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

Q-Rapids Quality Model (April 2019)

This document contains the Q-Rapids quality model offered by default in the Q-Rapids software analytics tool. It can be customized in the different companies. The document is divided in three sections:

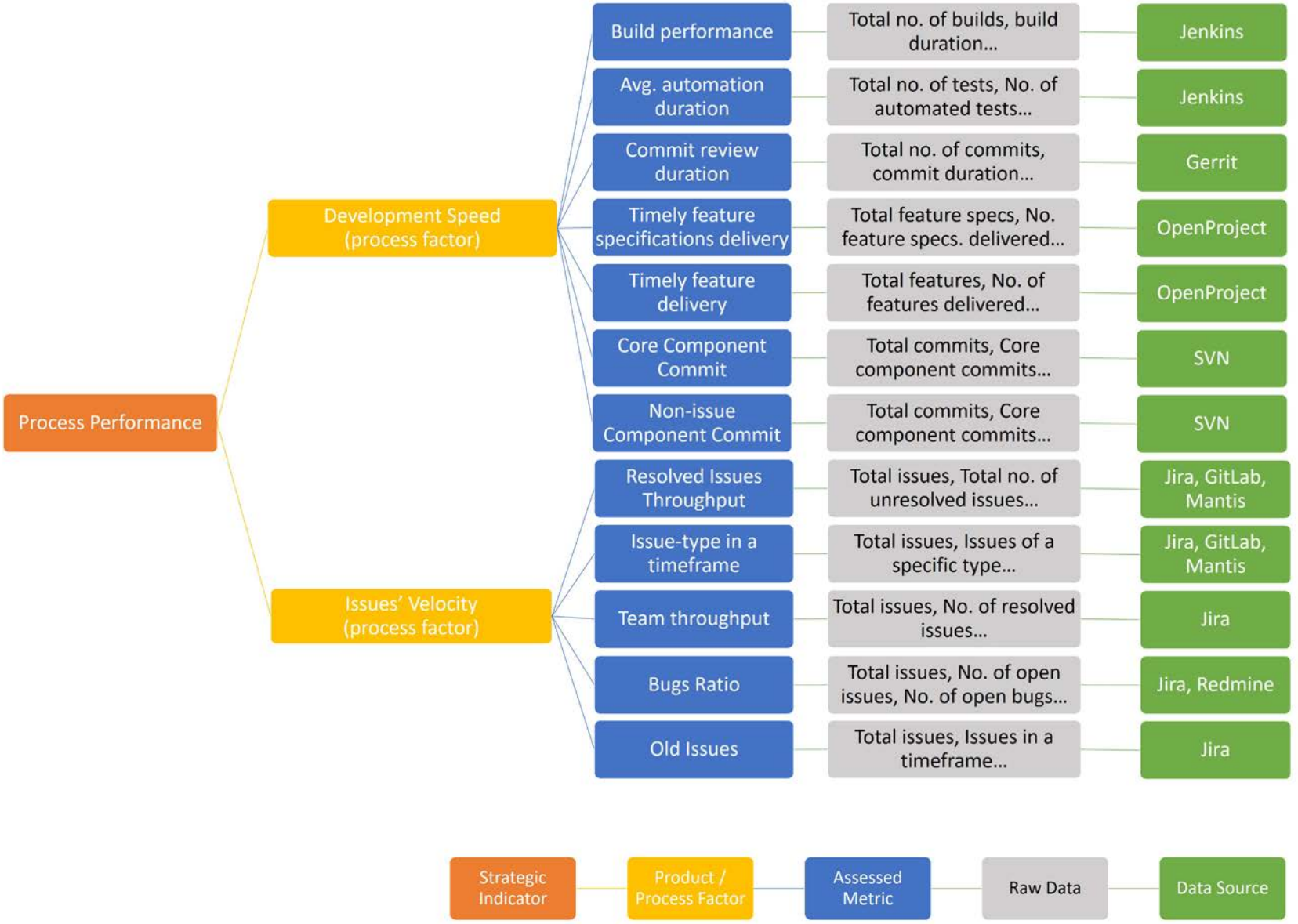
1. Graphical representation of the quality model
2. Description of factors, assessed metrics, raw data, and data sources
3. Description of strategic indicators and factors

Information about how to customize the quality model can be found on https://github.com/q-rapids/qrapids-eval
This document is available as an appendix of the publication "Silverio Martínez-Fernández, Anna Maria Vollmer, Andreas Jedlitschka, Xavier Franch, Lidia López, Prabhat Ram, Pilar Rodríguez Marín, Sanja Aaramaa, Alessandra Bagnato, Michal Choras, Jari Partanen: *Continuously Assessing and Improving Software Quality With Software Analytics Tools: A Case Study*. IEEE Access 7: 68219-68239 (2019)" on https://figshare.com/s/217851eed7ec1da7ab86

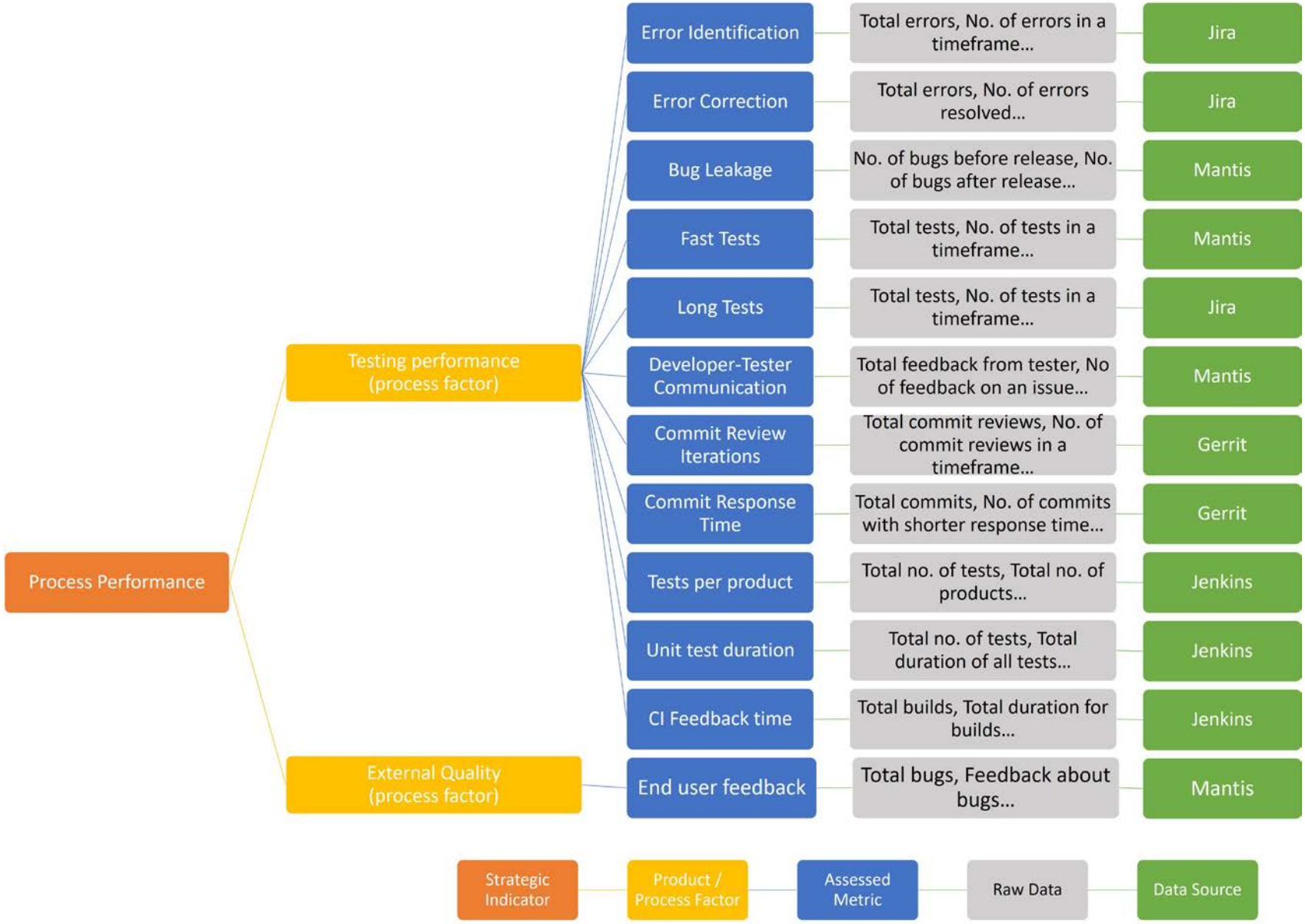
This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 732253.

Graphical representation of the quality model

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



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



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

Factors, assessed metrics, raw data, and data sources

Quality Aspect	Factor	Assessed Metric	Description	Calculation	Raw Data	Data Source
Maintainability	Code Quality	Complexity	Files below the threshold of cyclomatic complexity (10 by default)	$\text{Density of non} - \text{complex files} = \frac{\text{Non} - \text{Complex files}}{\text{Total number of files}}$ where a file is complex if the <cyclomatic complexity> per <function> is greater than <10>.	*Cyclomatic complexity* per *function* of each file, *total number of files*	SonarQube
		Comments	Files whose comment density is outside the defined thresholds (by default 10%-30%)	$\text{Density of commented files} = \frac{\text{Commented files}}{\text{Total number of files}}$ where a file is commented if the < density of comment lines> is between 10% and 30% $\text{Density of comment lines} = \frac{\text{Comment lines}}{(\text{Lines of code} + \text{Comment lines})}$	*Density of comment lines* and *lines of code* per each file	SonarQube
		Duplication	Files below the threshold of duplicated lines percentage	$\text{Absence of duplications} = \frac{\text{Files without duplications}}{\text{Total number of files}}$ where a file has no duplications if the < density of duplication> is less than 5% $\text{Density of duplication} = \frac{Duplicated\ lines}{\text{Lines}}$	*Duplicated lines* and *lines of code* per file	SonarQube
	Blocking Code	Non-blocking files	Files without critical or blocker quality rule violations	$\text{Fulfillment of critical/blocker quality rules} = \frac{Files\ with\ critical\ or\ blocker\ issues}{\text{Total number of files}}$	Number of *quality rule violations* per file and their *severity* (blocker, critical, major, minor, info) and *type* (code smell, bug, vulnerability)	SonarQube, Coverity, CodeSonar
	Quality Issues' Specification	Well defined issues	Density of issues that have good definition	$\text{Well defined issues} = \frac{\text{Well defined issues}}{\text{Issues}}$ where an issue is well defined if the fields {<description>, <due date>} are not empty	*Fields of each issue* (e.g., description, due date, assignee, estimated time)	Jira, GitLab, Redmine, Mantis
Reliability	Testing Status	Test success	Unit test success density	$\text{Passed tests} = \frac{(Unit\ tests - (Unit\ test\ errors + Unit\ test\ failures))}{\text{Unit tests}}$	Number of *unit test errors*, *failures*, *skipped*, and *total*	Jenkins, GitLab
	Testing Performance	Fast tests	Test builds below the duration threshold	$\text{Density of fast tests' builds} = \frac{\text{Fast unit test}}{\text{Unit tests}}$	*Duration of unit test execution*, *tests* conforming to a *pipeline*	Jenkins, GitLab

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

Quality Aspect	Factor	Assessed Metric	Description	Calculation	Raw Data	Data Source
				where an unit test is fast if its duration is lower than <5 minutes>		
	Software Stability	Bug density	Ratio of open issues of the type bug with respect to the total number of issues within a customized timeframe	Ratio of bugs $= \frac{Number\ of\ open\ issues/in\ progress/re-opened\ of\ type\ "bug"}{Number\ of\ open\ issues/in\ progress/re-opened} * 100$	Total number of *issues* (a.k.a. tasks) per *status* (e.g., open, done), *type* (e.g., bug, maintenance, feature), and *timeframe* (e.g., current/last month or current/last sprint)	Jira, GitLab, Redmine, Mantis
	Known Remaining Defects	Postponed Issues Ratio	Ratio of non-critical issues that are not closed	$\frac{Number\ of\ low-severity\ closed\ issues}{Total\ number\ of\ low-severity\ issues}$	*Total no. of open and closed issues* with *low severity*	Mantis
	Product Stability	Critical Issues Ratio	Ratio of critical issues that have been closed	$= \frac{Number\ of\ high-severity\ closed\ issues}{Total\ number\ of\ high-severity\ issues}$	*Total no. of open and closed issues* with *high severity*	Mantis
		Build Stability	Density of successful builds	$= \frac{Number\ of\ successful\ builds}{Total\ number\ of\ builds}$	*Number of successful builds* and *total number of builds*	Jenkins
Functional Suitability	External Quality	End user feedback	Density of end user feedback related to an issue in a given timeframe	$= \frac{User\ feedback\ in\ a\ given\ timeframe}{Total\ user\ feedback}$ where the "*customer*" field is not empty	All the *feedback* received in a timeframe (week, month, etc.) and the *total number of feedback*	Mantis
Productivity	Issues' Velocity	Resolved Issues' Throughput	Density of issues whose resolution didn't take longer than the defined duration threshold	$= \frac{Number\ of\ issues\ resolved\ under\ the\ duration\ threshold}{Total\ number\ of\ issues\ in\ a\ given\ timeframe}$ where the duration threshold is user defined	*Total number of issues* with *resolved status* (e.g. sprint, week, month) and *total issues* in *timeframe*	Jira, Mantis, GitLab
		Issues-type in a timeframe	Density of issues of a specific type within a defined timeframe	$= \frac{Number\ of\ issues\ of\ a\ specific\ type}{Total\ number\ of\ issues\ in\ a\ given\ timeframe}$ where *type = resolved, open, task, bugs*, etc.	*Total number of issues* with *type* (e.g. open, closed, resolved, task, bugs) and *total issues* in *timeframe* (sprint, week, month)	Jira, Mantis, GitLab
		Team Throughput	Density of issues resolved by a team in a given timeframe	$= \frac{Number\ of\ resolved\ issues}{Total\ number\ of\ issues\ in\ a\ given\ timeframe}$	*Total number of issues* (a.k.a. story points) with *status resolved/completed* and *total issues* in *timeframe* (sprint, week, month)	Jira

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

Quality Aspect	Factor	Assessed Metric	Description	Calculation	Raw Data	Data Source
		Bugs Ratio	Density of open issues not being bugs	$= \frac{Number\ of\ non-bug\ open\ issues}{Total\ number\ of\ issues\ in\ a\ given\ timeframe}$	*Total number of issues* with *status open, total no. of issues* of *type bug, and total no. of issues in timeframe* (sprint, week, month)	Jira, Redmine
		Old Issues	Density of issues of a given priority older than the defined creation threshold	$= \frac{Number\ of\ issues\ \ older\ than\ a\ creation\ threshold}{Total\ number\ of\ issues\ in\ a\ given\ timeframe}$ where the duration threshold is user defined	*Total number of issues* (a.k.a. story points) with any *status* older than a threshold and *total issues total issues* in *timeframe* (sprint, week, month)	Jira
	Activities Completion	Development Task Completion	Density of project tasks that have been completed	$= \frac{Number\ of\ completed\ project\ tasks}{Total\ number\ of\ project\ tasks}$	*All project tasks* that are *completed*, and *total number of project tasks*.	OpenProject
		Specification Task Completion	Density of specification tasks that have been completed	$= \frac{Number\ of\ completed\ specification\ tasks}{Total\ number\ of\ specification\ tasks}$	*All specification tasks* that are *completed*, and *total number of specification tasks*.	OpenProject
	Development Speed	Build performance	Density of daily builds that didn't take longer than the defined duration threshold	$= \frac{Number\ of\ builds\ not\ taking\ more\ than\ the\ duration\ threshold}{Total\ number\ of\ builds\ in\ a\ given\ timeframe}$ where the duration threshold is user defined	Every *build* with *build duration* under a threshold, and *total number of builds* in *timeframe* (week, month, sprint, etc.)	Jenkins
		Avg. Automation Duration	Density of automated tests that didn't take longer than the defined duration threshold	$= \frac{Number\ of\ automated\ tests\ under\ the\ duration\ threshold}{Total\ number\ of\ automated\ tests\ in\ a\ given\ timeframe}$ where the duration threshold is user defined	Every *automated tests* with *test duration* under a threshold, and *total number of automated tests* in *timeframe* (week, month, sprint, etc.)	Jenkins
		Commit review duration	Average commit review speed for a period	$= \frac{Number\ of\ commit\ reviews\ done\ within\ the\ duration\ threshold}{Total\ number\ of commit\ reviews\ in\ a\ given\ timeframe}$ where the duration threshold is user defined	Every *commit review* with *duration* under a threshold, and *total number of commit reviews* in a *timeframe* (week, month, sprint, etc.)	Gerrit

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

Quality Aspect	Factor	Assessed Metric	Description	Calculation	Raw Data	Data Source
		Timely feature specifications delivery	Density of feature specifications delivered on time during development cycle	$= \frac{Feature\ specifications\ delivered\ ontime}{Total\ feature\ specifications\ delivered}$ where an issue is a feature specifications if the "*description*" field has "*specification"* value, and is delivered on time if the "*ontime*" field doesn't have "*F*"	All the *feature specifications* delivered on time, and *total feature specifications* delivered	OpenProject
		Timely feature delivery	Density of feature delivered on time during development cycle	$= \frac{Feature\ delivered\ ontime}{Total\ features\ delivered}$ where an issue is a feature if the "*description*" field has "*specification"* value, and is delivered on time if the "*ontime*" field doesn't have "*F*"	All the *feature* delivered on time, and *total features* delivered	OpenProject
		Core component commits	Density of core component commits in a given timeframe before the planned release	$= \frac{Core\ component\ commits\ in\ a\ given\ timeframe}{Total\ core\ component\ commits}$ where a commit is on a core component if the "*filename*" field points to "*.bpmn" files*	All the *commits* on core component in a timeframe (week, month, etc.) and the *total commits* made on the core components	SVN
		Non-issue component commit	Density of commits that are not related to an issue	$= \frac{Number\ of\ commits\ not\ related\ to\ any\ issue}{Total\ number\ of\ commits}$ here, "issue" refers to actual issue in the code, and not the Jira terminology for a tickt	All the *commits* not related to any *issue*, and *total no. of commits* in a timeframe (week, sprint, month, etc.)	SVN
	Testing Performance	Error identification	Density of errors identified in a given timeframe	$= \frac{Number\ of\ issue\ of\ type\ error/bug\ identified\ in\ a\ timeframe}{Total\ number\ of errors\ identified\ since\ start}$	All the *issue* of type *error/bug* in a *timeframe* (week, month, sprint, etc.), and *total issues* of all type since start of the project	Jira
		Error correction	Density of errors that were corrected within the defined duration threshold	$= \frac{Number\ of\ issues\ of\ type\ error/bug\ resolved\ within\ duration\ threshold}{Total\ number\ of errors\ resolved\ in\ a\ timeframe}$ where the duration threshold is user defined	All the *issue* of type *error/bug* with status *resolved* within a duration threshold, and *total errors/bugs resolved* in a timeframe (week, month, sprint, etc.)	Jira
		Bug leakage	Density of bugs identified after the release	$= \frac{Number\ of\ bugs\ identified\ after\ the\ release}{Total\ number\ of bugs\ identified\ before\ the\ release}$ where bugs after the release are identified as customer-reported issues that does not have the keyword "feature"	All the *bugs* reported by the customer after the release, and total number of *bugs* identified before the release	Mantis

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

Quality Aspect	Factor	Assessed Metric	Description	Calculation	Raw Data	Data Source
		Fast Tests	Density of tests that are under the testing duration threshold	$= \frac{Number\ of\ tests\ with\ total\ duration\ under\ the\ duration\ threshold}{Total\ number\ of tests\ executed\ in\ a\ timeframe}$ where the duration threshold is user defined	All the *tests* within execution duration within the threshold, and *total tests* executed in a timeframe (week, month, sprint, etc.)	Mantis
		Long Tests	Density of tests over the testing duration threshold	$= \frac{Number\ of\ tests\ with\ total\ duration\ over\ the\ duration\ threshold}{Total\ number\ of tests\ executed\ in\ a\ timeframe}$ where the duration threshold is user defined	All the *tests* within execution duration over the threshold, and *total tests* executed in a timeframe (week, month, sprint, etc.)	Jira
		Developer-Tester Communication	Density of developer-tester communication on a given issue	$= \frac{Number\ of\ feedback\ from\ tester\ to\ developer\ on\ an\ issue}{Total\ number\ of\ feedbacks\ for\ all\ the\ issues}$	All the *feedbacks* on an issue, and *total feedbacks* for all the issues	Mantis
		Commit Review Iterations	Density of commits whose reviews didn't exceed the defined iteration threshold	$= \frac{Number\ of\ commit\ reviews\ with\ iterations\ within\ the\ iteration\ threshold}{Total\ number\ of commit\ reviews\ in\ a\ timeframe}$ where the iteration threshold is user defined	Every *commit review* with *iterations* under the threshold, and *total number of commit reviews* in a *timeframe* (week, month, sprint, etc.)	Gerrit
		Commit Response Time	Density of commits whose response time wasn't longer than the defined duration threshold	$= \frac{Number\ of\ commit\ with\ response\ time\ under\ the\ duration\ threshold}{Total\ number\ of\ commit\ reviews\ in\ a\ timeframe}$ where the duration threshold is user defined	Every *commit* with *response time* within the threshold, and *total number of commits* in a *timeframe* (week, month, sprint, etc.)	Gerrit
		Test Per Product	Average number of test executed for a product in a timeframe	$= \frac{Number\ of\ tests\ executed\ per\ product}{Total\ number\ of\ issues\ in\ a\ timeframe}$	All the *tests* executed for every issue, and *total issues* tested in a timeframe (week, month, sprint, etc.)	Jenkins
		CI Feedback Time	Average feedback time from CI system	$= \frac{Feedback\ time\ for\ every\ build}{Total\ builds\ in\ a\ timeframe}$	Every *build* with *feedback duration*, and total *builds* in a timeframe	Jenkins
		Unit Test Duration	Density of unit tests that do not take longer than the defined threshold	$= \frac{Number\ of\ unit\ tests\ executed\ within\ the\ duration\ threshold}{Total\ number\ of\ unit\ theses\ in\ a\ timeframe}$ where the duration threshold is user defined	All the *unit tests* executed in under *defined threshold*, and *total unit tests* executed in a timeframe (week, sprint, month, etc.)	Jenkins

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

Strategic Indicators and Factors

Strategic Indicator	Definition	Factor	Description	Type	User story	Actionable analytics
Product Quality	It refers here to the maintainability, reliability, and functional suitability of your software product.	Code Quality	It measures the impact of code changes in source code quality	Product Factor	Developers, code guardians, and integrators want to gather data about the impact of code changes on code quality, so that they can manage maintainability resources.	Code quality is actionable when the product owner decides to invest a cycle into maintainability and understandability.
		Software Stability	It measures the most critical issues	Product Factor	Product directors and quality managers want to gather data about the most critical issues at runtime, so that they can maintain efficient service capability/quality/prioritization.	Actionable analytics for software stability include the urgent generation of alerts when a fault has occurred.
		Software Usage	It measures the usage pattern of a feature/software	Product Factor	Product directors and product owners want to gather data about the product usage (e.g., total time spent on functionalities, and functionalities used most/least), so that it makes completely clear how heavily each feature is used by customers and in which order the features should be prioritized for inclusion.	An example of an action point for software usage is the removal of features that are not used in the software product.
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, … .	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.	Product Factor	Developers and testers want to gather data about the state of the issues description that enter the backlog, in order to enable tracing of these tickets to story points, epics, and/or features	An action point here could be improving the practice of describing the issues well before they are taken up in a sprint
		Testing Status	It measures the quality and stability level of executed tests during development.	Product Factor	Test managers, quality assessors, and integrators want to gather data about the quality and stability level of testing, so that tests meaningful on the one hand and not skipped on the other hand.	Action points for testing status include improving tests that do not detect critical bugs during development, or improving the performance of the test pipeline.
		Blocking Code	It measures the technical debt of software code, in terms of quality rule violations	Product Factor	Developers, code guardians, and integrators want to gather data about code changes, so that they can identify new quality issues and blocking code.	Action points for blocking code include resolving blocker quality rule violations or refactoring highly changed files (e.g., God objects or configuration files).
Process Performance	It refers to your sotware devlpment lifecycle processes' efficiency and quality	Issues' Velocity	The capability of fulfilling the issues planned for a sprint/iteration	Process Factor	Product owners and project managers want to gather data about content delivered at feature build compared to planned content on exit, so that they can see the planning capability and accuracy of the team.	Actionable analytics for issues' velocity require updates for the process, such as learning from inaccurate planning or estimation, in order to better plan the next

Strategic Indicator	Definition	Factor	Description	Type	User story	Actionable analytics
						cycles, or specifying how to use the issue tracking system.
		Testing Performance	The capability of effectively and efficiently carrying out testing activities	Process Factor	Test managers, quality assessors, and integrators want to gather data about the cost-benefit ratio of testing against the resources used, so that they can optimize the use of resources and target efficiency in testing	Action pointss could include more exhaustive testing to detect and fix as many defects as possible before release, or shortening commit review duration
		Development Speed	It measures the CI related activities	Process Factor	Developers, Integrators, Product owners, and project managers want to gather data about resources consumed as part of CI activities and automated testing performance, so that they can tweak and optimize their daily build process and improve its efficiency	Action points could include increasing build frequency, improving commit review throughput
		External Quality	It measures the software quality from customer's viewpoint	Process Factor	Product owners and product directors want to gather data about their software quality as reported by the end users, so that they can act on the information and incorporate them in the next software iteration	Action points here could include enhancing and adopting a more exhaustive testing strategy to capture all possible defects before product release
Product Readiness	Product Readiness provides high level information on product readiness for the next release. A product "ready to be released" is a product which implements the features planed in the release and without blocking issues.	Activities Completion	This factor represents the status of the completion of activities plan for this release, including development and specification tasks	Process Factor	Product owners and product directors want to gather data about the status of the development in terms of tasks' completion so that they can act on the process if needed to facilitate a more timely delivery schedule	Action points here could be prioritizing pending tasks to meet delivery schedules, and allotting more resources to resolve pending tasks to meet delivery schedules
		Code Quality	It measures the impact of code changes in source code quality	Product Factor	Developers, code guardians, and integrators want to gather data about the impact of code changes on code quality, so that they can manage maintainability resources.	Code quality is actionable when the product owner decides to invest a cycle into maintainability and understandability.
		Product Stability	It measures the most critical issues	Product Factor	Product directors and quality managers want to gather data about the most critical issues at runtime, so that they can maintain efficient service capability/quality/prioritization.	Actionable analytics for software stability include the urgent generation of alerts when a fault has occurred.
		Known Remaining Defects	Bugs or defects that are outside of the major bug category that can be deferred	Process Factor	Developers, code guardians, and integrators want to gather data about the density of minor bugs or defects that can be deferred for fixing in later releases.	Action points here would be deferment of minor bugs resolution until the next release