemented as a Java projects in eclipse (see Figure 2 and Table 1):

1. **“qr-connect”**: It consists of several Kafka connectors to gather data from data producers (*connect-jenkins*, *connect-jira*, *connect-redmine*, *connect-sonarqube*, *connect-svn*) and push this data to elastic (*connect-elastic*).
2. **“qr-eval”**: It performs the quality model assessment based on the gathered raw data by qr-connect.

These two modules (qr-connect and qr-eval) rely on existing Big Data technologies. First, an **Apache Kafka** cluster serving as primary ingestion layer and messaging platform. Second, an **ELK stack** (Elasticsearch/Kibana) used for data indexing and analysis purposes.

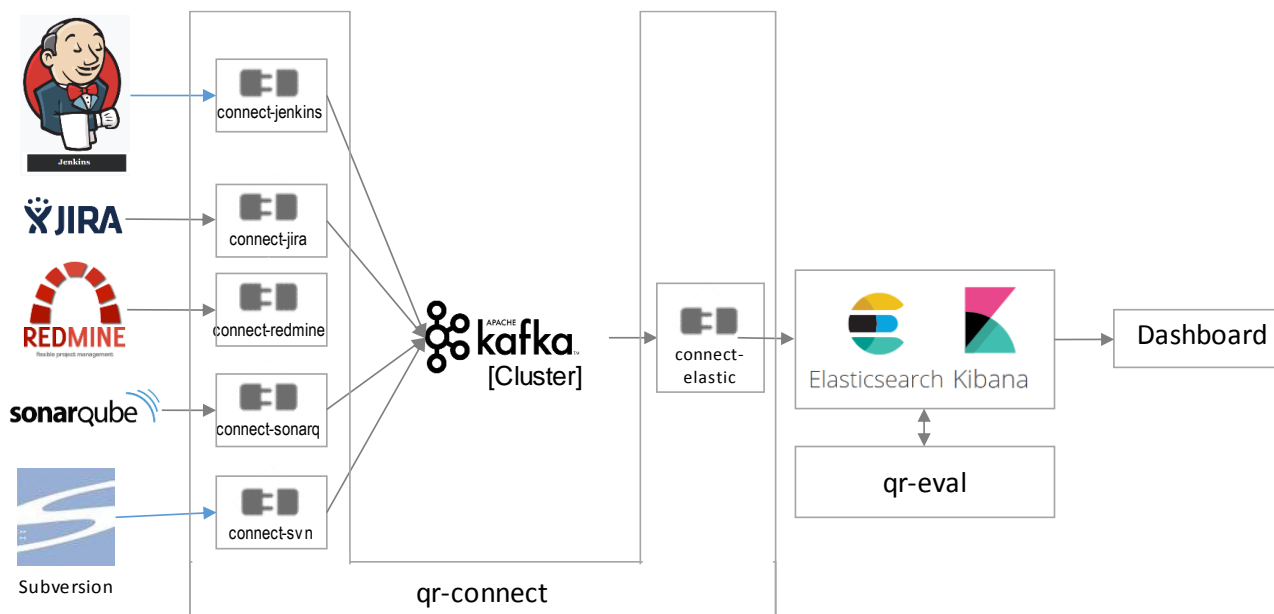


Figure 2. Data gathering and analysis modules.

Table 1. Structure of qr-connect and qr-eval modules.

<pre> └─ &gt; qr-connect [QRapidsWP1 master]   └─ src/main/java     ├── connect.jenkins     ├── connect.jira     ├── connect.redmine     ├── connect.sonarqube     ├── connect.svn     ├── model.jenkins     ├── model.jira     ├── model.redmine     ├── model.sonarcube     ├── model.sonarcube.issues     ├── model.sonarcube.measures     ├── rest     └── util   └─ &gt; src/main/resources   ├── JRE System Library [JavaSE-1.8]   ├── Maven Dependencies   ├── &gt; src   ├── target   ├── changes.txt   └── pom.xml         </pre>	<pre> └─ &gt; qr-eval [QRapidsWP1 master]   └─ src/main/java     ├── elastic     ├── elastic.model     ├── eval     ├── eval.props     └── log4j2.xml   ├── JRE System Library [JavaSE-1.8]   ├── Maven Dependencies   ├── src   ├── target   ├── factors.properties   ├── index.properties   ├── indicators.properties   ├── metrics.properties   ├── pom.xml   ├── qr-eval-0.0.1-jar-with-dependencies.jar   └── README.txt         </pre>
<b>qr-connect</b>	<b>qr-eval</b>



## Data gathering: the qr-connect module

This module includes several Kafka connectors. Table 2 briefly summarises their functionality.

Table 2. Connectors of qr-connect module.

Connector	Functionality
connect-jenkins	This connector reads data of the builds (e.g., passed) and tests (e.g., failed, skipped, and duration) from the API of Jenkins <sup>1</sup> . The information is accessed by a Rest Invoker directly accessing a JSON document, located in http. It needs authentication. You merely need to perform an HTTP request on: JENKINS_URL/job/<JOBNAME>/<BUILD_NUMBER>/api/json This is done from the poll() method in JenkinsSourceTask.java
connect-jira	This connector reads all the features from each issue (e.g., description, assignee, and due date) from the API of Jira <sup>2</sup> . The information is accessed by an HTTP request on: JIRA_URL/rest/api/2/search?jql=updated>"MostRecentUpdate"AND+project="ProjectKey"+order+by+updated+ASC This is done from the poll() method in JiraSourceTask.java
connect-redmine	This connector reads all the features from each issue (e.g., description, assignee, and due date) from the Redmine REST API <sup>3</sup> . The information is accessed by an HTTP request on REDMINE_URL/issues.json ?sort=updated_on:asc&updated_on=>= ... This is done from the poll() method in RedmineSourceTask.java
connect-sonarqube	This connector reads all static code analysis measures (e.g., cyclomatic complexity of files/directories/projects) and quality rules (e.g., from the Sonarway default quality profile) from the API of SonarQube <sup>4</sup> . The information is accessed by several HTTP request (e.g., SONARQUBE_URL/api/measures/component_tree?metricKeys= ... This is done from the poll() method in SonarqubeSourceTask.java
connect-svn	This connector reads a version control log. An open source java library <sup>5</sup> is reused. This is done from the poll() method in SubversionSourceTask.java
connect-elastic	A connector included in the confluent OSS Kafka distribution that reads data from Kafka and puts it into the Elastic stack.

As an alternative approach to data ingestion with Apache Kafka, in the ITTI use case and the GitLab data source, we have implemented the collection of information using a mixture of Node.js and MongoDB frameworks. Using JavaScript, we periodically pool the GitLab RESTful interface to retrieve the raw data, as well as to process and store information in MongoDB. Similarly, we use JavaScript scripts to periodically push the pre-processed data the ElasticSearch platform.

A demonstration of the functionalities of the qr-connect module is available in Section 4.1.

## Data analysis: the qr-eval module

This module includes:

1. The static view of the quality model (i.e., the relationships among relevant strategic indicators, product factors, and assessed metrics).

<sup>1</sup> Jenkins API: <https://wiki.jenkins.io/display/JENKINS/Remote+access+API>

<sup>2</sup> JIRA API: <https://docs.atlassian.com/software/jira/docs/api/REST/6.0.1/>

<sup>3</sup> Redmine API: [http://www.redmine.org/projects/redmine/wiki/Rest\\_api](http://www.redmine.org/projects/redmine/wiki/Rest_api)

<sup>4</sup> SonarQube API: <https://docs.sonarqube.org/display/DEV/Web+API#WebAPI-HTTPBasicAccess>

<sup>5</sup> SVN Kit Java library <https://svnkit.com/>



2. The functionalities to produce the operationalised quality model assessments (i.e., the operationalisation and execution of a quality model with specific values/measures for all its elements in a timestamp: strategic indicators, product factors, and assessed metrics).
3. The capabilities to access the full operationalised quality model from other modules or external systems.

We respectively show these three aspects below.

#### Quality model: static view

In the proof-of-concept, the hierarchy of the quality model has four levels:

- Strategic indicator: An aspect that a company considers relevant for the decision-making process.
- Product factors: Attributes of parts of the product. They need to be concrete enough to be measured.
- Assessed metric: A concrete description of how a specific product factor should be quantified for a specific context.
- Raw data and data sources: Raw data are the data as it comes from the different data sources (without any modification). Typically it cannot be broken down into simpler or more granular forms of data.

In the beginning of the proof-of-concept phase (month 7), it was decided in the plenary meeting held in Kaiserslautern that the proof-of-concept should be scoped to two strategic indicators: product quality, and blocking. Below, we show two static views of the quality model for product quality (see Figure 3) and blocking (see Figure 4). For the proof-of-concept, this scoped quality model has been deployed in the four use cases of the Q-Rapids project. Further details about the calculation of assessed metrics are shown in Annex A. The deliverable D5.3 about the evaluation of the use cases contains further details about the usage of both the quality model and its assessments by the companies (evaluations and feedback). As an activity during these evaluations, an explanatory video showing the quality model was presented.

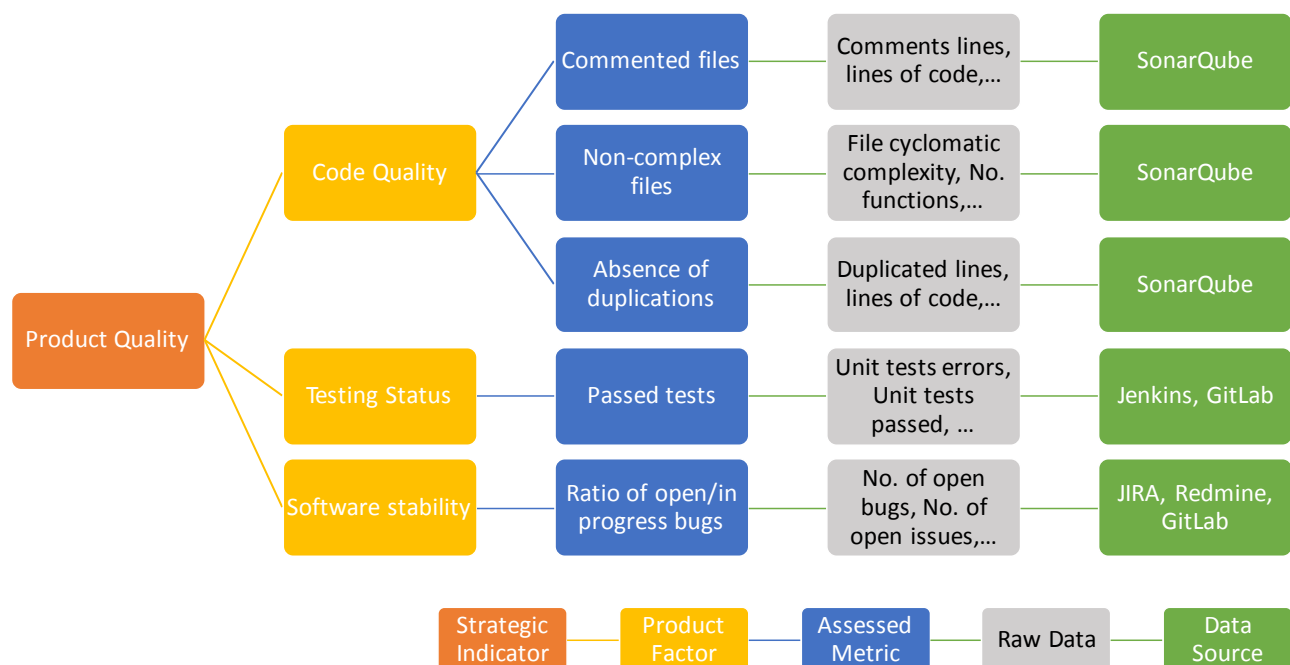


Figure 3. Quality model for the proof-of-concept: the "product quality" strategic indicator.

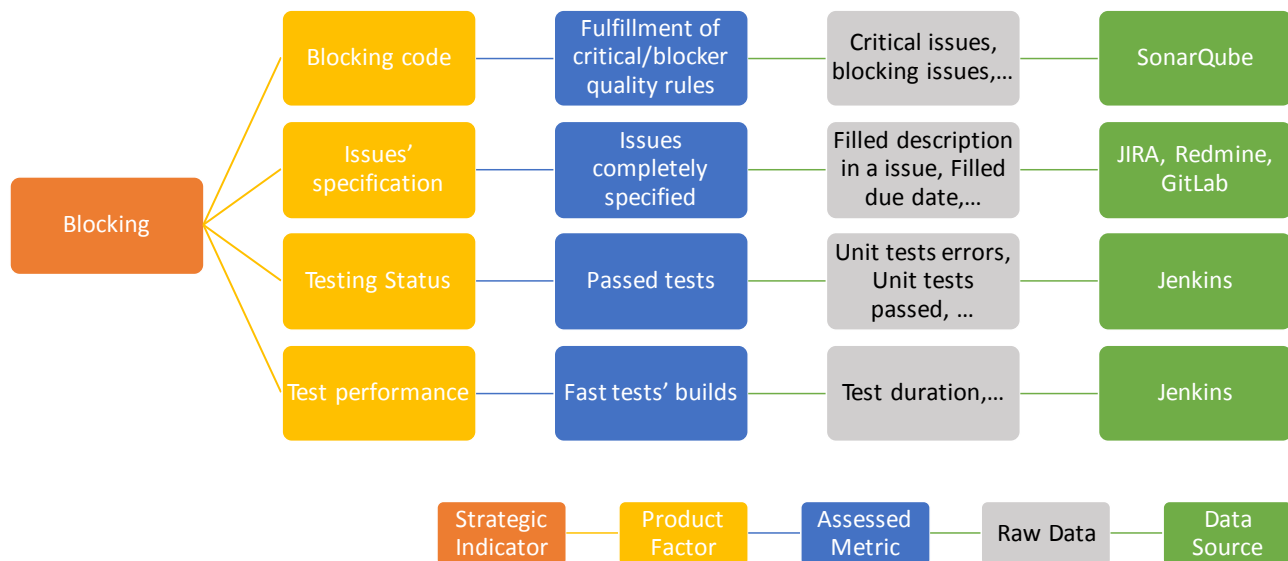


Figure 4. Quality model for the proof-of-concept: the "blocking" strategic indicator.

The static view of the quality model is stored in the `.properties` files of the `qr-eval` module. In these files, the user can indicate which product factors belong to which strategic indicators, which assessed metrics to which product factors, and customise the thresholds of the assessed metrics (see a demo in Section 4.2).

#### Quality model assessment execution

The quality model assessment (also known as operationalisation or execution of the quality model) consists of:

1. The computation of assessed metrics based on raw data coming from data sources. For this, concrete formulas and utility functions are needed.
2. The aggregation of:
  - a. Assessed metrics into product factors (the weights of assessed metrics are needed).
  - b. Product factors into strategic indicators (the weights of product factors are needed).

The bottom-up approach to perform the quality model assessment is shown in the example of Figure 5.

First, assessed metrics are computed from raw data (e.g., M1: Density of comments of a file, and M2: Total number of files), and the utility function. Utility functions are customised in the `metrics.properties` file, and interpret the raw data value by either the preferences of experts or learned data. After this interpretation, assessed metrics (e.g., AM1: Commented files) have values from 0 to 1, where 0 is the worst value and 1 is best value regarding quality.

Second, assessed metrics are aggregated into product factors (considering their weights), and then product factors are aggregated into strategic indicators.

In the `qr-eval` module, these two steps of the operationalised quality model assessment is performed by the `eval()` method of `Eval.java`. By default, a cron task is executing this method once per day, but further options are also available (e.g., manual).

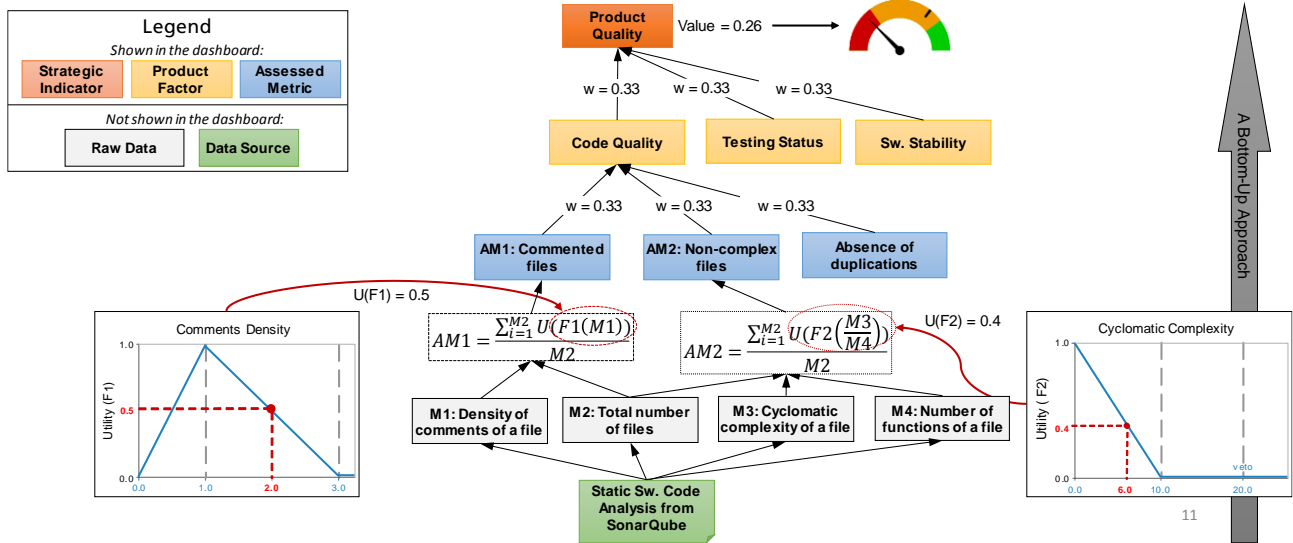


Figure 5. Examples with the computation of the assessed metrics: “commented files” and “non-complex files” of code quality (product quality strategic indicator).

### Quality model assessment storage and access

The quality model assessment is stored in the Elastic stack. As we have already seen, a quality model has several levels of abstraction (see Figure 3 and Figure 4). Each of these levels can be accessed as an index in the Elastic stack. We defined four types of indexes in the Elastic stack cluster:

1. Strategic indicators: *poc.strategic\_indicators*
2. Product factors and process factors: *poc.factors*
3. Normalized metrics: *poc.metrics*
4. Raw data, metrics as collected: *sonarqube.measures*, *sonarqube.issues*, *jira*, *svn*, *jenkins*, *gitlab.stats*

A demonstration of the functionalities of the qr-eval module is available in Section 4.2.

## 3. Data Gathering and Analysis Implementation

This section describes the progress made on the relevant user stories from D1.1, and reports the feedback on technical aspects.

### 3.1 Progress on User Stories for Data Gathering

Table 3 reports the progress made in the Q-Rapids tool for the proof-of-concept version about data gathering. The third column reports the progress, which could fit into one of the following categories:

- Open — This user story is in the initial 'Open' state.
- In Progress — This user story is being actively worked on at the moment.
- Resolved — A Resolution has been implemented for the proof-of-concept, and this user story is awaiting verification by the feedback from the use cases and evaluation. From here, issues are either 'Reopened' or are 'Closed'.
- Reopened — This user story was once 'Resolved' or 'Closed', but is now being re-examined. From here, issues are either marked In Progress, Resolved or Closed.
- Closed — This user story is complete.





Table 3. Progress on the user stories for data gathering from D1.1.

Factor	User stories	Progress	Comments
Code Quality (Maintainability)	As a developer and integrator, I want to gather data about the quality of the code committed by the developers, so that we can improve the quality of the code delivered by the developers.	Resolved	<i>connect-sonarqube</i> connector of <i>qr-connect</i> module
	As a code guardian, I want to gather data about how coding guidelines are followed (e.g., complexity and function size, coverage and duplications, and technical debt), so that we can manage resources for maintainability.	Resolved	<i>connect-sonarqube</i> connector of <i>qr-connect</i> module
	As a developer, I want to gather data about the impact a code change has on complexity, so that we can identify how rising complexity deteriorates maintainability.	In progress	<i>connect-sonarqube</i> and <i>connect-svn</i> connectors of <i>qr-connect</i> module
Testing (Reliability)	As a test manager, I want to gather data about the quality and stability level of regression testing, so that regression tests cases are not failed or skipped (ensuring reliability and integratability of the platform)	Resolved	<i>connect-jenkins</i> connector of <i>qr-connect</i> module
	As a quality assessor and integrator, I want to gather data about the current state of the integration process, so that we can improve the quality of the code delivered by the developers.	In progress	<i>connect-jenkins</i> connector of <i>qr-connect</i> module
Time-to-Complete Issues (Productivity)	As a product owner/project manager, I want to gather data about content delivered at Feature Build (FB) compared to planned content on exit, so that we can see the planning capability and accuracy of the team.	In progress	<i>connect-jira</i> and <i>connect-redmine</i> connectors of <i>qr-connect</i> module; alternative GitLab connector
Usage (Functional Suitability)	As a product director, I want to gather data about the features heavily used by customers, so that it is completely clear how heavily each feature is used by customers and why.	Open	
	As a product owner/UX designer, I want to gather data about product usage (e.g., total time spent on functionalities, and most/least used functionalities), so that we can know if an application is used and to what extent.	In progress	We are working on an extension of <i>qr-connect</i> for mining logs to study the usage of systems.
Bugs and Issues (Reliability)	As a quality assessor, I want to gather data about the number of defects discovered during validation and in operation, so that the proportion of bugs discovered in validation and in operation is completely clear.	In progress	<i>connect-jira</i> and <i>connect-redmine</i> connectors of <i>qr-connect</i> module; alternative GitLab connector
	As a product director, I want to gather data about the most critical issues in operation, so that it is completely clear what the most critical issues are (we need stats over time and over user base).	Open	
	As a quality manager, I want to gather product reliability data for KPIs (e.g., MTBF), so that we can maintain efficient service capability/quality/prioritisation.	In progress	<i>connect-jira</i> and <i>connect-redmine</i> connectors of <i>qr-connect</i> module





### 3.2 Progress on User Stories for Data Analysis

Table 4 reports the progress made in the Q-Rapids tool at the proof-of-concept version about data analysis. The third column reports the progress as explained in the above subsection.

Table 4. Progress on the user stories for data analysis from D1.1.

Subject	User stories	Progress	Comments
Computation of basic metrics from possibly diverse data sources	As a Q-Rapids researcher, I want basic metrics from the Q-Rapids quality model to be computed from the data gathered during data ingestion, so that metrics can be available for data analysis approaches.	In progress	<i>qr-eval</i> module. The assessed metrics of the proof-of-concept are resolved, but new assessed metrics will be added in the next phases.
Storage of basic metrics	As a Q-Rapids researcher, I want the computed basic metrics to be stored in a suitable format (either relational or NoSQL, e.g., HDFS or HBase), so that we can later exploit parallelism with Big Data analysis technologies (e.g., Spark or Hadoop).	Resolved	<i>connect-elatic</i> connector of <i>qr-connect</i> module and <i>qr-eval</i> module
Combination of metrics from heterogeneous data sources	As a Q-Rapids researcher, I want to be able to combine metrics from different data sources, so that we can reason about product factors with more than one data source.	In progress	<i>qr-eval</i> module
Aggregation of basic metrics into product factors	As a Q-Rapids researcher, I want the computed basic metrics to be stored in a way that they can be easily grouped or aggregated into product factors (following the Quamoco hierarchy), so that we can build a quality model from the basic measures.	Resolved	<i>qr-eval</i> module
Aggregation of product factors into quality factors	As a Q-Rapids researcher, I want product factors to be stored in a way that they can be easily grouped or aggregated into quality factors (following the Quamoco hierarchy), so that we can build a quality model from the product factors.	Open	
Utility functions for proposing candidate quality requirements	As a Q-Rapids researcher, I want to be able to define utility functions to model the preferences of decision makers with respect to the measures defined for the factors, so that we can analyse when a basic metric has exceeded a defined threshold.	In progress	<i>qr-eval</i> module
Relevant metrics for product/quality factors	As a Q-Rapids researcher, I want to be able to perform analysis approaches to identify relevant metrics for predicting quality factors, so that we can rely on the most relevant metrics.	Open	
Prediction of future values of basic metrics	As a Q-Rapids researcher, I want to be able to perform analysis approaches to predict the value of basic metrics in different contexts (i.e., with or without the consideration of candidate quality requirements), so that managers can see the benefits of prioritising a quality requirement in the product backlog.	Open	



### 3.3 Evaluation of Technical Aspects

This section respectively reports:

- Feedback on technical aspects of the data gathering and analysis modules.
- Lessons learned and challenges faced during the deployment of the data gathering and analysis modules at the use cases.

#### Evaluation of technical aspects

The quality model for the proof-of-concept has been evaluated in the use cases. The details about general feasibility (in the shape of reliability), appropriateness of the gathered data for analysis purposes, and appropriateness of the results from the analysis for decision support purposes will be reported in the parallel deliverable 5.3. As a summary, we report below the scalability and feasibility of the approach:

- **Scalability:** Kafka and Elastic Stack offer scalability via cluster capabilities. They are popular Big Data technologies. To check the scalability of storing and accessing the operationalised quality model assessments in the Elastic stack, we executed a stress testing scenario. We populated Elasticsearch with 30 million data points containing several characteristics and perform counting based on the name of those data points. All aggregations and counting tests took less than 1 second for those 30 million data points.
- **Feasibility:** One of the challenges for the proof-of-concept was the deployment of the Q-Rapids tool in the four use cases. Such deployment was feasible in all use cases, in which the following functionalities succeeded: gathering data with the Kafka connectors, storing the data in elastic search, executing the quality model assessments and storing them in the Elastic stack, and accessing the data gathering and analysis results from other modules (e.g., the dashboard module).

#### Challenges faced during the Q-Rapids tool deployment in the use cases

Regarding data gathering and analysis modules, the deployment of the Q-Rapids tool faced several challenges. It is relevant to report these problems because they might lead to improvement in the Q-Rapids tool and its architecture. These challenges have been mainly due to the deployment effort (it was suggested to consider a “container” strategy to support for easier deployment strategies in the companies for next phases) and the diversity in the use cases in:

- data producers and their versions,
- use of data producers, and,
- environment and network infrastructure.

#### Diversity of data producers and their versions

For the same activity, use cases have different tools. This has been already reported in D1.1. However, even when the use cases use the same tools, this does not imply that they use the same version. The connectors are mainly developed to extract information from the APIs of data producers, whose characteristics change among versions.

For instance, we have experienced different functionalities/features and constraints among SonarQube versions in the use cases. The API of SonarQube for v4.x.y only gives the measures for the first 500 resources (e.g., directory, files), and therefore it does not offer pagination. There is no way to get the metrics from all resources without a workaround. From ( $\geq$ v5.x.y), SonarQube offers pagination. On the other hand, newer versions ( $\geq$ v6.x.y) have a new constraint: no more than 15 measures can be obtained.

As a consequence, connectors of each tool have been customised to properly work for the different versions used by the use cases.



---

#### *Diversity of use among the data producers*

When the use cases use the same tool for a purpose, it does not imply that they use it in the same way.

For instance, when they use an issue tracking system as Jira, they may use customised “issue types” instead of the defaults ones. If we are looking for bugs, and they are not categorised in a de-facto manner, the quality model assessment module has to be customised to properly aggregate data. This was solved by making a customised option in which the type use for bugs could be defined.

Another example of different use is Jenkins. Depending on the structure of the builds, the tests results and duration may be in different places of the JSON documents. Hence, the connectors reading this information have been customised to properly parser and read customised JSON documents.

#### *Diversity of environment and network infrastructure in the use cases*

The environment and network infrastructure also plays an important role for connectors. Connectors should access external data producers. These external data producers and tools, even if they are the same, are deployed in different environments and networks with diverse security levels. As an example, SonarQube may not require authentication credentials, or even require an SSL connection. These problems are very difficult to solve, since it is needed to test the connectors in real conditions. To solve these issues, the use cases reported these technical issues via GitLab<sup>6</sup>.

---

<sup>6</sup> GitLab to report technical problems with the installation of the data gathering and analysis modules: <http://193.142.112.102/root/Proof-of-Concept/issues>



## 4. Demonstration of Data Gathering and Analysis

This section shows the flow of the data gathering and analysis modules through screenshots.

### 4.1 Demonstration of Data Gathering User Stories

The data-gathering infrastructure consists of the following elements:

- An Apache Kafka serving as primary ingestion layer and messaging platform.
- A set of so-called Kafka connectors (briefly summarized in Table 2) responsible for the collection of data from relevant source systems (i.e., SVN/git Version Control, Redmine/Jira issue tracking, SonarQube source code analysis, Jenkins Continuous Integration).
- An ELK stack (Elasticsearch/Kibana) used for data indexing and analysis purposes.

#### Start Apache Kafka

Apache Kafka depends on Apache Zookeeper, a tool for distributed configuration management. Zookeeper has to be started first and is configured by a properties file (zookeeper.properties). For the proof-of-concept, we used a basic single node setup. The configuration file for zookeeper defines a data directory, the zookeeper client port and a connection limit (disabled).

Start Zookeeper with the following command:

```
<Apache-Kafka-Dir>/bin/zookeeper-server-start <zookeeper-properties-file>
```

```
[2018-01-16 13:57:05,604] INFO Server environment:os.name=Linux (org.apache.zookeeper.server.ZooKeeperServer)
[2018-01-16 13:57:05,604] INFO Server environment:os.arch=amd64 (org.apache.zookeeper.server.ZooKeeperServer)
[2018-01-16 13:57:05,604] INFO Server environment:os.version=3.10.0-693.5.2.el7.x86_64 (org.apache.zookeeper.server.ZooKeeperServer)
[2018-01-16 13:57:05,604] INFO Server environment:user.name=bigdata (org.apache.zookeeper.server.ZooKeeperServer)
[2018-01-16 13:57:05,604] INFO Server environment:user.home=/home/bigdata (org.apache.zookeeper.server.ZooKeeperServer)
[2018-01-16 13:57:05,604] INFO Server environment:user.dir=/home/bigdata/app/qr-connect (org.apache.zookeeper.server.ZooKeeperServer)
[2018-01-16 13:57:05,611] INFO tickTime set to 3000 (org.apache.zookeeper.server.ZooKeeperServer)
[2018-01-16 13:57:05,611] INFO minSessionTimeout set to -1 (org.apache.zookeeper.server.ZooKeeperServer)
[2018-01-16 13:57:05,612] INFO maxSessionTimeout set to -1 (org.apache.zookeeper.server.ZooKeeperServer)
[2018-01-16 13:57:05,621] INFO binding to port 0.0.0.0/0.0.0.0:2181 (org.apache.zookeeper.server.NIOServerCnxnFactory)
```

Figure 6. Start Zookeeper.

Apache Kafka also needs a configuration file (server.properties). With the basic one-node configuration, the configuration file that comes with Apache Kafka is used without any change. Start Apache Kafka with the following command:

```
<Apache-Kafka-Dir>/bin/kafka-server-start <kafka-server-properties-file>
```

```
bigdata@bdc11:~/app/qr-connect
File Edit View Search Terminal Help
[2018-01-16 14:03:27,676] INFO [GroupCoordinator 0]: Startup complete. (kafka.coordinator.GroupCoordinator)
[2018-01-16 14:03:27,677] INFO [GroupMetadata Manager on Broker 0]: Removed 0 expired offsets in 1 milliseconds. (kafka.coordinator.GroupMetadataManager)
[2018-01-16 14:03:27,693] INFO Will not load MX4J, mx4j-tools.jar is not in the classpath (kafka.utils.Mx4JLoader)
[2018-01-16 14:03:27,714] INFO Creating /brokers/ids/0 (is it secure? false) (kafka.utils.ZKCheckedEphemeral)
[2018-01-16 14:03:27,719] INFO Result of znode creation is: OK (kafka.utils.ZKCheckedEphemeral)
[2018-01-16 14:03:27,719] INFO Registered broker 0 at path /brokers/ids/0 with addresses: EndPoint(bigdata-cl11.iiese.de,9092,ListenerName:bigdata-cl11.iiese.de:9092) (kafka.server.BrokerMetadataCheckpoint)
[2018-01-16 14:03:27,720] WARN No meta.properties file under dir /tmp/kafka-logs/meta.properties (kafka.server.BrokerMetadataCheckpoint)
[2018-01-16 14:03:27,739] INFO New leader is 0 (kafka.server.ZooKeeperLeaderElector$LeaderChangeListener)
[2018-01-16 14:03:27,742] INFO Kafka version : 0.10.2.1-cp1 (org.apache.kafka.common.utils.AppInfoParser)
[2018-01-16 14:03:27,742] INFO Kafka commitId : 078e7dc02a100018 (org.apache.kafka.common.utils.AppInfoParser)
[2018-01-16 14:03:27,743] INFO [Kafka Server 0], started (kafka.server.KafkaServer)
[2018-01-16 14:03:27,745] INFO Waiting 10015 ms for the monitored broker to finish starting up... (io.confluent.support.metrics.MetricsReporter)
[2018-01-16 14:03:37,763] INFO Monitored broker is now ready (io.confluent.support.metrics.MetricsReporter)
[2018-01-16 14:03:37,763] INFO Starting metrics collection from monitored broker... (io.confluent.support.metrics.MetricsReporter)
```

Figure 7. Start Kafka.



## SVN Connector

The SVN connector periodically polls information about commits from the SVN version control system. The execution of the connector is driven by a set of configuration files.

*connect-svn.properties* defines (among some other settings) the network address of the Kafka server to use, the location of a connect-offsets file (responsible for storing the last read position which is used when the connector is restarted) and a REST port for the connector. When multiple connectors are running on the same machine, the connect-offsets file and REST port has to be different for each running connector.

*connect-svn-source.properties* defines the connection to the SVN repository. It configures the Kafka Worker process responsible for fetching data from SVN.

Table 5. Example of configuring the connection of a connector to a data producer: SVN repository.

Attribute	Description	Example
<b>repository.url</b>	The repository base URL	https://<svn-server-ip>/repo
<b>repository.path</b>	The path is appended to the repository URL. Concrete values depend on your repository layout	/<projectname>/trunk
<b>repository.user</b>	Authentication Username	<Username>
<b>repository.pass</b>	Authentication Password	<Password>
<b>topic</b>	Kafka topic name for storing the svn data	svn
<b>name</b>	Name of the connector	kafka-svn-source-connector
<b>connector.class</b>	Classname of the class implementing the connector	connect.svn.SubversionSourceConnector
<b>poll.interval.seconds</b>	Polling interval in seconds	60

*svn-connect-elasticsearch.properties* defines the connection to an Elasticsearch server. It configures a Kafka Worker process responsible for reading data collected by the SVN source Worker and puts every record read into an Elasticsearch index.

Table 6. Example of a connection to an Elasticsearch server.

Attribute	Description	Example
<b>name</b>	Name of the connector	svn-elasticsearch
<b>connector.class</b>	Classname of the class implementing the connector	io.confluent.connect.elasticsearch.ElasticsearchSinkConnector
<b>tasks.max</b>	Max. number of concurrent tasks	1
<b>topics</b>	The Kafka Topic to read, Elasticsearch index name to store to	svn
<b>name</b>	Name of the connector	kafka-svn-source-connector
<b>connection.url</b>	URL of the Elasticsearch Server	http://<elasticsearch-server-ip>:9200
<b>type.name</b>	Index type name for Elasticsearch	svn

Start the SVN connector with the following command:

```
CLASSPATH=qr-connect-0.0.2-jar-with-dependencies.jar  
<kafka-install-dir>/bin/connect-standalone  
<qr-connect-install-dir>/connect-svn.properties
```



```
<qr-connect-install-dir>/connect-svn-source.properties
<qr-connect-install-dir>/connect-svn-elasticsearch.properties
```

```
bigdata@bdc11:~/app/qr-connect
File Edit View Search Terminal Help

schema.ignore = false
topic.index.map = []
topic.key.ignore = []
topic.schema.ignore = []
type.name = svn
(io.confluent.connect.elasticsearch.ElasticsearchSinkConnectorConfig:180)
[2018-01-16 15:15:35,419] INFO Using multi thread/connection supporting pooling connection manager (io.searchbox
[2018-01-16 15:15:35,477] INFO Using default GSON instance (io.searchbox.client.JestClientFactory:61)
[2018-01-16 15:15:35,477] INFO Node Discovery disabled... (io.searchbox.client.JestClientFactory:75)
[2018-01-16 15:15:35,477] INFO Idle connection reaping disabled... (io.searchbox.client.JestClientFactory:87)
[2018-01-16 15:15:35,481] INFO Sink task WorkerSinkTask{id=svn-elasticsearch-0} finished initialization and star
Jan 16, 2018 3:15:36 PM connect.svn.SubversionSourceTask poll
INFO: POLLED 403 SourceRecords.
```

Figure 8. Example of starting a connector: connect-svn.

After a successful run of the connector, the data collected is available in the Elastic stack. We can visualize the data in the Elastic stack through Kibana. In Figure 9, we can see the selected index (svn), the hits got during the selected period (403 hits), and chart with the frequency of data points, and a list with the attributes of each data point.

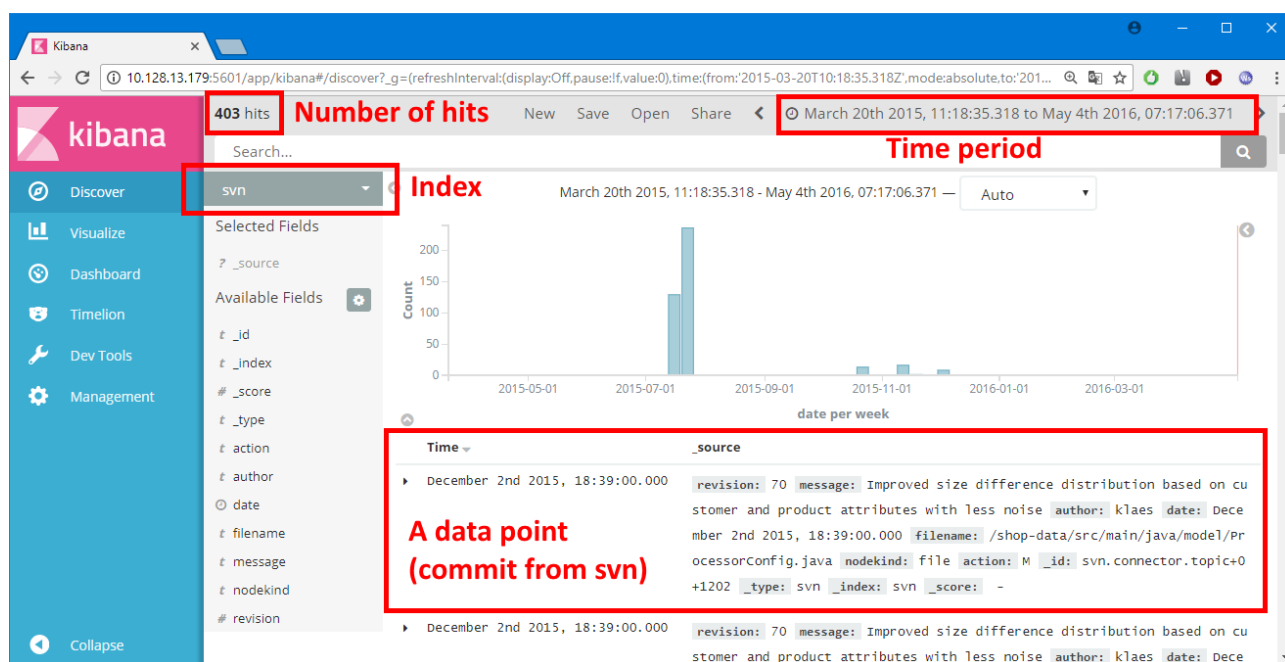


Figure 9. Commit data gathered by the connect-svn connector and stored in the svn elastic search index.

The data collected by the SVN connector has the following attributes: revision, message, author, date, filename, nodekind, action. Descriptions and examples of these attributes are available in Table 12 of Annex B.

Jenkins Connector

The Jenkins connector is configured by a similar set of configuration files (*connect-jenkins.properties*, *connect-jenkins-source.properties*, *connect-jenkins-elasticsearch.properties*) as the aforementioned SVN connector with similar content, we leave out the details for brevity. Details are available in Annex B.

After a successful run of the Jenkins connector, the data is available in Elasticsearch/Kibana:

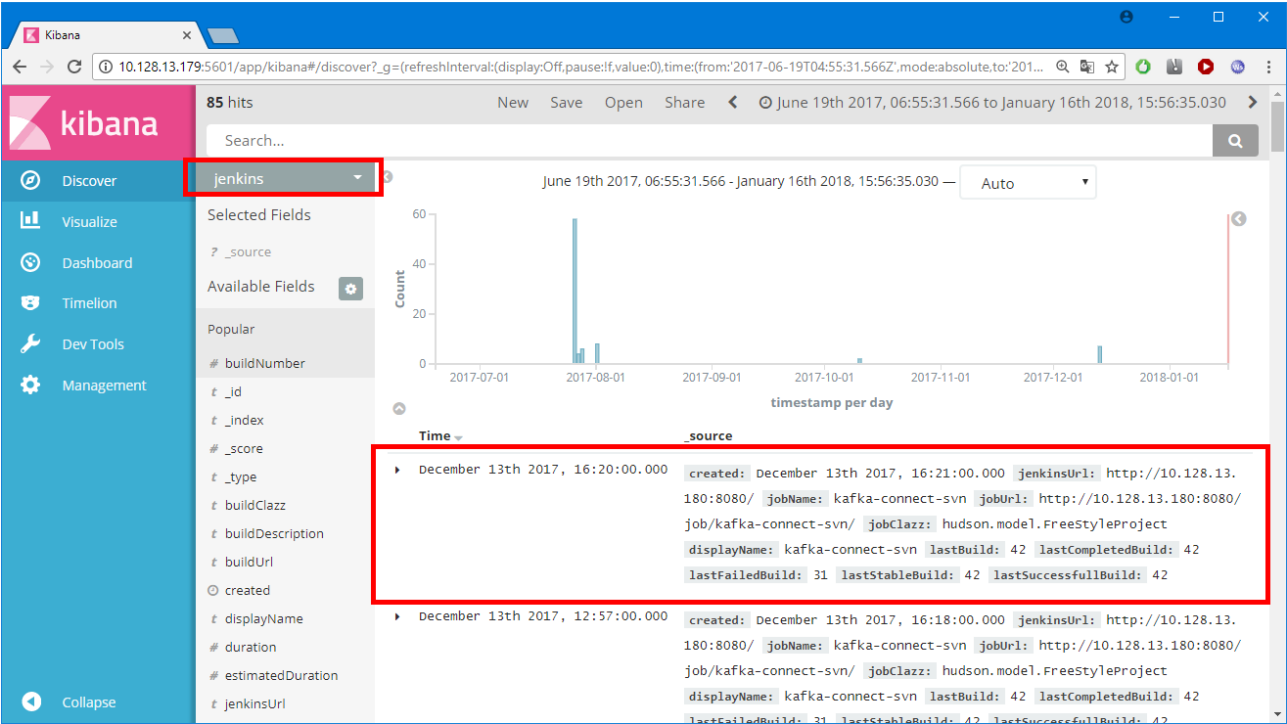


Figure 10. Test data gathered by the connect-jenkins connector and stored in the jenkins elastic search index.

The data collected by the Jenkins connector has the following attributes: created, jenkinsUrl, jobName, jobUrl, jobClazz, displayName, lastBuild, lastCompletedBuild, lastFailedBuild, lastStableBuild, lastSuccessfulBuild, lastUnstableBuild, lastUnsuccessfulBuild, buildNumber, buildUrl, buildDescription, duration, estimatedDuration, result, timestamp, testsFail, testsPass, testsSkip, testDuration. Descriptions and examples of these attributes are available in Table 13 of Annex B.





## Sonarqube Connector

The sonarqube connector collects two different datasets from sonarqube: measures obtained via static code analysis and issues (i.e., quality rules violations). The measures consists of typical size and complexity metrics like lines-of-code, cyclomatic complexity, comment-density and so on. The issue data (i.e., quality rule violations, which are different from the issues of issue tracking systems) identifies pieces of code that break configured coding rules along with the severity of the rule violation (i.e. BLOCKER, CRITICAL, MAYOR, MINOR, INFO).

After a successful run of the connector, the two datasets are available in Elasticsearch/Kibana:

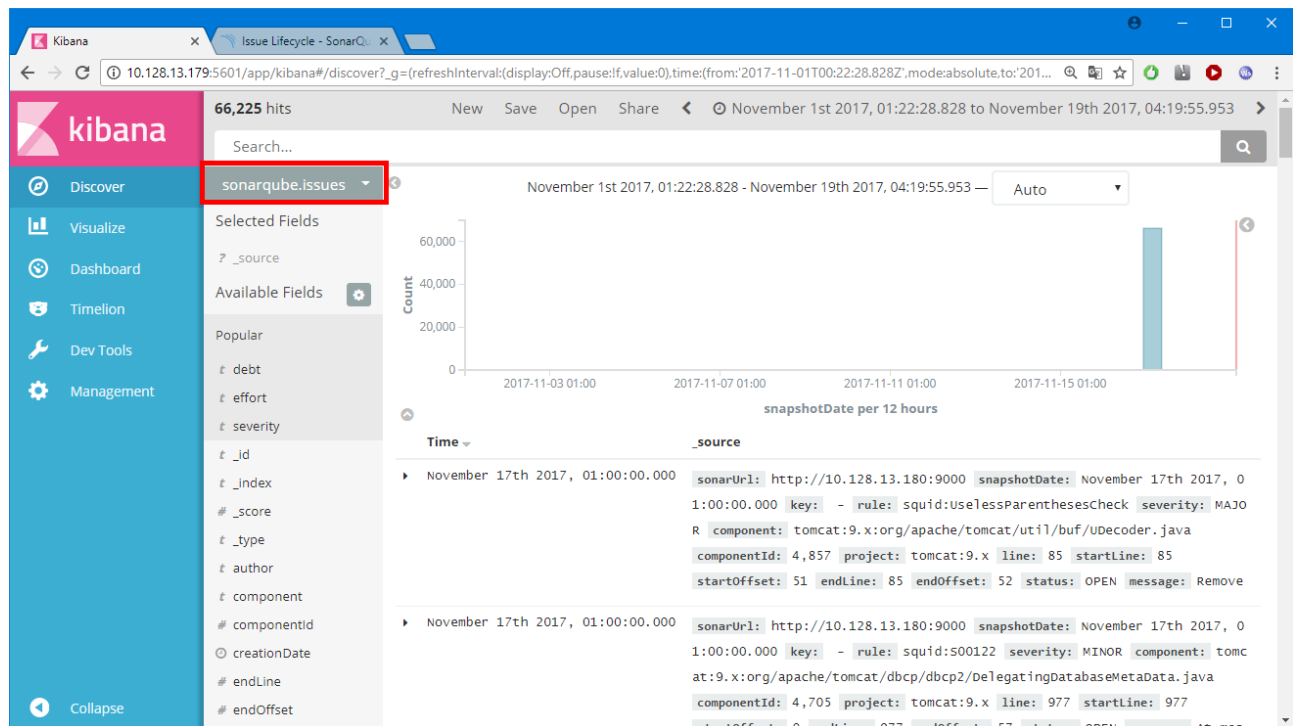


Figure 11. Issues (i.e., quality rules violations) data gathered by the connect-sonarqube connector and stored in the sonarqube.issues elastic search index.

The format of the metrics data collected contains the following attributes: sonarUrl, snapshotDate, bclid, bcKey, bcName, bcQualifier, Id, key, name, qualifier, language, metric, value, floatValue. The attribute metric consist of the name of the metric, such as cyclomatic complexity, comments ratio, and duplications ratio. Descriptions and examples of these attributes are available in Table 16 of Annex B.



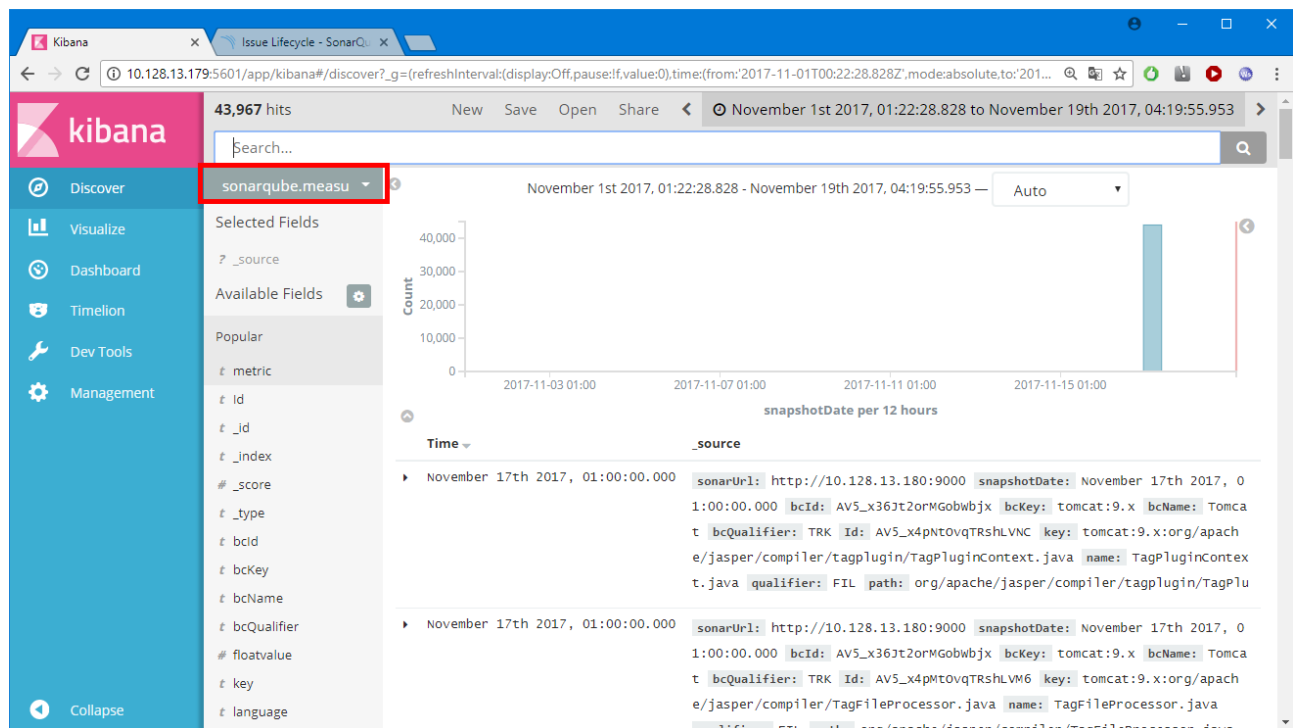


Figure 12. Static code analysis measures data gathered by the connect-sonarqube connector and stored in the sonarqube.measures elastic search index.

The format of the issues data collected contains the following attributes: sonarUrl, snapshotDate, rule, severity, component, componentId, project, line, startLine, startOffset, endLine, endOffset, effort, debt, author, creationDate. Descriptions and examples of these attributes are available in Table 15 of Annex B.

Jira Connector

The connector collects data from a Jira issue tracking system. After a successful run of the Jenkins connector, the data is available in Elasticsearch/Kibana:

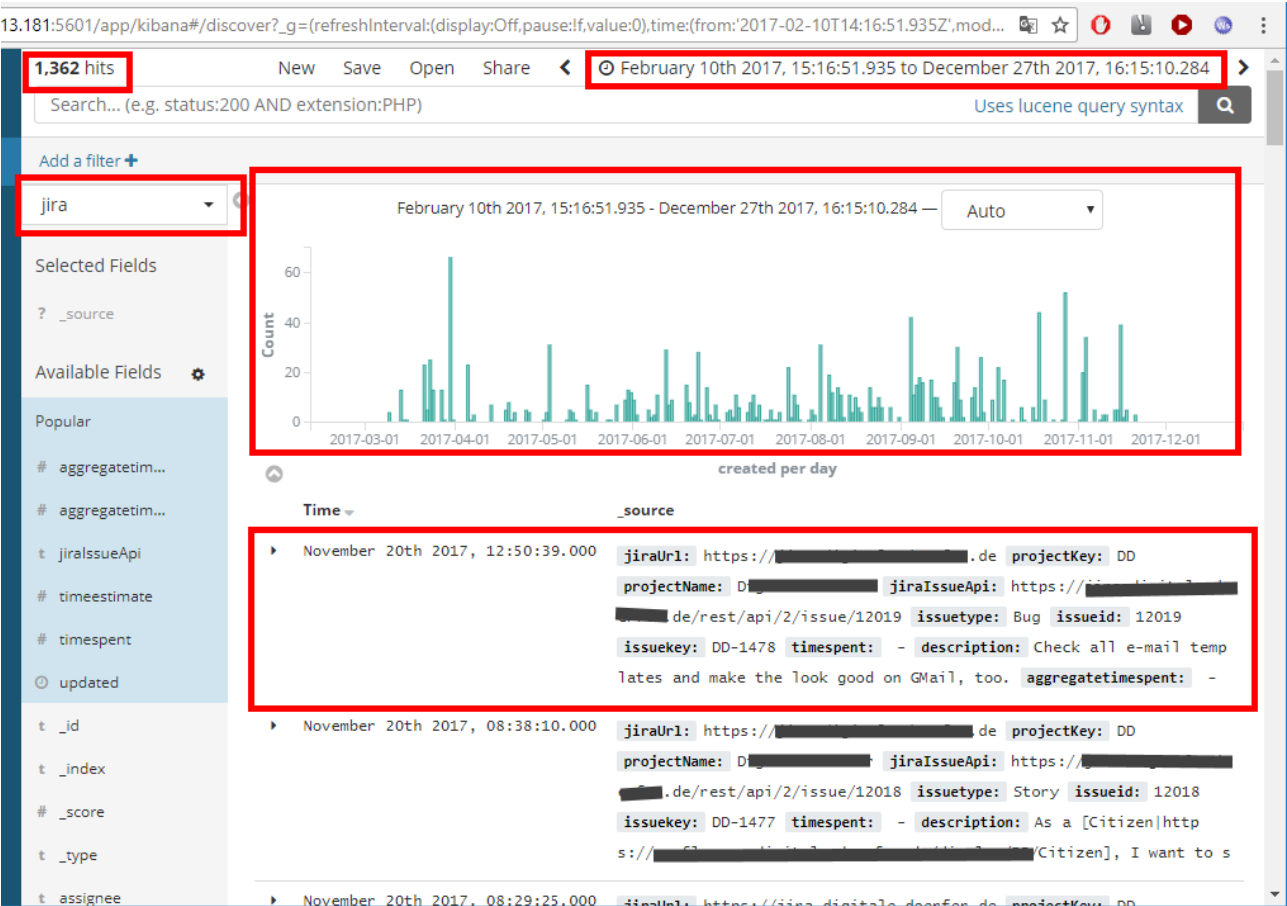


Figure 13. Issues (i.e., tasks) data gathered by the connect-jira connector and stored in the jira elastic search index.

The data format is as follows: jiraUrl, projectKey, projectName, jiraIssueApi, issuetype, timespent, description, aggregatetimespent, resolution, aggregatetimeestimate, resolutiondate, summary, lastViewed, creator, created, reporter, priority, timeestimate, duedate, assignee, updated, status. Descriptions and examples of these attributes are available in Table 14 of Annex B.

Redmine Connector

In case they are using other issue tracking connector than Jira, the qr-connect module has other connectors. This connector collects data from a Redmine issue tracking system. After a successful run of the Redmine connector, the data is available in Elasticsearch/Kibana:

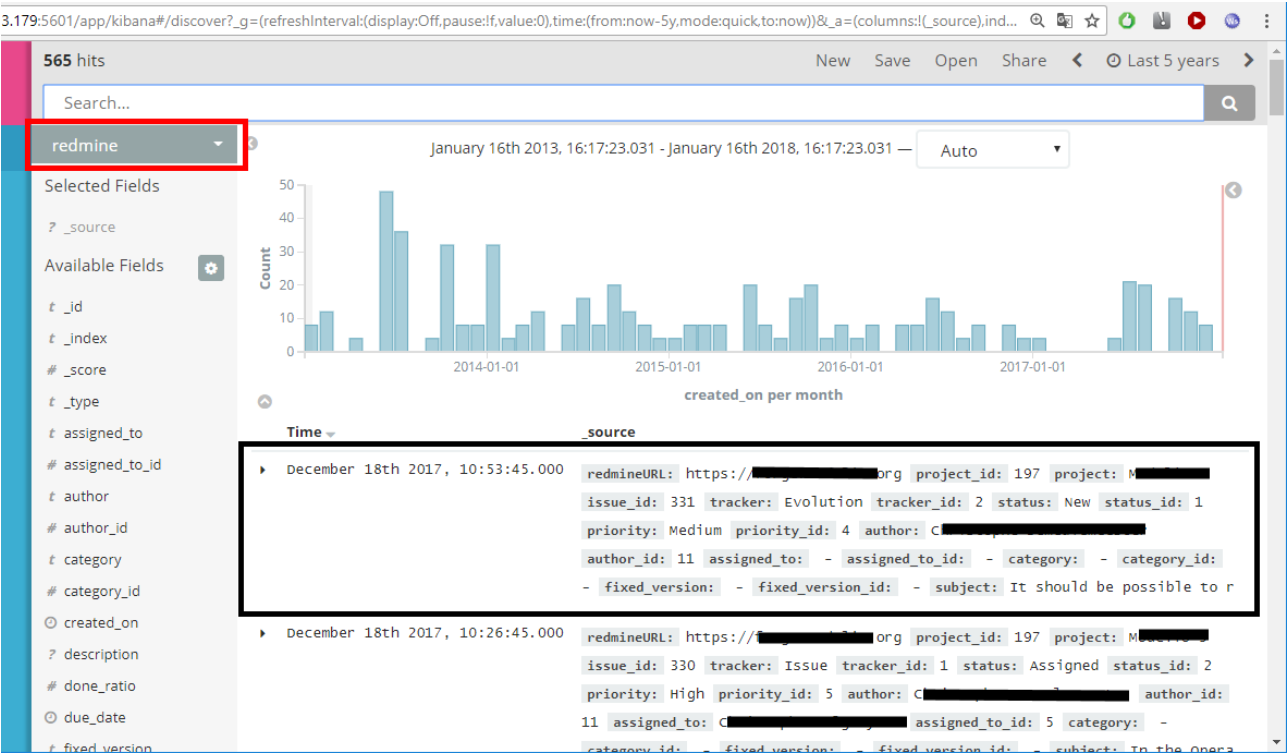


Figure 14. Issues (i.e., tasks) data gathered by the connect-redmine connector and stored in the redmine elastic search index.

The data format is as follows: redmineURL, project\_id, project, issue\_id, tracker, tracker\_id, status, status\_id, priority, priority\_id, author, author\_id, assigned\_to, assigned\_to\_id, fixed\_version, fixed\_version\_id, subject, start\_date, done\_ratio, created\_on, updated\_on. Descriptions and examples of these attributes are available in Table 17 of Annex B.



## GitLab connector

For the GitLab connector, in the ITTI use case, the data is fed on a daily basis to the Elasticsearch platform. The data are stored under the 'gitlab.stats.<project name>' index. The schema of the index is shown below (Figure 15):

```
1  {
2    "gitlab.stats.esa-otx" : {
3      "aliases" : { },
4      "mappings" : {
5        "stat" : {
6          "properties" : {
7            "bug" : {
8              "type" : "long"
9            },
10           "close" : {
11             "type" : "long"
12           },
13           "delay" : {
14             "type" : "long"
15           },
16           "evertested" : {
17             "type" : "long"
18           },
19           "inbacklog" : {
20             "type" : "long"
21           },
22           "inprogress" : {
23             "type" : "long"
24           },
25           "open" : {
26             "type" : "long"
27           },
28           "ready" : {
29             "type" : "long"
30           },
31           "test" : {
32             "type" : "long"
33           },
34           "testfailure" : {
35             "type" : "long"
36           },
37           "tstamp" : {
38             "type" : "date"
39           }
40         }
41       }
42     },
43   }
44 }
```

Figure 15. The schema of gitlab.stats.<project name> index



## 4.2 Demonstration of Data Analysis User Stories

After the collection of raw data from diverse data producers (i.e., an issue tracking system, static source code analysis, continuous build systems, and version control), the qr-eval command-line tool computes assessed metrics, product factors, and strategic indicators from the raw data as explained in Figure 5. The qr-eval reads the indexes as it set up in the index.properties file (see Annex B). Then, the assessed metrics, product factors, and strategic indicators are computed by qr-eval and stored three Elasticsearch indices (poc.metrics, poc.factors, poc.strategic\_indicators). From there, the strategic dashboard reads the data to display several analysis views to the user.

### Assessed Metrics

The quality model is contained in a set of configuration files (metrics.properties, factors.properties, and indicator.properties). In a first step, a fixed set of metrics is computed from the raw data collected by the qr-connect module. All metrics are normalized to the interval [0..1], where 1 stands for a good and desirable value and 0 for the opposite. The properties of the collected metrics are defined within the metrics.properties file. In this file, metrics computation can be turned on and off (depending on the available raw data) and other properties like metric thresholds are defined. For instance, for the “ratio of open/in progress bugs”, it should be specified if this is calculated from data in Jira or Redmine.

Table 7. An excerpt of the metrics.properties file. An example for “non-complex files” and “commented files”.

```
# computed from sonarqube measures index
complexity.enabled=true
complexity.name=Complexity
complexity.description=Percentage of files that do not exceed a defined
average complexity per function
complexity.factors=codequality
complexity.threshold.upper=15
complexity.threshold.lower=1
complexity.source=function_complexity

# computed from sonarqube measures index
comments.enabled=true
comments.name=Comment Ratio
comments.description=Percentage of files lying within a defined range of
comment density
comments.factors=codequality
comments.threshold.upper=30
comments.threshold.lower=10
comments.source=comment_lines_density
```

The computed metrics are stored in the poc.metrics index. For instance, Figure 16 is showing the index with a filter: only the “Complexity” assessed metric is shown.

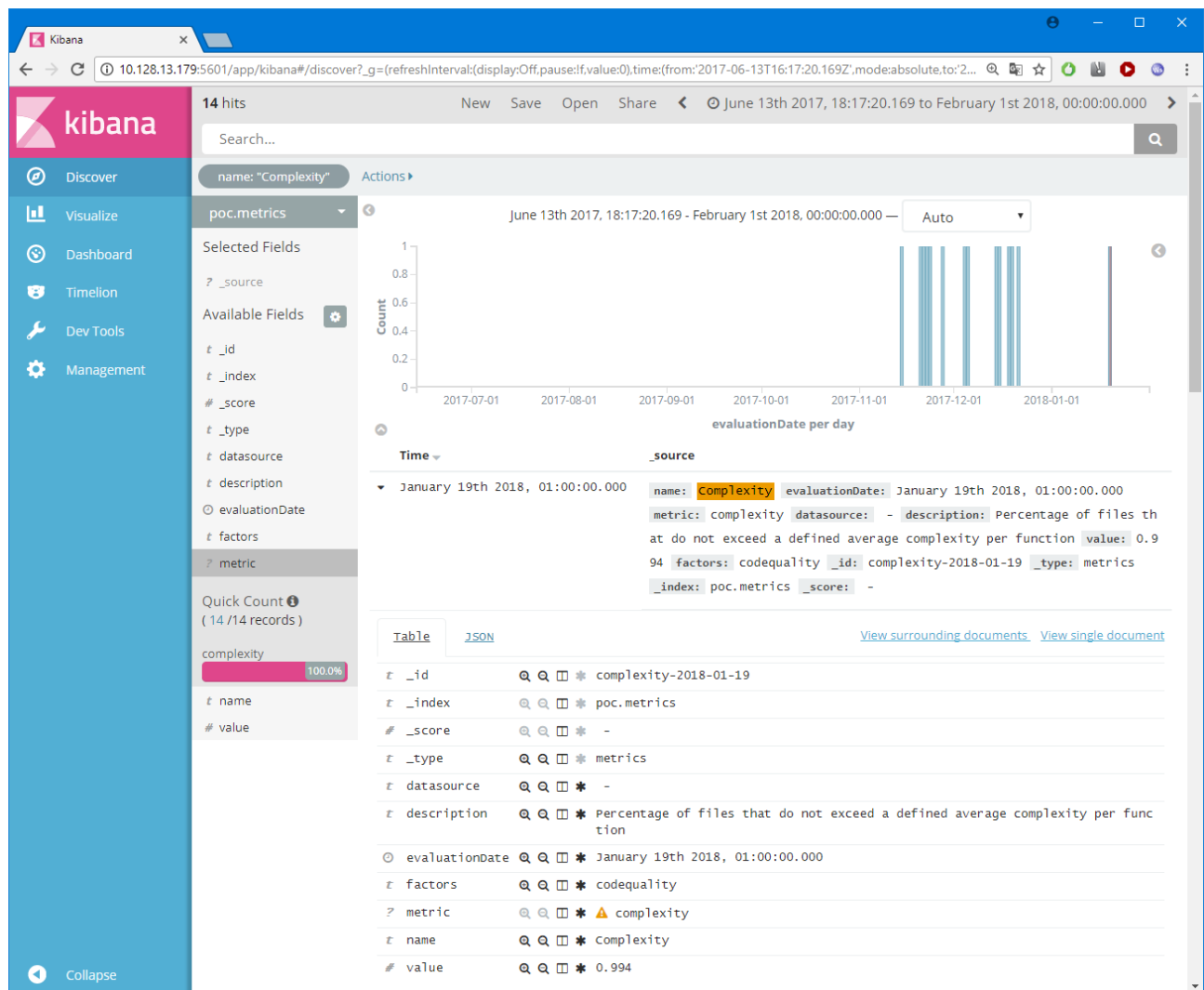


Figure 16. The assessed metrics from quality model assessments: the poc.metrics elastic search index.

## Product Factors

In a second step, qr-eval computes the factors defined within the factors.properties file.

Table 8. The factors.properties file, containing the product factors.

```
codequality.enabled=true
codequality.name=Code Quality
codequality.description=It measures the impact of code changes in source code
quality. Specifically, it is an aggregation of the metrics after static code
analysis in the specified evaluation date.
codequality.strategic_indicators=productquality

blockingcode.enabled=true
blockingcode.name=Blocking Code
blockingcode.description=Technical debt in software code in terms of rule
violations
blockingcode.strategic_indicators=blocking

qualityissuespecification.enabled=true
qualityissuespecification.name=Quality Issue Specification
qualityissuespecification.description=Completeness of Issue Specification
```



```
qualityissuespecification.strategic_indicators=blocking

softwarestability.enabled=true
softwarestability.name=Software stability
softwarestability.description=Completeness of Issue Specification
softwarestability.strategic_indicators=productquality

testingstatus.enabled=true
testingstatus.name=Testing Status
testingstatus.description=Quality and stability of executed tests
testingstatus.strategic_indicators=blocking,product quality
```

For each product factor, qr-eval computes an average of all metrics influencing the factor stores the result in the Elasticsearch index poc.factors:

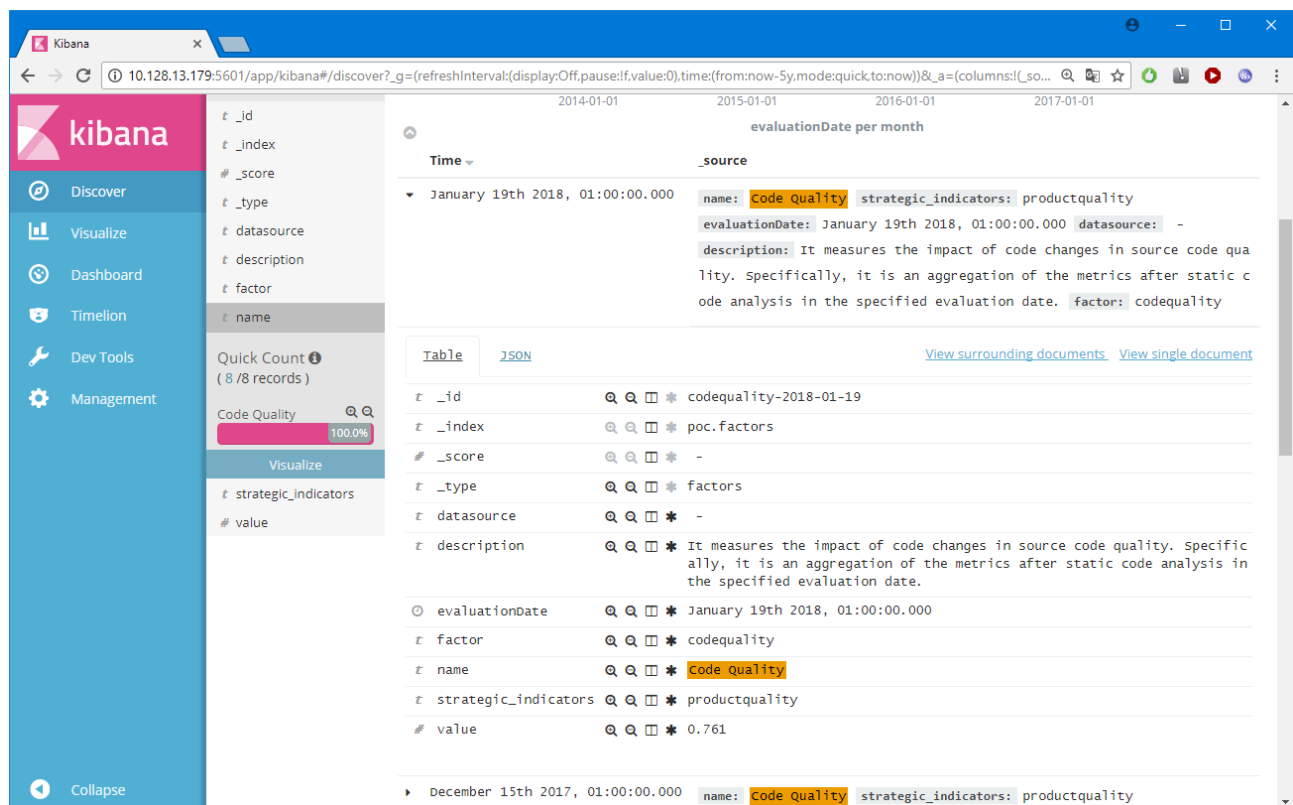


Figure 17. The product factors from quality model assessments: the poc.factors elastic search index.



## Strategic Indicators

After the product factors computation, qr-eval computes the strategic indicators defined in the `indicators.properties` file:

Table 9. An excerpt of the `indicator.properties` file, containing the strategic indicators.

```
productquality.enabled=true
productquality.name=Product Quality
productquality.description=It refers to how a product meets code quality,
testing status, and software stability.

blocking.enabled=true
blocking.name=Blocking
blocking.description=It informs that some condition has been detected
influencing negatively in the progress of the regular workflow.
```

As with the factor computation, the value of an indicator computes as the average of all factor values influencing the indicator. The resulting indicators are stored in the `poc.strategic_indicators` index:

The screenshot shows the Kibana interface with a search query. The left sidebar contains navigation links: Discover, Visualize, Dashboard, Timelion, Dev Tools, and Management. The main content area displays search results for the query. The results are shown in a table view, with columns for \_id, \_index, \_score, \_type, datasource, description, evaluationDate, name, strategic\_indicator, and value. The results are grouped by time, showing two entries for January 19th 2018, 01:00:00.000. The first entry is for the 'productquality' indicator, and the second is for the 'blocking' indicator. Both indicators have a value of 0.559.

Time	_source
January 19th 2018, 01:00:00.000	<pre> strategic_indicator: productquality evaluationDate: January 19th 2018, 01:00:00.000 datasource: - name: Product Quality description: "Quality of the Product" value: 0.559 _id: productquality-2018-01-19 _type: strategic_indicators _index: poc.strategic_indicators _score: - </pre>
January 19th 2018, 01:00:00.000	<pre> strategic_indicator: blocking evaluationDate: January 19th 2018, 01:00:00.000 datasource: - name: Blocking description: Stops us from making progress value: 0.556 _id: blocking-2018-01-19 _type: strategic_indicators _index: poc.strategic_indicators _score: - </pre>

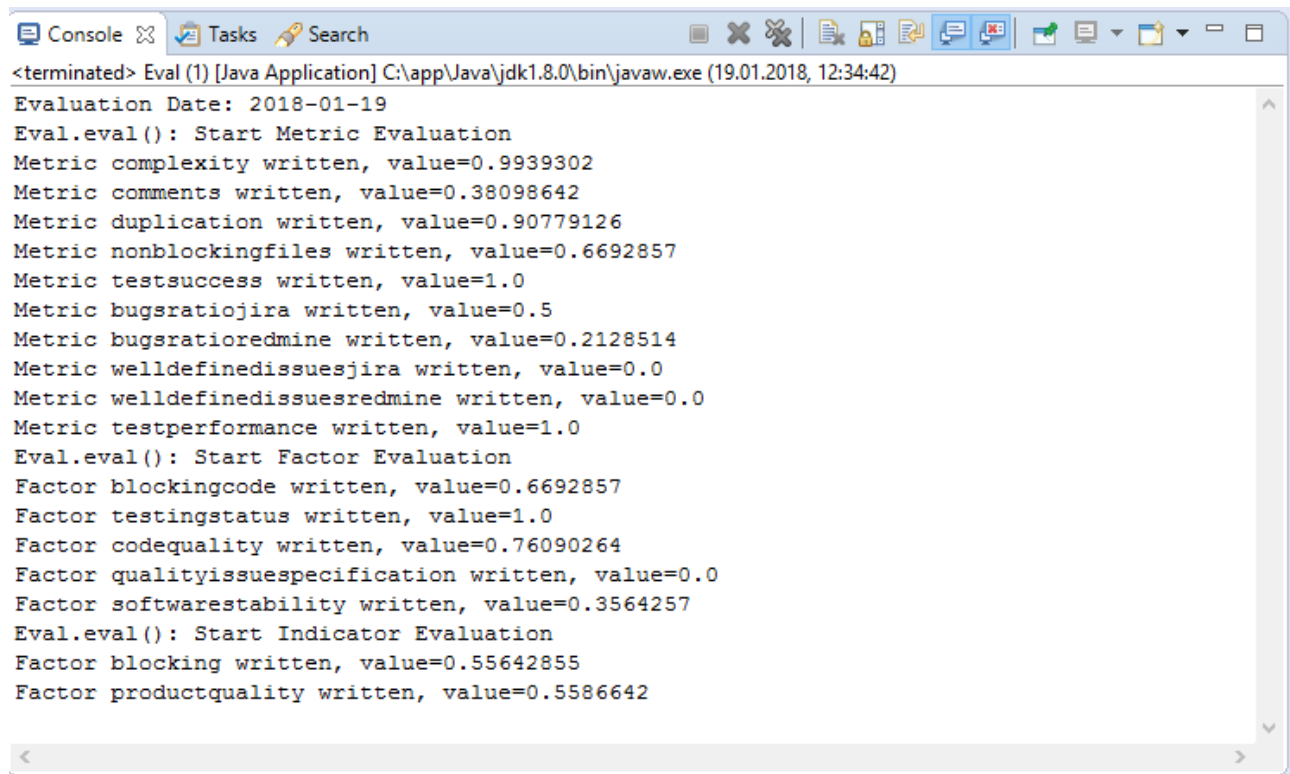
Figure 18. The strategic indicators from quality model assessments: the `poc.strategic_indicators` elastic search index.





## Performing the quality model assessment

The qr-eval command-line tool reports the results of the quality model execution in the console or log-file shown in Figure 19.



```
<terminated> Eval (1) [Java Application] C:\app\Java\jdk1.8.0\bin\javaw.exe (19.01.2018, 12:34:42)
Evaluation Date: 2018-01-19
Eval.eval(): Start Metric Evaluation
Metric complexity written, value=0.9939302
Metric comments written, value=0.38098642
Metric duplication written, value=0.90779126
Metric nonblockingfiles written, value=0.6692857
Metric testsuccess written, value=1.0
Metric bugsratiojira written, value=0.5
Metric bugsratiojira written, value=0.2128514
Metric welldefinedissuesjira written, value=0.0
Metric welldefinedissuesredmine written, value=0.0
Metric testperformance written, value=1.0
Eval.eval(): Start Factor Evaluation
Factor blockingcode written, value=0.6692857
Factor testingstatus written, value=1.0
Factor codequality written, value=0.76090264
Factor qualityissuespecification written, value=0.0
Factor softwarestability written, value=0.3564257
Eval.eval(): Start Indicator Evaluation
Factor blocking written, value=0.55642855
Factor productquality written, value=0.5586642
```

Figure 19. Example of an execution of a quality model assessment for the day 19.01.2018.

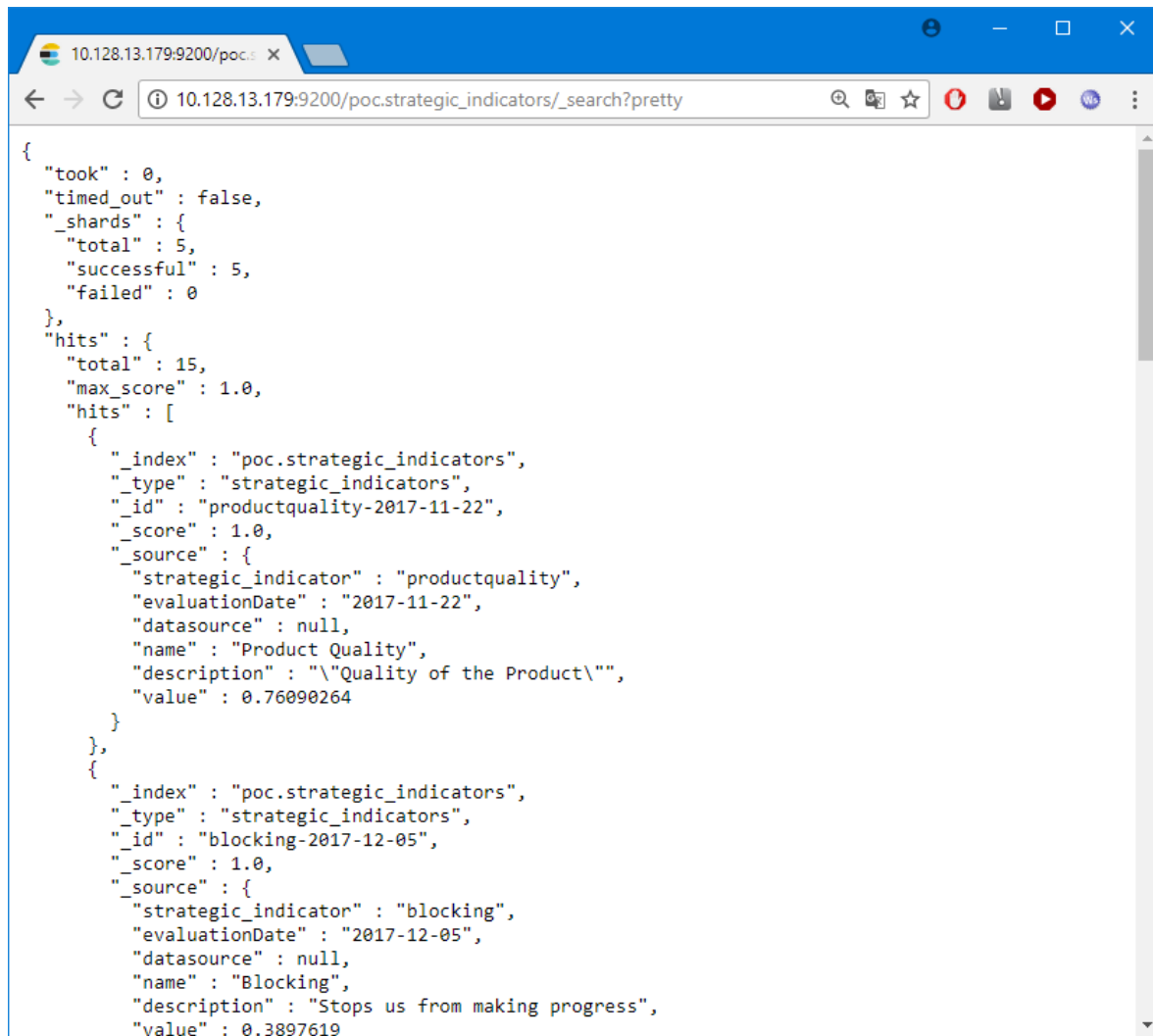
Currently, the aggregation is done daily, based on the data for that day. The unique id of any of the assessed metrics/product factors/strategic indicators stored is located in the default Elasticsearch id: "\_id". The id consist of the name of the element (e.g., productquality), and the date (e.g., 2018-01-19). Several examples are shown in:

- Figure 16 for assessed metric: complexity-2018-01-19.
- Figure 17 for product factor: codequality-2018-01-19.
- Figure 18 for strategic indicator: productquality-2018-01-19.

## Elasticsearch API: Access

In the same way that the qr-connect connectors access the data producers, other modules (e.g., the strategic dashboard) can access the aforementioned indexes containing the quality model assessment in three ways:

1. JSON API by following the Kibana or Elasticsearch URLs (see Figure 20):
  - For instance: Elasticsearch Search API: [http://<ELK\\_URL>:9200/<index-name>/\\_search?q=<query>&pretty](http://<ELK_URL>:9200/<index-name>/_search?q=<query>&pretty)
2. Embed the connection to the API in Java.
3. Using the dev tools of Kibana:  
<https://www.elastic.co/guide/en/elasticsearch/reference/current/search.html>



```
{
  "took" : 0,
  "timed_out" : false,
  "_shards" : {
    "total" : 5,
    "successful" : 5,
    "failed" : 0
  },
  "hits" : {
    "total" : 15,
    "max_score" : 1.0,
    "hits" : [
      {
        "_index" : "poc.strategic_indicators",
        "_type" : "strategic_indicators",
        "_id" : "productquality-2017-11-22",
        "_score" : 1.0,
        "_source" : {
          "strategic_indicator" : "productquality",
          "evaluationDate" : "2017-11-22",
          "datasource" : null,
          "name" : "Product Quality",
          "description" : "\"Quality of the Product\"",
          "value" : 0.76090264
        }
      },
      {
        "_index" : "poc.strategic_indicators",
        "_type" : "strategic_indicators",
        "_id" : "blocking-2017-12-05",
        "_score" : 1.0,
        "_source" : {
          "strategic_indicator" : "blocking",
          "evaluationDate" : "2017-12-05",
          "datasource" : null,
          "name" : "Blocking",
          "description" : "Stops us from making progress",
          "value" : 0.3897619
        }
      }
    ]
  }
}
```

Figure 20. Accessing the quality model assessment via Elasticsearch Search API.

#### Relations between indexes:

- Strategic indicators and product factors: To get all factors relevant for a strategic indicator, take the value of the attribute "strategic\_indicator" (e.g. "productquality") and query all factors that have the value "productquality" in the array "strategic\_indicators" and restrict to the evaluationDate. Example: [http://ELASTIC\\_URL:9200/poc.metrics/\\_search?q=factors:codequality+AND+evaluationDate:2017-11-14](http://ELASTIC_URL:9200/poc.metrics/_search?q=factors:codequality+AND+evaluationDate:2017-11-14)
- Product factors and assessed metrics: To get all metrics relevant for a factor, take the value of the attribute "factor" (e.g. "codequality") and query all metrics that have the value "codequality" in the array "factors" and restrict to the evaluationDate. Example: [http://ELASTIC\\_URL:9200/poc.metrics/\\_search?q=factors:codequality+AND+evaluationDate:2017-11-14](http://ELASTIC_URL:9200/poc.metrics/_search?q=factors:codequality+AND+evaluationDate:2017-11-14)



## 5. Data Gathering and Analysis Software and Installation

This section contains the links to download the delivered software and briefly reports the development and communication around the Q-Rapids GitLab.

### 5.1 Q-Rapids Data Gathering and Analysis Modules: Download

The *qr-connect* and *qr-eval* modules have the following prerequisites:

- **Java SE:** The data gathering and analysis tools (*qr-connect*, *qr-eval*) are based on Java. An actual version of the Java SE (version  $\geq 1.8$ ) is required.
- **Apache Kafka:** The raw data collected from the source systems is stored in Apache Kafka. For the proof-of-concept, the confluent OSS version 3.2.1 is required.
- **Elasticsearch:** For data analysis (assessed metrics, product factors, and strategic indicators computation), an actual version of the Elastic stack (version  $\geq 5.4$ ) is required.
- **Kibana (optional):** Kibana is used to view data stored in Elasticsearch and to build basic dashboards. The installation is recommended.

The *qr-connect* and *qr-eval* modules for the proof-of-concept are provided as a zip archive files in public links in the Q-Rapids basecamp:

- *qr-connect*:  
<https://public.3.basecamp.com/p/fXhyhinHW23apFM7xyEbtNsQ>
- *qr-eval*:  
<https://public.3.basecamp.com/p/EBYGXZEqBExrVoCm6XnjDhnd>
- source code: <https://public.3.basecamp.com/p/441DaJqkY6YfXoDRjUEqtzUR>

See Annex B for complete and detailed documentation for installation and use of the *qr-connect* and *qr-eval* modules.

### 5.2 Development, Communication and Support among Q-Rapids Partners

During the proof-of-concept, we have kept updated the source code in our GitLab repository<sup>7</sup>. The source code is available to all vested 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 [q-rapids-pilots-technical@essi.upc.edu](mailto:q-rapids-pilots-technical@essi.upc.edu) to communicate the releases of *qr-connect* and *qr-eval* together with release notes. In this e-mail list, there are responsible people for the deployment of the Q-Rapids tool in all partners.

Regarding support, use cases' technicians have reported technical problems with the installation and deployment of the data gathering and analysis by creating issues in our GitLab repository. We have dealt with these issues and run technical telco meetings with every industrial partner to support the Q-Rapids tool deployment before the proof-of-concept evaluation.

---

<sup>7</sup> Q-Rapids GitLab repository: [http://193.142.112.102/users/sign\\_in](http://193.142.112.102/users/sign_in)  
Issues communication: <http://193.142.112.102/root/Proof-of-Concept/issues>



## Conclusion

This document reports a demonstration on the modules of the Q-Rapids tool for data gathering and analysis. The main functionalities resolved at the proof-of-concept are:

- Connectors (from the *qr-connect* module of the Q-Rapids tool) to gather data from:
  - Static code analysis measures: SonarQube.
  - Quality rules: SonarQube.
  - Tests: Jenkins, GitLab.
  - Issues: Redmine, Jira, GitLab.
  - Commits: SVN.
- Automatic execution of the quality model assessment (the *qr-eval* module of the Q-Rapids tool) for:
  - Product quality (based on code quality, testing status, and software stability).
  - Blocking (based on blocking code, issues' specification, testing status, testing performance).

Next steps include the implementation of connectors to gather data at runtime, generation of quality requirements, the implementation of process factors, and the correlation of multiple data sources for data analysis.



## Annex A – Quality Model for the Proof-of-Concept

This annex provides a short explanation of the quality model implemented in the proof-of-concept: *Product Quality* and *Blocking* strategic indicators.

Note: all the thresholds below can be easily customized in each company.

### Blocking

It refers to conditions that negatively affects the progress of your project workflow. Blocking situations are e.g. stop-the-line, postpone a feature, ... .

Product factors:

- **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.
- **Testing Status:** It measures the quality and stability level of executed tests during development.
- **Test performance:** It measures the time consumed for the execution of tests.

Table 10. Quality model for the proof-of-concept: blocking strategic indicator.

Product Factor	Assessed Metric	Description	Calculation	Data source
Blocking code	<b>Fulfilment of critical/blocker quality rules</b>	Files fulfilling the critical or blocker quality rules violations threshold	$\text{Fulfilment of critical/blocker quality rules} = \frac{\text{Files with critical or blocker issues}}{\text{Total number of files}}$	SonarQube
Quality Issues' specification	<b>Issues completely specified</b>	Density of incomplete issues in a timeframe	$\text{Issues completely specified} = \frac{\text{Issues completely specified}}{\text{Issues}}$ where an issue is completely specified if the fields {<description>, <due date>} are not empty	JIRA, Redmine, GitLab
Testing Status	<b>Passed tests</b>	Unit test success density	$\text{Passed tests} = \frac{(\text{Unit tests} - (\text{Unit test errors} + \text{Unit test failures}))}{\text{Unit tests}}$	Jenkins
Test performance	<b>Fast tests' builds</b>	Tests' builds under the threshold of duration	$\text{Density of fast tests' builds} = \frac{\text{Fast unit test}}{\text{Unit tests}}$ where an unit test is fast if its duration is lower than <5 minutes>	Jenkins



## Product Quality

It refers here to the maintainability, reliability, and functional suitability of your software product.

Product factors:

- **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.

Table 11. Quality model for the proof-of-concept: product quality strategic indicator.

Product Factor	Assessed Metric	Description	Calculation	Data source
Code Quality	<b>Non-complex files</b>	Files under the threshold of complexity	$\text{Density of non – complex files} = \frac{\text{Non – Complex files}}{\text{Total number of files}}$ <p>where a file is complex if the &lt;cyclomatic complexity&gt; per &lt;function&gt; is greater than &lt;15&gt;.</p>	SonarQube
	<b>Commented files</b>	Files exceeding the comments percentage threshold	$\text{Density of commented files} = \frac{\text{Commented files}}{\text{Total number of files}}$ <p>where a file is commented if the &lt; density of comment lines&gt; is between 10% and 30%</p> $\text{Density of comment lines} = \frac{\text{Comment lines}}{(\text{Lines of code} + \text{Comment lines})}$	SonarQube
	<b>Absence of duplications</b>	Files under the duplicated lines percentage threshold	$\text{Absence of duplications} = \frac{\text{Files without duplications}}{\text{Total number of files}}$ <p>where a file has no duplications if the &lt; density of duplication&gt; is less than 5%</p> $\text{Density of duplication} = \frac{\text{Duplicated lines}}{\text{Lines}}$	SonarQube
Testing Status	<b>Passed tests</b>	Unit test success density	$\text{Passed tests} = \frac{(\text{Unit tests} - (\text{Unit test errors} + \text{Unit test failures}))}{\text{Unit tests}}$	Jenkins, GitLab



Software Stability	<b>Ratio of open/in progress bugs</b>	Ratio of open issues of type bug with respect to the total number of issues in a customized time frame (e.g., current/last month or current/last sprint)	Ratio of bugs = $\frac{\text{Number of open issues/in progress/re – opened of type "bug"}}{\text{Number of open issues/in progress/re – opened}} * 100$	JIRA, Redmine, GitLab
--------------------	---------------------------------------	--	---	-----------------------



## Annex B – Data Gathering and Analysis Modules Installation and Use (Axel)

This annex includes:

1. How to deploy the Q-Rapids Source Data Connectors: the qr-connect module.
2. How to define the quality model indexes in the Elastic stack.
3. How to customise the quality model and perform the quality model assessment: the qr-eval-module.

### How to deploy the Q-Rapids Source Data Connectors: the qr-connect module

#### Prerequisites

The Q-Rapids source data connectors consist of a set of Apache Kafka connectors<sup>8</sup> that allows for storing data from various sources (Jenkins, Jira, sonarqube, and svn) into an Apache Kafka Server and from there into an Elasticsearch server. The connectors are based on the Confluent Kafka Connect Framework and are developed in Java. An already available Kafka connector for Elasticsearch<sup>9</sup> allows transferring the data into an Elasticsearch Cluster to allow for searching and querying the collected data. The prerequisites to deploy qr-connect are:

- A Linux Server with Oracle Java SE 8 installed
- Elasticsearch and Kibana installed (tested with version 5.4). The setup instructions for Elasticsearch are located at <https://www.elastic.co/guide/en/elasticsearch/reference/current/setup.html> while the instructions for Kibana are located at <https://www.elastic.co/guide/en/kibana/current/setup.html>.
- Confluent Open Source Kafka framework (tested with version 3.2.1). The installation instructions for the Confluent Platform are found at <https://docs.confluent.io/current/installation/index.html>

#### QR-Connect Framework

The qr-connect framework consists of an executable jar file containing the Apache Kafka Connectors, a set of configuration files that configure the options for the data collection (e.g. ip-addresses of the source systems), and a set of schema files for Elasticsearch that define the datatypes of the target indices.

Before running any connector, make sure that

- The Elasticsearch server is up and running. Check with a browser that the address `<your-elasticsearch-server-ip>:9200` shows something like

```
{
  "name" : "Mu_73pu",
  "cluster_name" : "elasticsearch",
  "cluster_uuid" : "GbvwOdsAKub4595F0WHg",
  "version" : {
    "number" : "5.4.0",
    "build_hash" : "780f8c4",
    "build_date" : "2017-04-28T17:43:27.229Z",
    "build_snapshot" : false,
    "lucene_version" : "6.5.0"
  },
```

<sup>8</sup> <https://www.confluent.io/product/connectors/>

<sup>9</sup> <https://www.elastic.co/de/products/elasticsearch>





- ```
    "tagline" : "You Know, for Search"
  }

```
- The Kibana Server is up and running Check with a browser that the Kibana GUI is displayed at `<your-elasticsearch-server-ip>:5601`
  - The Kafka Server is up and running. You have to start zookeeper first:  
`<Apache-Kafka-Dir>/bin/zookeeper-server-start <zookeeper-properties-file>`  
`<Apache-Kafka-Dir>/bin/kafka-server-start <kafka-server-properties-file>`

## SVN Source Connector

The svn connector periodically polls a svn repository for new commits.

The connector collects data about commits in the following format:

Table 12. Data gathered about commits with the svn connector: attributes, descriptions, and examples.

| Attribute       | Description                                                                      | Example                                        |
|-----------------|----------------------------------------------------------------------------------|------------------------------------------------|
| <b>revision</b> | The svn revision                                                                 | 70                                             |
| <b>message</b>  | The commit message of the revision                                               | Issue XXXX: Description                        |
| <b>author</b>   | The author of the revision (svn username)                                        | <svn username>                                 |
| <b>date</b>     | UTC date                                                                         | 2015-12-02T17:39Z                              |
| <b>filename</b> | A name of a file or directory that is added, removed, or changed with the commit | /<projectname>/trunk/.../Some.java             |
| <b>nodekind</b> | Kind of element                                                                  | "file" or "dir"                                |
| <b>action</b>   | Repository action                                                                | "A" – added<br>"D" – deleted<br>"M" – modified |

The index mapping for the svn Elasticsearch index (elasticsearch.svn.schema) has to be put into Elasticsearch before the connector is run (e.g. by using Kibana Dev Tools):

```
PUT svn
{
  "mappings": {
    "svn": {
      "properties": {
        "action": { "type": "keyword" },
        "author": { "type": "keyword" },
        "date": { "type": "date" },
        "filename": { "type": "keyword" },
        "message": { "type": "text" },
        "nodekind": { "type": "keyword" },
        "revision": { "type": "long" }
      }
    }
  }
}
```



The svn source connector configuration is stored in a properties file (connect-svn-source.properties):

| Attribute                    | Description                                                                                  | Example                               |
|------------------------------|----------------------------------------------------------------------------------------------|---------------------------------------|
| <b>repository.url</b>        | The repository base URL                                                                      | https://<svn-server-ip>/repo          |
| <b>repository.path</b>       | The path is appended to the repository URL. Concrete values depend on your repository layout | /<projectname>/trunk                  |
| <b>repository.user</b>       | Authentication Username                                                                      | <Username>                            |
| <b>repository.pass</b>       | Authentication Password                                                                      | <Password>                            |
| <b>topic</b>                 | Kafka topic name for storing the svn data                                                    | svn                                   |
| <b>name</b>                  | Name of the connector                                                                        | kafka-svn-source-connector            |
| <b>connector.class</b>       | Classname of the class implementing the connector                                            | connect.svn.SubversionSourceConnector |
| <b>poll.interval.seconds</b> | Polling interval in seconds                                                                  | 60                                    |

An additional configuration file is used to define the Elasticsearch server as a data sink (connect-svn-elasticsearch.properties). Every item stored in the Kafka server is also passed to Elasticsearch.

| Attribute              | Description                                                   | Example                                                       |
|------------------------|---------------------------------------------------------------|---------------------------------------------------------------|
| <b>name</b>            | Name of the connector                                         | svn-elasticsearch                                             |
| <b>connector.class</b> | Classname of the class implementing the connector             | io.confluent.connect.elasticsearch.ElasticsearchSinkConnector |
| <b>tasks.max</b>       | Max. number of concurrent tasks                               | 1                                                             |
| <b>topics</b>          | The Kafka Topic to read, Elasticsearch index name to store to | svn                                                           |
| <b>name</b>            | Name of the connector                                         | kafka-svn-source-connector                                    |
| <b>connection.url</b>  | URL of the Elasticsearch Server                               | http://<elasticsearch-server-ip>:9200                         |
| <b>type.name</b>       | Index type name for Elasticsearch                             | svn                                                           |

One more configuration file defines the basic connector configuration (connect-svn.properties):

| Attribute                           | Description                                                                                                  | Example                          |
|-------------------------------------|--------------------------------------------------------------------------------------------------------------|----------------------------------|
| <b>bootstrap.servers</b>            | Kafka Server URL                                                                                             | <kafka-bootstrap-server-ip>:9092 |
| <b>offset.storage.file.filename</b> | Kafka offset storage, resume reading source data at the correct position                                     | /tmp/connect-svn.offsets         |
| <b>rest.port</b>                    | Rest port of the connector. If multiple connectors are run, this port has to be different for every instance | 8085                             |
| ...                                 | Standard attributes not shown                                                                                | ...                              |

Before running the connector, at least adapt the following configuration values:

- connect-svn.properties:
  - bootstrap.servers (set to your kafka server ip)



- connect-svn-source.properties:
  - repository.url (set to your svn server base url)
  - repository.path (set to a path within your repository, e.g. “/project/trunk”)
  - repository.user (valid svn username)
  - repository.pass (valid password)

To run the connector for svn, use the following command within the qr-connect directory (put into one line):

```
CLASSPATH=qr-connect-0.0.2-jar-with-dependencies.jar
<kafka-install-dir>/bin/connect-standalone
<qr-connect-install-dir>/connect-svn.properties
<qr-connect-install-dir>/connect-svn-source.properties
<qr-connect-install-dir>/connect-svn-elasticsearch.properties
```

### Jenkins Source Connector

The Jenkins connector periodically polls a Jenkins server for new builds of a defined list of jobs. The connector collects data about builds in the following format:

Table 13. Data gathered about builds of projects with the Jenkins connector: attributes, descriptions, and examples.

| Attribute                    | Description                                               | Example                                                                                               |
|------------------------------|-----------------------------------------------------------|-------------------------------------------------------------------------------------------------------|
| <b>created</b>               | Timestamp of data collection UTC                          | 2017-10-12T09:38Z                                                                                     |
| <b>jenkinsUrl</b>            | URL of the Jenkins server                                 | http://<ip-address>:8080/                                                                             |
| <b>jobName</b>               | Job name                                                  | job1                                                                                                  |
| <b>jobUrl</b>                | URL of the job                                            | http://<ip-address>:8080/job/job1/                                                                    |
| <b>jobClazz</b>              | Jenkins job class                                         | hudson.model.FreeStyleProject                                                                         |
| <b>displayName</b>           | Jenkins job display name                                  | job1                                                                                                  |
| <b>lastBuild</b>             | Last build number                                         | 39                                                                                                    |
| <b>lastCompletedBuild</b>    | Last complete build number                                | 31                                                                                                    |
| <b>lastFailedBuild</b>       | Last failed build number                                  | 31                                                                                                    |
| <b>lastStableBuild</b>       | Last stable build number                                  | 39                                                                                                    |
| <b>lastSuccessfulBuild</b>   | Last successful build number                              | 39                                                                                                    |
| <b>lastUnstableBuild</b>     | Last unstable build number                                | -1                                                                                                    |
| <b>lastUnsuccessfulBuild</b> | Last unsuccessful build number                            | 31                                                                                                    |
| <b>buildNumber</b>           | Job build number                                          | 39                                                                                                    |
| <b>buildUrl</b>              | URL of the build                                          | <a href="http://&lt;ip-address&gt;:8080/job/job1/39/">http://&lt;ip-address&gt;:8080/job/job1/39/</a> |
| <b>buildDescription</b>      | Build description                                         | null                                                                                                  |
| <b>duration</b>              | Build duration                                            | 13947                                                                                                 |
| <b>estimatedDuration</b>     | Estimated build duration                                  | 11983                                                                                                 |
| <b>result</b>                | build result                                              | SUCCESS                                                                                               |
| <b>timestamp</b>             | Timestamp of build                                        | 2017-10-10T09:15Z                                                                                     |
| <b>testsFail</b>             | Number of failed tests (e.g. provided by JUnit test run)  | 0                                                                                                     |
| <b>testsPass</b>             | Number of passed tests (e.g. provided by JUnit test run)  | 3                                                                                                     |
| <b>testsSkip</b>             | Number of skipped tests (e.g. provided by JUnit test run) | 0                                                                                                     |



|                     |               |       |
|---------------------|---------------|-------|
| <b>testDuration</b> | Test Duration | 0.002 |
|---------------------|---------------|-------|

The index mapping for the jenkins Elasticsearch index (elasticsearch.jenkins.schema) has to be put into Elasticsearch before the connector is run (e.g. by using Kibana Dev Tools):

```
PUT jenkins
{
  "mappings": {
    "jenkins": {
      "properties": {
        "buildClazz": { "type": "keyword" },
        "buildDescription": { "type": "text" },
        "buildNumber": { "type": "integer" },
        "buildUrl": { "type": "keyword" },
        "created": { "type": "date" },
        "displayName": { "type": "keyword" },
        "duration": { "type": "long" },
        "estimatedDuration": { "type": "long" },
        "jenkinsUrl": { "type": "keyword" },
        "jobClazz": { "type": "keyword" },
        "jobName": { "type": "keyword" },
        "jobUrl": { "type": "keyword" },
        "lastBuild": { "type": "integer" },
        "lastCompletedBuild": { "type": "integer" },
        "lastFailedBuild": { "type": "integer" },
        "lastStableBuild": { "type": "integer" },
        "lastSuccessfulBuild": { "type": "integer" },
        "lastUnstableBuild": { "type": "integer" },
        "lastUnsuccessfulBuild": { "type": "integer" },
        "result": { "type": "keyword" },
        "testDuration": { "type": "double" },
        "testsFail": { "type": "long" },
        "testsPass": { "type": "long" },
        "testsSkip": { "type": "long" },
        "timestamp": { "type": "date" }
      }
    }
  }
}
```

The Jenkins source connector configuration is stored in a properties file (connect-jenkins-source.properties):

| Attribute              | Description                                       | Example                                |
|------------------------|---------------------------------------------------|----------------------------------------|
| <b>name</b>            | Name of the connector                             | kafka-jenkins-source-connector         |
| <b>connector.class</b> | Classname of the class implementing the connector | connect.jenkins.JenkinsSourceConnector |
| <b>jenkins.url</b>     | URL of the Jenkins server                         | http://<jenkins-server-ip>:8080/       |
| <b>jenkins.user</b>    | Authentication Username                           | <Username>                             |
| <b>jenkins.pass</b>    | Authentication Password                           | <Password>                             |



|                                 |                                                                                    |            |
|---------------------------------|------------------------------------------------------------------------------------|------------|
| <b>jenkins.topic</b>            | Kafka topic name                                                                   | jenkins    |
| <b>jenkins.jobs</b>             | List of job names to be analyzed                                                   | job1, job2 |
| <b>jenkins.lookback</b>         | When started the first time, fetch only this number of recent builds               | 100        |
| <b>jenkins.interval.seconds</b> | Poll interval in seconds                                                           | 60         |
| <b>use.preemptive.auth</b>      | Activates Nokia version with special authentication if set to true, default false. | false      |

An additional configuration file is used to define the Elasticsearch server as a data sink (connect-jenkins-elasticsearch.properties). Every Jenkins item stored in the Kafka server is also passed to Elasticsearch.

| Attribute              | Description                                                   | Example                                                       |
|------------------------|---------------------------------------------------------------|---------------------------------------------------------------|
| <b>name</b>            | Name of the connector                                         | kafka-jenkins-elasticsearch                                   |
| <b>connector.class</b> | Classname of the class implementing the connector             | io.confluent.connect.elasticsearch.ElasticsearchSinkConnector |
| <b>tasks.max</b>       | Max. number of concurrent tasks                               | 1                                                             |
| <b>topics</b>          | The Kafka Topic to read, Elasticsearch index name to store to | jenkins                                                       |
| <b>connection.url</b>  | URL of the Elasticsearch Server                               | http://<elasticsearch-server-ip>:9200                         |
| <b>type.name</b>       | Index type name for Elasticsearch                             | jenkins                                                       |

One more configuration file defines the basic connector configuration (connect-jenkins.properties):

| Attribute                           | Description                                                                                                  | Example                          |
|-------------------------------------|--------------------------------------------------------------------------------------------------------------|----------------------------------|
| <b>bootstrap.servers</b>            | Kafka Server URL                                                                                             | <kafka-bootstrap-server-ip>:9092 |
| <b>offset.storage.file.filename</b> | Kafka offset storage, resume reading source data at the correct position                                     | /tmp/connect-jenkins.offsets     |
| <b>rest.port</b>                    | Rest port of the connector. If multiple connectors are run, this port has to be different for every instance | 8086                             |
| ...                                 | Standard attributes not shown                                                                                | ...                              |

Before running the connector, at least adapt the following configuration values:

- connect-jenkins.properties:
  - bootstrap.servers (set to your kafka server ip)
- connect-jenkins-source.properties:
  - jenkins.url (set to your jenkins server url)
  - jenkins.user (valid jenkins username)
  - jenkins.pass (valid password)
  - jenkins.jobs (list of jobs to observe)

To run the connector for svn, use the following command within the qr-connect directory (put into one line):



```
CLASSPATH=qr-connect-0.0.2-jar-with-dependencies.jar
<kafka-install-dir>/bin/connect-standalone
<qr-connect-install-dir>/connect-jenkins.properties
<qr-connect-install-dir>/connect-jenkins-source.properties
<qr-connect-install-dir>/connect-jenkins-elasticsearch.properties
```

## Jira Source Connector

The Jira connector periodically polls a Jira server for issues of a defined project. The connector collects data about issues in the following format:

Table 14. Data gathered about issues with the Jira connector: attributes, descriptions, and examples.

| Attribute                    | Description                                | Example                                             |
|------------------------------|--------------------------------------------|-----------------------------------------------------|
| <b>jiraUrl</b>               | URL of the Jira server                     | http://<jira-ip>:<jira-port>                        |
| <b>projectKey</b>            | The project key                            | QRAP                                                |
| <b>projectName</b>           | The project name                           | Q-Rapids-Project                                    |
| <b>jiraIssueApi</b>          | API URL for the issue                      | http://<jira-ip>:<jira-port>/rest/api/2/issue/10102 |
| <b>issuetype</b>             | Issue type                                 | Bug                                                 |
| <b>timespent</b>             | Time spent on issue                        | null                                                |
| <b>description</b>           | Description of the issue                   | Repair this stuff.                                  |
| <b>aggregatetimespent</b>    | Time spent on issue including child issues | null                                                |
| <b>resolution</b>            | Issue resolution                           | null                                                |
| <b>aggregatetimeestimate</b> | Aggregate time estimate                    | null                                                |
| <b>resolutiondate</b>        | Date no resolution                         | null                                                |
| <b>summary</b>               | Issue summary                              | Always something wrong!                             |
| <b>lastViewed</b>            | Timestamp                                  | 2017-09-13T10:40:36.572+0200                        |
| <b>creator</b>               | User who created the issue                 | admin                                               |
| <b>created</b>               | Timestamp                                  | 2017-09-13T10:40:36.000+0200                        |
| <b>reporter</b>              | User who reported the issue                | admin                                               |
| <b>priority</b>              | Issue priority                             | Medium                                              |
| <b>timeestimate</b>          | Time estimate                              | null                                                |
| <b>duedate</b>               | Due date                                   | null                                                |
| <b>assignee</b>              | Assigned to ...                            | null                                                |
| <b>updated</b>               | Timestamp                                  | 2017-09-13T10:40:36.000+0200                        |
| <b>status</b>                | Issue status                               |                                                     |

The index mapping for the Jira Elasticsearch index (elasticsearch.jira.schema) has to be put into Elasticsearch before the connector is run (e.g. by using Kibana Dev Tools):

```
PUT jira
{
  "mappings": {
    "jira": {
      "properties": {
        "aggregatetimeestimate": { "type": "long" },
        "aggregatetimespent": { "type": "long" },
        "assignee": { "type": "keyword" },
```



```

    "created": { "type": "date" },
    "creator": { "type": "keyword" },
    "description": { "type": "text" },
    "duedate": { "type": "date" },
    "issuetype": { "type": "keyword" },
    "issueid": { "type": "keyword" },
    "issuekey": { "type": "keyword" },
    "jiraIssueApi": { "type": "keyword" },
    "jiraUrl": { "type": "keyword" },
    "lastViewed": { "type": "date" },
    "priority": { "type": "keyword" },
    "projectKey": { "type": "keyword" },
    "projectName": { "type": "keyword" },
    "reporter": { "type": "keyword" },
    "resolution": { "type": "keyword" },
    "resolutiondate": { "type": "date" },
    "status": { "type": "keyword" },
    "summary": { "type": "text" },
    "timeestimate": { "type": "long" },
    "timespent": { "type": "long" },
    "updated": { "type": "date" }
  }
}
}
}

```

The Jira source connector configuration is stored in a properties file (connect-jira-source.properties):

| Attribute                    | Description                                       | Example                          |
|------------------------------|---------------------------------------------------|----------------------------------|
| <b>name</b>                  | Name of the connector                             | kafka-jira-source-connector      |
| <b>connector.class</b>       | Classname of the class implementing the connector | connect.jira.JiraSourceConnector |
| <b>jira.url</b>              | URL of the Jira server                            | http://<jira-server-ip>:<port>/  |
| <b>jira.user</b>             | Authentication Username                           | <Username>                       |
| <b>jira.pass</b>             | Authentication Password                           | <Password>                       |
| <b>jira.topic</b>            | Kafka topic name                                  | jira                             |
| <b>jira.project</b>          | The Jira project to be analyzed                   | q-rapids                         |
| <b>jira.interval.seconds</b> | Poll interval in seconds                          | 60                               |
| <b>jira.created.since</b>    | Fetch issues created since this date              | 2017-01-01                       |

An additional configuration file is used to define the Elasticsearch server as a data sink (connect-jira-elasticsearch.properties). Every Jenkins item stored in the Kafka server is also passed to Elasticsearch.

| Attribute              | Description                                                   | Example                                                       |
|------------------------|---------------------------------------------------------------|---------------------------------------------------------------|
| <b>name</b>            | Name of the connector                                         | kafka-jira-elasticsearch                                      |
| <b>connector.class</b> | Classname of the class implementing the connector             | io.confluent.connect.elasticsearch.ElasticsearchSinkConnector |
| <b>tasks.max</b>       | Max. number of concurrent tasks                               | 1                                                             |
| <b>topics</b>          | The Kafka Topic to read, Elasticsearch index name to store to | jira                                                          |
| <b>connection.url</b>  | URL of the Elasticsearch Server                               | http://<elasticsearch-server-ip>:9200                         |



|                  |                                   |      |
|------------------|-----------------------------------|------|
| <b>type.name</b> | Index type name for Elasticsearch | jira |
|------------------|-----------------------------------|------|

One more configuration file defines the basic connector configuration (connect-jira.properties):

| Attribute                           | Description                                                                                                  | Example                          |
|-------------------------------------|--------------------------------------------------------------------------------------------------------------|----------------------------------|
| <b>bootstrap.servers</b>            | Kafka Server URL                                                                                             | <kafka-bootstrap-server-ip>:9092 |
| <b>offset.storage.file.filename</b> | Kafka offset storage, resume reading source data at the correct position                                     | /tmp/connect-jira.offsets        |
| <b>rest.port</b>                    | Rest port of the connector. If multiple connectors are run, this port has to be different for every instance | 8087                             |
| ...                                 | Standard attributes not shown                                                                                | ...                              |

Before running the connector, at least adapt the following configuration values:

- connect-jira.properties:
  - bootstrap.servers (set to your kafka server ip)
- connect-jira-source.properties:
  - jira.url (set to your jenkins server url)
  - jira.user (valid jenkins username)
  - jira.pass (valid password)
  - jira.project (projects to collect issues from)

To run the connector for jira, use the following command within the qr-connect directory (put into one line):

```
CLASSPATH=qr-connect-0.0.2-jar-with-dependencies.jar
<confluent-kafka-install-dir>/bin/connect-standalone
<qr-connect-install-dir>/connect-jira.properties
<qr-connect-install-dir>/connect-jira-source.properties
<qr-connect-install-dir>/connect-jira-elasticsearch.properties
```

## Sonarqube Source Connector

The Sonarqube connector periodically polls a Sonarqube server for measures and code issues and produces two types of records:

### Measures:

Table 15. Data gathered about measures with the SonarQube connector: attributes, descriptions, and examples.

| Attribute           | Description                                                                                                      | Example                                       |
|---------------------|------------------------------------------------------------------------------------------------------------------|-----------------------------------------------|
| <b>sonarUrl</b>     | URL of the Sonarqube server                                                                                      | <b>http://&lt;ip-address&gt;:&lt;port&gt;</b> |
| <b>snapshotDate</b> | Date record was produced (read into kafka).                                                                      | 2017-09-15T11:30Z                             |
| <b>bcId</b>         | baseComponentId in Sonarqube                                                                                     | AV5_x36Jt2orMGobWbjx                          |
| <b>bcKey</b>        | baseComponentKey in Sonarqube. This equals the “sonar.projectKey” specified in the sonar-project.properties file | tomcat:9.x                                    |





|                    |                                                                                                                                                                              |                                                  |
|--------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|--------------------------------------------------|
| <b>bcName</b>      | Project name in Sonarqube. This equals the "sonar.ProjectName" specified in the sonar-project.properties file                                                                | Tomcat                                           |
| <b>bcQualifier</b> | One of:<br><br>VW: view<br>SVW: sub-view<br>TRK: project<br>BRC: module<br>CLA: class<br>UTS: unit test<br>DIR: directory<br>FIL: file<br>DEV: developer                     | TRK                                              |
| <b>Id</b>          | Internal Sonarqube id                                                                                                                                                        | AV5_x4oatOvqTRshLU_S                             |
| <b>key</b>         | Component key                                                                                                                                                                | tomcat:9.x:javax/servlet/jsp/tagext/TagInfo.java |
| <b>name</b>        | Component name                                                                                                                                                               | TagInfo.java                                     |
| <b>qualifier</b>   | Component Qualifier. One of<br>VW: view<br>SVW: sub-view<br>TRK: project<br>BRC: module<br>CLA: class<br>UTS: unit test<br>DIR: directory<br>FIL: file<br><br>DEV: developer | FIL                                              |
| <b>language</b>    | Programming language                                                                                                                                                         | java                                             |
| <b>metric</b>      | Metric name                                                                                                                                                                  | functions                                        |
| <b>value</b>       | Textual value Representation                                                                                                                                                 | "7"                                              |
| <b>floatValue</b>  | Float value represenatatin (if applicable)                                                                                                                                   | 7.0                                              |

The index mapping for the Sonarqube Measures Elasticsearch index (elasticsearch.sonarqube.measure.schema) has to be put into Elasticsearch before the connector is run (e.g. by using Kibana Dev Tools):

```
PUT sonar.measure
{
  "mappings": {
    "sonarqube": {
      "properties": {
        "Id": { "type": "keyword" },
        "bcId": { "type": "keyword" },
        "bcKey": { "type": "keyword" },
        "bcName": { "type": "keyword" },
        "bcQualifier": { "type": "keyword" },
        "floatvalue": { "type": "float" },
        "key": { "type": "keyword" },
        "language": { "type": "keyword" },

```



```

    "metric": { "type": "keyword" },
    "name": { "type": "keyword" },
    "path": { "type": "keyword" },
    "qualifier": { "type": "keyword" },
    "snapshotDate": { "type": "date" },
    "sonarUrl": { "type": "keyword" },
    "value": { "type": "keyword" }
  }
}
}
}
}

```

## Sonarqube Code Issues:

Table 16. Data gathered about quality rules issues with the SonarQube connector: attributes, descriptions, and examples.

| Attribute           | Description                                                                                             | Example                                                            |
|---------------------|---------------------------------------------------------------------------------------------------------|--------------------------------------------------------------------|
| <b>sonarUrl</b>     | URL of the Sonarqube server                                                                             | <b>http://&lt;ip-address&gt;:&lt;port&gt;</b>                      |
| <b>snapshotDate</b> | Date record was produced (read into kafka).                                                             | 2017-09-15T11:30Z                                                  |
| <b>rule</b>         | Id of violated rule                                                                                     | squid:S00122                                                       |
| <b>severity</b>     | Severity of violation                                                                                   | MINOR                                                              |
| <b>component</b>    | Sonarqube component identifier                                                                          | tomcat:9.x:org/apache/tomcat/<br>util/net/NioBlockingSelector.java |
| <b>componentId</b>  | Sonarqube internal component id                                                                         | 5045                                                               |
| <b>project</b>      | Sonarqube project id. This equals the “sonar.projectKey” specified in the sonar-project.properties file | tomcat:9.x                                                         |
| <b>line</b>         | Code line of violation                                                                                  | 339                                                                |
| <b>startLine</b>    | Start line of violation                                                                                 | 339                                                                |
| <b>startOffset</b>  | Character offset in startLine                                                                           | 0                                                                  |
| <b>endLine</b>      | End line of violation                                                                                   | java                                                               |
| <b>endOffset</b>    | Character offset in endLine                                                                             | 60                                                                 |
| <b>effort</b>       | Estimated time to solve the issue                                                                       | 1min                                                               |
| <b>debt</b>         | Issue debt                                                                                              | 1min                                                               |
| <b>author</b>       | Issue author                                                                                            |                                                                    |
| <b>creationDate</b> | Sonarqube creation date                                                                                 | 2017-09-14T11:43:37+0200                                           |

The index mapping for the Sonarqube Measures index (elasticsearch.sonarqube.issue.schema) has to be put into Elasticsearch before the connector is run (e.g. by using Kibana Dev Tools):

PUT sonarqube.issue

```

{
  "mappings": {
    "sonarqube": {
      "properties": {
        "author": { "type": "keyword" },
        "component": { "type": "keyword" },
        "componentId": { "type": "integer" },

```



```

    "creationDate": { "type": "date" },
    "debt": { "type": "keyword" },
    "effort": { "type": "keyword" },
    "endLine": { "type": "integer" },
    "endOffset": { "type": "integer" },
    "key": { "type": "keyword" },
    "line": { "type": "integer" },
    "message": { "type": "text" },
    "project": { "type": "keyword" },
    "rule": { "type": "keyword" },
    "severity": { "type": "keyword" },
    "snapshotDate": { "type": "date" },
    "sonarUrl": { "type": "keyword" },
    "startLine": { "type": "integer" },
    "startOffset": { "type": "integer" },
    "status": { "type": "keyword" }
  }
}
}
}

```

The Sonarqube source connector configuration for both types of records is stored in a properties file (connect-sonarqube-source.properties):

| Attribute                       | Description                                                                                           | Example                                                                                                                                                                                                 |
|---------------------------------|-------------------------------------------------------------------------------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>name</b>                     | Name of the connector                                                                                 | kafka-sonar-source-connector                                                                                                                                                                            |
| <b>connector.class</b>          | Classname of the class implementing the connector                                                     | connect.sonarqube.SonarqubeSourceConnector                                                                                                                                                              |
| <b>sonar.url</b>                | URL of the Sonarqube server                                                                           | http://<sonar-ip>:<port>/                                                                                                                                                                               |
| <b>sonar.user</b>               | Authentication Username                                                                               | <Username>                                                                                                                                                                                              |
| <b>sonar.pass</b>               | Authentication Password                                                                               | <Password>                                                                                                                                                                                              |
| <b>sonar.basecomponent.key</b>  | ComponentID for analysis as defined "sonar.projectKey" specified in the sonar-project.properties file |                                                                                                                                                                                                         |
| <b>sonar.componentroots.key</b> | ComponentID for analysis                                                                              |                                                                                                                                                                                                         |
| <b>sonar.measure.topic</b>      | Kafka topic name for measures. Also, name for Elasticsearch index.                                    | sonarqube.measure                                                                                                                                                                                       |
| <b>sonar.issue.topic</b>        | Kafka topic name for issues. Also, name for Elasticsearch index.                                      | sonarqube.issue                                                                                                                                                                                         |
| <b>sonar.metric.keys</b>        | Metric keys to be collected                                                                           | ncloc,lines,comment_lines,complexity,<br>...<br>See <a href="https://docs.sonarqube.org/display/SONAR/Metric+Definitions">https://docs.sonarqube.org/display/SONAR/Metric+Definitions</a> for full list |
| <b>sonar.interval.seconds</b>   | Polling interval. At every poll, a new set of records is collected, regardless if data has changed or | 86400 (one day)                                                                                                                                                                                         |



|  |                                                                                  |  |
|--|----------------------------------------------------------------------------------|--|
|  | not. The interval has to be in sync with the actual sonarqube metric collection. |  |
|--|----------------------------------------------------------------------------------|--|

An additional configuration file is used to define the Elasticsearch server as a data sink (connect-sonarqube-elasticsearch.properties). Every Sonarqube item stored in the Kafka server is also passed to Elasticsearch.

| Attribute              | Description                                                   | Example                                                       |
|------------------------|---------------------------------------------------------------|---------------------------------------------------------------|
| <b>name</b>            | Name of the connector                                         | kafka-sonarqube-elasticsearch                                 |
| <b>connector.class</b> | Classname of the class implementing the connector             | io.confluent.connect.elasticsearch.ElasticsearchSinkConnector |
| <b>tasks.max</b>       | Max. number of concurrent tasks                               | 1                                                             |
| <b>topics</b>          | The Kafka Topic to read, Elasticsearch index name to store to | sonarqube.measure,sonarqube.issue                             |
| <b>connection.url</b>  | URL of the Elasticsearch Server                               | http://<elasticsearch-server-ip>:9200                         |
| <b>type.name</b>       | Index type name for Elasticsearch                             | sonarqube                                                     |

One more configuration file defines the basic connector configuration (connect-sonarqube.properties):

| Attribute                           | Description                                                                                                  | Example                          |
|-------------------------------------|--------------------------------------------------------------------------------------------------------------|----------------------------------|
| <b>bootstrap.servers</b>            | Kafka Server URL                                                                                             | <kafka-bootstrap-server-ip>:9092 |
| <b>offset.storage.file.filename</b> | Kafka offset storage, resume reading source data at the correct position                                     | /tmp/connect-sonarqube.offsets   |
| <b>rest.port</b>                    | Rest port of the connector. If multiple connectors are run, this port has to be different for every instance | 8088                             |
| ...                                 | Standard attributes not shown                                                                                | ...                              |

Before running the connector, at least adapt the following configuration values:

- connect-sonarqube.properties:
  - bootstrap.servers (set to your kafka server ip)
- connect-sonarqube-source.properties:
  - sonarqube.url (set to your sonarqube server url)
  - sonarqube.user (valid jenkins username)
  - sonarqube.pass (valid password)
  - jira.project (projects to collect issues from)

To run the connector for sonarqube, use the following command within the qr-connect directory (put into one line):

```
CLASSPATH=qr-connect-0.0.2-jar-with-dependencies.jar
<confluent-kafka-install-dir>/bin/connect-standalone
<qr-connect-install-dir>/connect-sonarqube.properties
<qr-connect-install-dir>/connect-sonarqube-source.properties
<qr-connect-install-dir>/connect-sonarqube-elasticsearch.properties
```



## Redmine Source Connector

The Redmine connector periodically polls a Redmine API for new and updated issues:

### Measures:

Table 17. Data gathered about issues with the Redmine connector: attributes, descriptions, and examples.

| Attribute               | Description                                                          | Example                    |
|-------------------------|----------------------------------------------------------------------|----------------------------|
| <b>redmineURL</b>       | URL of the Redmine server                                            | http://<ip-address>:<port> |
| <b>project_id</b>       | Project id (from database)                                           | 10                         |
| <b>project</b>          | Redmine project name                                                 | MyProject                  |
| <b>issue_id</b>         | Issue id                                                             | 20                         |
| <b>tracker</b>          | One of "Issue", "Evolution", "Assistance" etc.                       | Issue                      |
| <b>tracker_id</b>       | Tracker id (from database)                                           | 99                         |
| <b>status</b>           | Issue status. One of "New", "Assigned", "Resolved", "Feedback", etc. | New                        |
| <b>status_id</b>        | Status id (from database)                                            | 100                        |
| <b>priority</b>         | Issue priority (e.g. "Low", "Medium", "High")                        | Medium                     |
| <b>priority_id</b>      | Priority id (from database)                                          | 101                        |
| <b>author</b>           | Issue author                                                         | JohnDoe                    |
| <b>author_id</b>        | Author id (from database)                                            | 101                        |
| <b>assigned_to</b>      | Issue assigned to person                                             | JohnDoe                    |
| <b>assigned_to_id</b>   | Assigned to id from database                                         | 102                        |
| <b>fixed_version</b>    | Version where issue was fixed                                        | Version-1.3.5              |
| <b>fixed_version_id</b> | Fixed version id (from database)                                     | 105                        |
| <b>subject</b>          | Issue subject                                                        | Lore ipsum ...             |
| <b>start_date</b>       | Start date                                                           | 2017-10-10                 |
| <b>done_ratio</b>       | Percentage issue done                                                | 50                         |
| <b>created_on</b>       | Creation timestamp                                                   | 2014-06-02T07:09:53Z       |
| <b>updated_on</b>       | Last update timestamp                                                | 2014-09-01T07:52:54Z       |

The index mapping for the Redmine Elasticsearch index (elasticsearch.redmine.schema) has to be put into Elasticsearch before the connector is run (e.g. by using Kibana Dev Tools):

```
PUT redmine
{
  "mappings": {
    "redmine": {
      "properties": {
        "redmineURL": { "type": "keyword" },
        "project_id": { "type": "long" },
        "project": { "type": "keyword" },
        "issue_id": { "type": "long" },
        "tracker": { "type": "keyword" },
        "tracker_id": { "type": "long" },
        "status": { "type": "keyword" },
        "status_id": { "type": "long" },

```



```

    "priority": { "type": "keyword" },
    "priority_id": { "type": "long" },
    "author": { "type": "keyword" },
    "author_id": { "type": "long" },
    "assigned_to": { "type": "keyword" },
    "assigned_to_id": { "type": "long" },
    "fixed_version": { "type": "keyword" },
    "fixed_version_id": { "type": "long" },
    "subject": { "type": "text" },
    "start_date": { "type": "keyword" },
    "done_ratio": { "type": "date" },
    "created_on": { "type": "keyword" },
    "updated_on": { "type": "text" }
  }
}
}
}

```

The Redmine source connector configuration is stored in a properties file (connect-redmine-source.properties):

| Attribute                       | Description                                                                                                                                                    | Example                                                                                                                                                                                              |
|---------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>name</b>                     | Name of the connector                                                                                                                                          | kafka-sonar-source-connector                                                                                                                                                                         |
| <b>connector.class</b>          | Classname of the class implementing the connector                                                                                                              | connect.redmine.RedmineSourceConnector                                                                                                                                                               |
| <b>redmine.url</b>              | URL of the Redmine server                                                                                                                                      | http://<sonar-ip>:<port>                                                                                                                                                                             |
| <b>redmine.user</b>             | Authentication Username, leave blank for no authentication                                                                                                     | <Username>                                                                                                                                                                                           |
| <b>redmine.pass</b>             | Authentication Password                                                                                                                                        | <Password>                                                                                                                                                                                           |
| <b>redmine.created.since</b>    | Read only issues created after this date                                                                                                                       | 2000-01-01                                                                                                                                                                                           |
| <b>redmine.topic</b>            | Kafka topic name for Redmine issues. Also, name for Elasticsearch index.                                                                                       | redmine                                                                                                                                                                                              |
| <b>sonar.metric.keys</b>        | Metric keys to be collected                                                                                                                                    | ncloc,lines,comment_lines,complexity, ...<br>See <a href="https://docs.sonarqube.org/display/SONAR/Metric+Definitions">https://docs.sonarqube.org/display/SONAR/Metric+Definitions</a> for full list |
| <b>redmine.interval.seconds</b> | Polling interval. At every poll, all new and updated issues are collected. or not. The interval has to be in sync with the actual sonarqube metric collection. | 3600 (one hour)                                                                                                                                                                                      |

An additional configuration file is used to define the Elasticsearch server as a data sink (connect-sonarqube-elasticsearch.properties). Every Sonarqube item stored in the Kafka server is also passed to Elasticsearch.

| Attribute   | Description           | Example                     |
|-------------|-----------------------|-----------------------------|
| <b>name</b> | Name of the connector | kafka-redmine-elasticsearch |



|                        |                                                               |                                                               |
|------------------------|---------------------------------------------------------------|---------------------------------------------------------------|
| <b>connector.class</b> | Classname of the class implementing the connector             | io.confluent.connect.elasticsearch.ElasticsearchSinkConnector |
| <b>tasks.max</b>       | Max. number of concurrent tasks                               | 1                                                             |
| <b>topics</b>          | The Kafka Topic to read, Elasticsearch index name to store to | redmine                                                       |
| <b>connection.url</b>  | URL of the Elasticsearch Server                               | http://<elasticsearch-server-ip>:9200                         |
| <b>type.name</b>       | Index type name for Elasticsearch                             | redmine                                                       |

One more configuration file defines the basic connector configuration (connect-sonarqube.properties):

| Attribute                           | Description                                                                                                  | Example                          |
|-------------------------------------|--------------------------------------------------------------------------------------------------------------|----------------------------------|
| <b>bootstrap.servers</b>            | Kafka Server URL                                                                                             | <kafka-bootstrap-server-ip>:9092 |
| <b>offset.storage.file.filename</b> | Kafka offset storage, resume reading source data at the correct position                                     | /tmp/connect-sonarqube.offsets   |
| <b>rest.port</b>                    | Rest port of the connector. If multiple connectors are run, this port has to be different for every instance | 8088                             |
| ...                                 | Standard attributes not shown                                                                                | ...                              |

Before running the connector, at least adapt the following configuration values:

- connect-redmine.properties:
  - bootstrap.servers (set to your kafka server ip)
- connect-redmine-source.properties:
  - redmine.url (set to your Redmine server url)
  - redmine.user (valid jenkins username, or empty if authentication is not needed)
  - redmine.pass (valid password)

To run the connector for sonarqube, use the following command within the qr-connect directory (put into one line):

```
CLASSPATH=qr-connect-0.0.2-jar-with-dependencies.jar  
<confluent-kafka-install-dir>/bin/connect-standalone  
<qr-connect-install-dir>/connect-redmine.properties  
<qr-connect-install-dir>/connect-redmine-source.properties  
<qr-connect-install-dir>/connect-redmine-elasticsearch.properties
```

### [GitLab connector \(alternative in the ITTI use case\)](#)

As an alternative approach to data ingestion with Apache Kafka, for the in the ITTI use case and the GitLab data source, we have implemented the collection of information using a mixture of Node.js and MongoDB frameworks. Using JavaScript, we periodically pool the GitLab RESTful interface to retrieve the raw data, as well as to process, and store information in MongoDB. Similarly, we use JavaScript scripts to periodically push the pre-processed data the ElasticSearch platform.



### Source code structure

The code of the GitLab connector, for the ITTI use case, is arranged in a single directory, which contains following content:

| File/Directory             | Description                                                                          |
|----------------------------|--------------------------------------------------------------------------------------|
| <b>elastic</b>             | Contains implementation of Graphical User Interface for collected data visualization |
| <b>lib</b>                 | Contains scripts implementing data retrieval via GitLab RESTful API                  |
| <b>node_modules</b>        | Third-party libraries                                                                |
| <b>log</b>                 | Contains log files                                                                   |
| <b>elk-transporter.js</b>  | Transports collected data to Elasticsearch via HTTP interface                        |
| <b>daily_dbr_push.js</b>   | Calculate PoC's factors for QR-Eval tool                                             |
| <b>historian.js</b>        | Analyses collected data to calculate metrics                                         |
| <b>sync_with_gitlab.sh</b> | Helper script that synchronizes data on a daily basis                                |
| <b>config.json</b>         | Contains configuration details                                                       |

### GitLab connector

Before the data is being collected, the access details have to be provided. These are maintained in a 'config.json' file. An example configuration file has been shown below (Figure 21):

```
"esa-otx":{
  "gitlab":{
    "url":"http://localhost:8383/api/v4/projects/73",
    "secret":"SecretKey",
    "labels_mapping":{
      "bug":67,
      "backlog":71,
      "in_progress":72,
      "ready":73,
      "testing":74
    }
  }
},
```

Figure 21. GitLab connector configuration file.

The file contains information about URL address, secret key (an access key instead of a login and password), and mapping of labels. The URL indicates the hostname, version of the API (currently we only support version 4), and the id of the project to be analyzed (it can be retrieved directly from the GitLab tool). The secret key is an access token that has to be generated with the GitLab. It prevents from storing user password in a clear form in the configuration file. Finally, the labels mapping table contains the ids of relevant (for our connector) labels that are assigned to the issues notes. These labels are further explained in the next sections.

Currently, we store two types of objects retrieved from GitLab, namely: issue and notes. These are arranged in one-to-many relations, meaning that each issue can have zero or many accompanying notes. As it is shown below (in Figure 22) these two objects are related with a field named 'notable\_iid'.



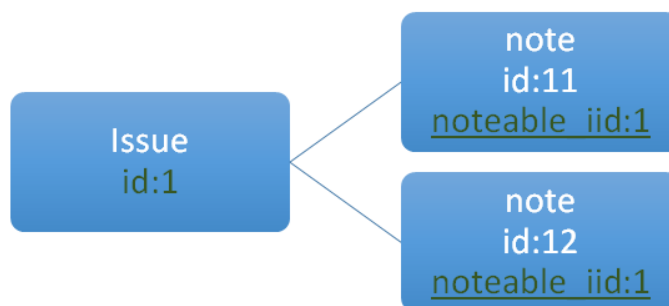


Figure 22. Relation between issue and its notes

The example of note is shown below in Figure 23. The object contains information about the author, the creation and update times, attachment (e.g. picture) and the body. The body field is the most interesting (from our perspective) since it encodes what kinds of labels have been added (or removed) and by whom.

```
1  {
2      "_id" : ObjectId("5a671561bbe0d32362d7811b"),
3      "id" : 38581,
4      "body" : "added ~216 and removed ~72 labels",
5      "attachment" : null,
6      "author" : ...,
7      "created_at" : "2018-01-22T09:47:51.675Z",
8      "updated_at" : "2018-01-22T09:47:51.675Z",
9      "system" : true,
10     "noteable_id" : 1156,
11     "noteable_type" : "Issue",
12     "noteable_iid" : 263,
13     "parent_id" : 263
14 }
```

Figure 23. Example of Issue note object

The example of issue object obtained with the connector is shown in Figure 24.



```
1  {
2      "_id" : ObjectId("5a67155fbbe0d32362d780b7"),
3      "id" : 1165,
4      "iid" : 271,
5      "project_id" : 73,
6      "title" : "Procedura",
7      "description" : "Celem zadania\n",
8      "state" : "opened",
9      "created_at" : "2018-01-23T09:54:17.286Z",
10     "updated_at" : "2018-01-23T10:06:59.952Z",
11     "closed_at" : null,
12     "labels" : [
13         "MUST",
14         "SUT",
15         "Sprint Backlog"
16     ],
17     "milestone" : {
18         "id" : 64,
19         "iid" : 14,
20         "project_id" : 73,
21         "title" : "Sprint 35",
22         "description" : "",
23         "state" : "active",
24         "created_at" : "2018-01-19T14:23:14.963Z",
25         "updated_at" : "2018-01-19T14:23:14.963Z",
26         "due_date" : "2018-01-28",
27         "start_date" : "2018-01-22"
28     },
29     "assignees" : ...,
30     "author" : ...,
31     "assignee" : ...,
32     "user_notes_count" : 1,
33     "upvotes" : 0,
34     "downvotes" : 0,
35     "due_date" : null,
36     "confidential" : false,
37     "discussion_locked" : null,
38     "web_url" : "http://gitlab.itti.com.pl/zwo/esa-otx/issues/271",
39     "time_stats" : {
40         "time_estimate" : 57600,
41         "total_time_spent" : 0,
42         "human_time_estimate" : "2d",
43         "human_total_time_spent" : null
44     }
45 }
```

Figure 24. Example of issue object

The issue allows us to retrieve such information as a unique id, creation time, last update time, assignee, the milestone (the deadline), author, time estimates (if provided), and state.

Both the objects (issue and note) allow us to record all the changes made on a specific issue. Essentially it allows us to track when specific issue/ticket has been created, when it was put into a Sprint Backlog, when it was implemented, tested and finally closed (see Figure 25).



Figure 25. Example conversation (on GitLab) composed of issue notes.

In the ITTI use case and for GitLab, for each of the issues we have created a kind of Finite State Machine (FSM) in order to track the current state of the issue. An example is shown in Figure 26.

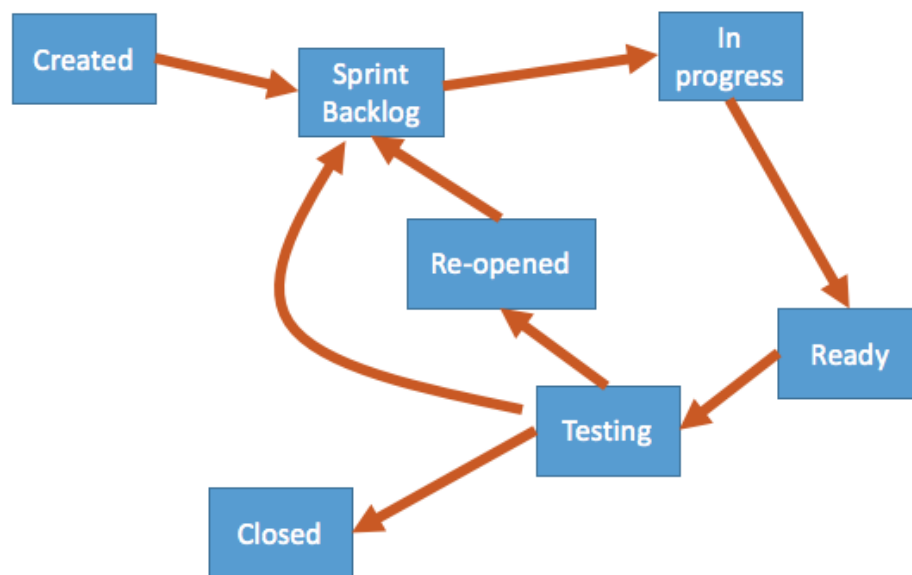


Figure 26. Example of FMS of an issue. Created is an initial state, while the Closed is a terminal state.

Each issue has its initial state, which is indicated as 'created' on the FSM diagram. Afterwards, an issue may be put into the backlog. This state is indicated as 'Sprint Backlog'. When a developer is assigned to a specific issue, the state is changed to 'In progress'. When the issue is ready, it has assigned 'Ready' state. From that moment, the result of the developer work can be tested by the Q&A engineer and hence the issue can be moved to the 'Testing' state. When all functional tests are successfully passed, the issue is moved to the terminal state indicated as 'Closed'. However, when some test fails, the issue is either moved to re-open state or (it depends on the project) it goes directly to the backlog.

Obviously, the names of the labels indicating specific states as well as the form of the FSM will depend on the company and the project. Therefore, the mapping table in the configuration file (config.json) allows the user to indicate what state correlates with a specific label.

The number of FSM we have to track is equal to the number of issues that have been defined in GitLab. It would be overwhelming from the point of view of a user to analyze all of them. Therefore, we calculate project-wise aggregated metrics for each daily snapshot. The idea is shown in the Figure 27.

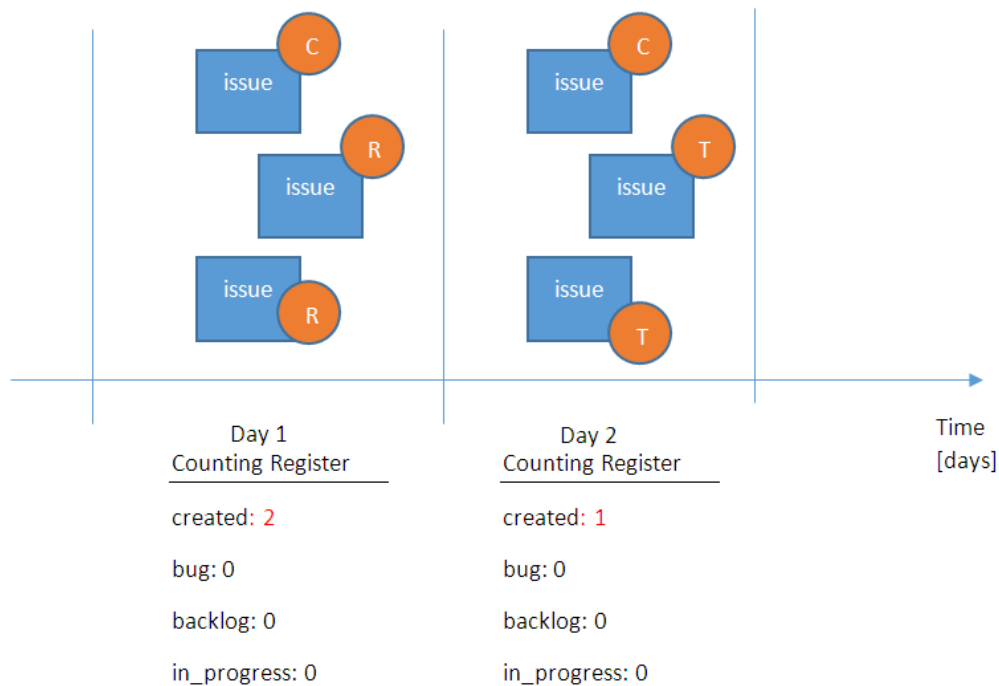


Figure 27. Example shown how we calculate daily snapshot statistics for the entire project.

Using that approach we can plot a time-series indicating how the specific metric changed over time (see Figure 28). In example, as it is shown below we can show, the per-sprint average number of tasks being in backlog, 'in progress' or finished (ready) states.

## Sprints

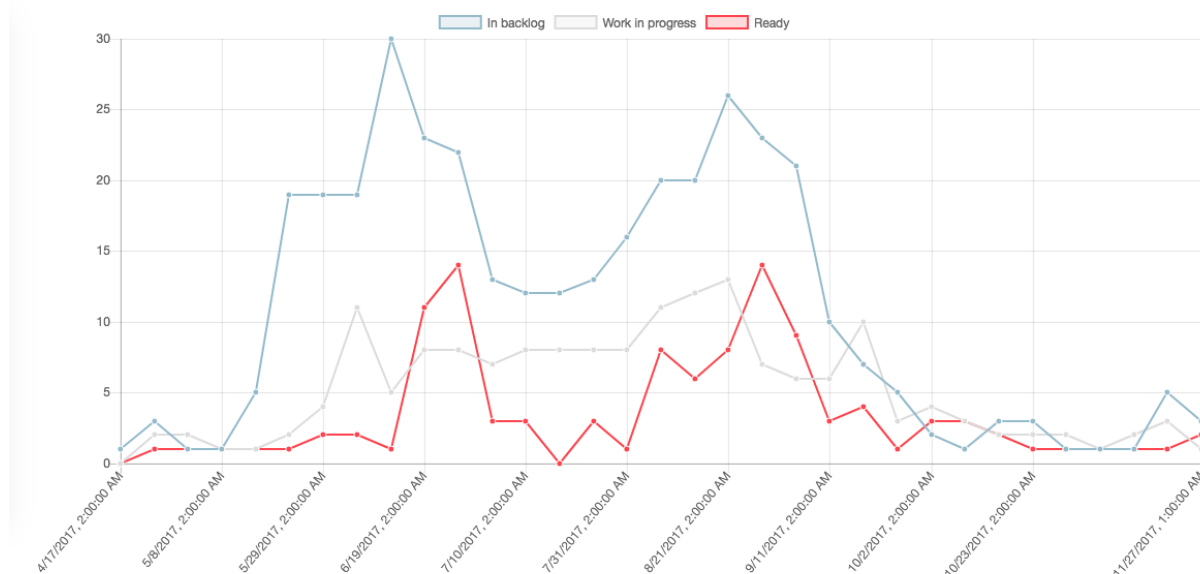


Figure 28. Metrics assessing the development part: number of tasks in backlog, number of task being under development, and number of tasks waiting for testing (the values are shown as average calculated for a period of a sprint).

Similarly, we can calculate metrics related to the performance of testing. As it is shown on the Figure 29, using the FSM we can calculate how many developed module has failed to pass the tests (Failed tests), how



many testing tasks are pending (Pending tests), and how many bug (Flagged as a bug) issues have been created.

## Testing

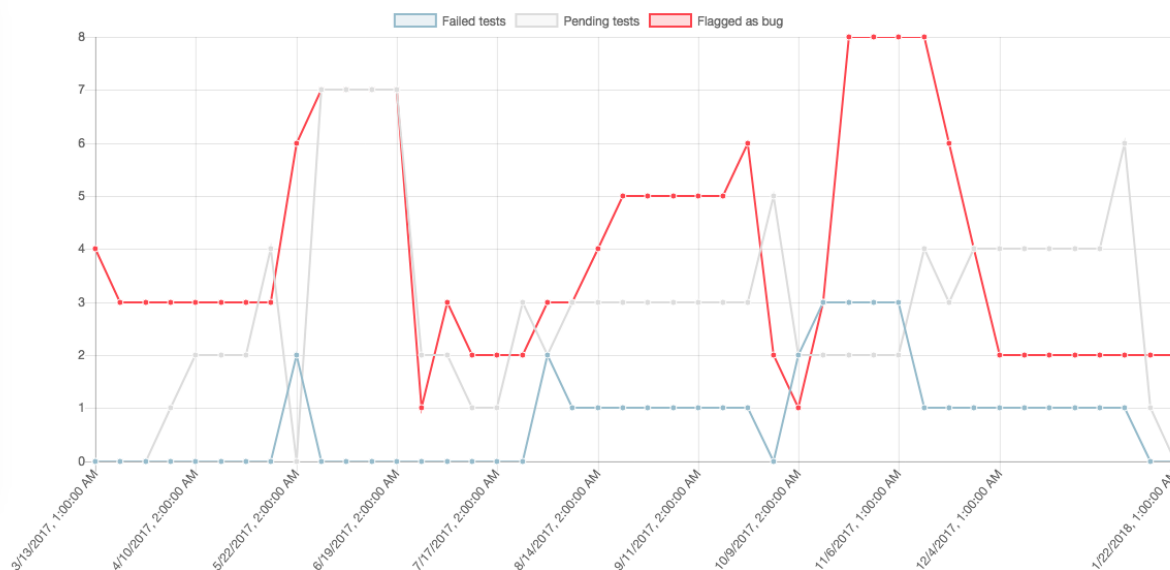


Figure 29. Metrics assessing the performance of testing.

Moreover, we also calculate additional useful from the perspective of a product owner metrics such as the total cumulative number of created, tested, and closed issues (see Figure 30). It is also possible to show the number of still opened issues and the number of delayed issues.

## Issues

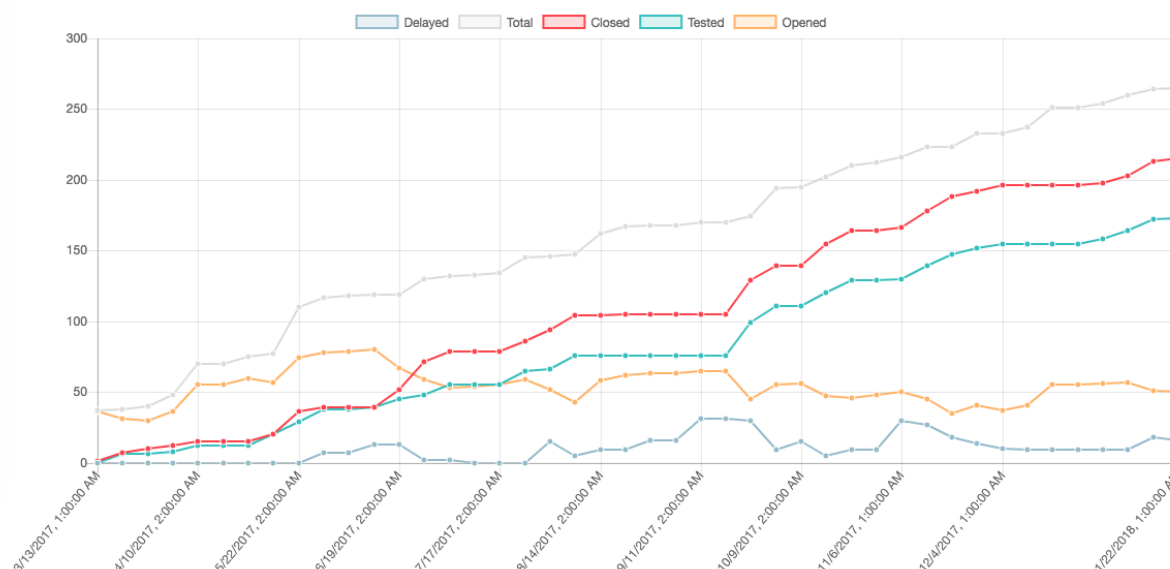


Figure 30. Example of global metrics characterizing general advancement of the project.

## Evaluation of technical aspects

In this sub-section, we describe technical aspects and architectural choices behind the implemented pipeline for GitLab data gathering and analyses.



**Scalability:** MongoDB offers scalability via cluster capabilities. Also, Node.js allows utilizing architectural design patterns to build scalable web services. Moreover, Node.js is accompanied with the wide spectrum of libraries allowing for rapid implementation of solutions interfacing with other web systems (e.g. REST APIs), fast JSON parsing (Node.js is JavaScript-based system), and asynchronous data processing.

**Flexibility:** Both MongoDB and Node.js are highly oriented on processing JSON-based document files. We decide to choose these solutions because the data and the interface provided by GitLab tool is based on JSON notation.

**Data structure complexity:** In order to measure relevant project-related metrics, we need additional processing. In fact, the data stored in the GitLab in raw form (day-snapshot) do not contain meaningful information. The data are part of a project FSM (Finite State Machine). Each daily snapshot, is an action that changes an internal state of a project model. Therefore, we have to track the current state and apply changes after each update (snapshot).

### How to define the quality model indexes in Elastic.

Execute in Kibana "Dev Tools" the index mappings defined in Table 18, Table 19, Table 20. For instance, the metric index includes the following properties:

- 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.

An example of executing the metrics index mapping is shown in Figure 31.

Table 18. Metrics index mapping.

```
-----
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"
      }
    }
  }
}
}
}

```

Table 19. Factors index mapping.

```

-----
FACTORS INDEX MAPPING
-----
PUT poc.factors
{
  "mappings": {
    "factors": {
      "properties": {
        "factor": {
          "type": "keyword"
        },
        "name": {
          "type": "keyword"
        },
        "description" : {
          "type" : "text"
        },
        "evaluationDate": {
          "type": "date"
        },
        "value": {
          "type": "float"
        },
        "strategic_indicators" : {
          "type": "keyword"
        },
        "datasource" : {
          "type" : "keyword"
        }
      }
    }
  }
}

```

Table 20. Strategic indicators index mapping.

```

-----
STRATEGIC_INDICATORS INDEX MAPPING
-----

PUT poc.strategic_indicators
{

```





```
"mappings": {
  "strategic_indicators": {
    "properties": {
      "strategic_indicator": {
        "type": "keyword"
      },
      "name": {
        "type": "keyword"
      },
      "description" : {
        "type" : "text"
      },
      "evaluationDate": {
        "type": "date"
      },
      "value": {
        "type": "float"
      },
      "datasource" : {
        "type" : "keyword"
      }
    }
  }
}
```

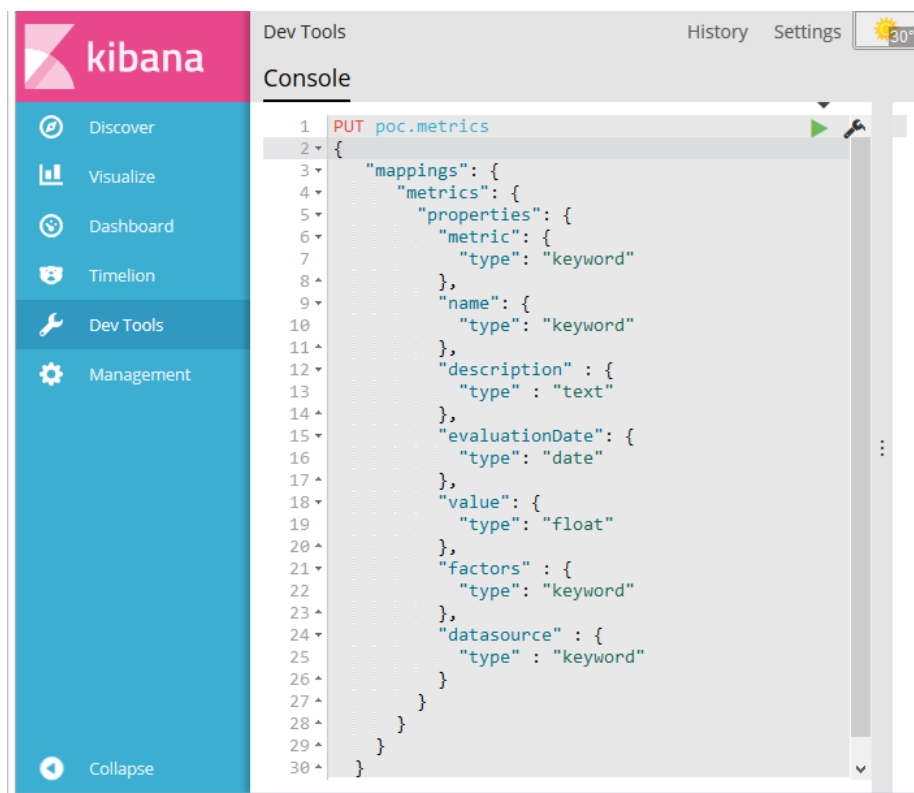


Figure 31. Defining quality model indexes in ELK.



## How to customise the quality model and perform the quality model assessment: the qr-eval-module

### Setup

1. Configure index names and project ids (e.g., jira, jenkins, sonarqube.measures) in index.properties. This is necessary to access to the data producers.
2. Enable/Disable Metrics, set thresholds, and configure target factors in metrics.properties
3. Enable/Disable Factors and configure target indicators in factors.properties
4. Enable/Disable indicators in indicators.properties

### Run

Run as executable jar file: `java -jar qr-eval-x.y.z-jar-with-dependencies.jar`