Quality-aware Rapid Software Development Project: The Q-Rapids Project

Xavier Franch[1], Lidia Lopez[1], Silverio Martínez-Fernández[2], Marc Oriol[1], Pilar Rodríguez[3] and Adam Trendowicz[2]

[1] Universitat Politècnica de Catalunya, Barcelona, Spain
`{franch, llopez, moriol}@essi.upc.edu`
[2] Fraunhofer IESE, Kaiserslautern, Germany
`{silverio.martinez, adam.trendowicz}@iese.fraunhofer.de`
[3] University of Oulu, Oulu, Finland
`pilar.rodriguez@oulu.fi`

Abstract. Software quality poses continuously new challenges in software development, including aspects related to both software development and system usage, which significantly impact the success of software systems. The Q-Rapids H2020 project defines an evidence-based, data-driven quality-aware rapid software development methodology. Quality requirements (QRs) are incrementally elicited, refined and improved based on data gathered from software repositories, project management tools, system usage and quality of service. This data is analysed and aggregated into quality-related key strategic indicators (e.g., development effort required to include a given QR in the next development cycle) which are presented to decision makers using a highly informative dashboard. The Q-Rapids platform is being evaluated in-premises by the four companies participating in the consortium, reporting useful lessons learned and directions for new development.

Keywords: Software quality, Data-driven requirements engineering, Software analytic tools, Software repositories, Quality models, Agile software development, Rapid software development, Quality requirements, Non-functional requirements

1 Introduction

The Q-Rapids project proposes a data-driven approach to the elicitation, prioritization and management of quality requirements (QRs), see Fig. 1. Data comes from: the organization, through software development repositories and project management tools; the users, through explicit feedback and usage logs. This basic data is elaborated into strategic indicators (e.g., team productivity, product quality) and presented to decision-makers through a dashboard that also offers techniques as what-if analysis and prediction. Expert-defined alerts inform about violations on quality thresholds, and QR patterns are suggested to remedy them. The decision-maker can explore the effects of applying them and eventually decide to include a QR in the backlog, closing the cycle.

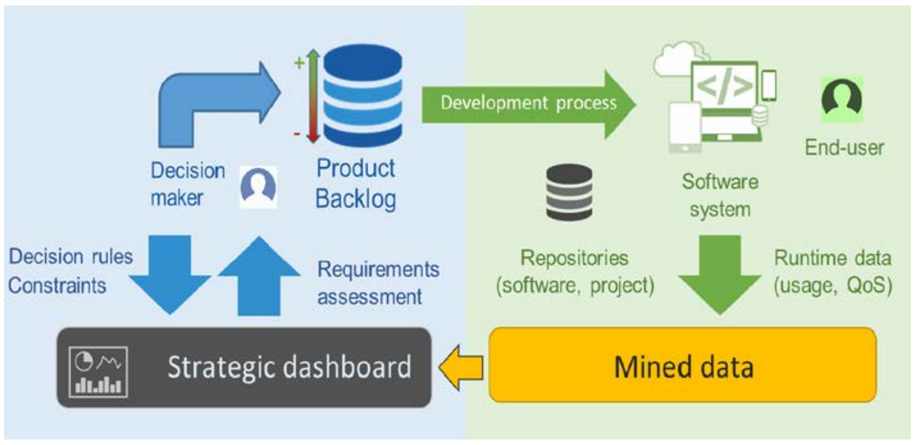

Fig. 1. The Q-Rapids Framework

The project started in November 2017 and finishes in October 2019. The consortium is composed of 3 research partners (UPC, U. Oulu and Fraunhofer-IESE) and 4 companies (Bittium, ITTI, Softeam and Nokia). The project URL is https://www.q-rapids.eu/ and the software may be found at https://github.com/q-rapids.

The rest of the paper is organized as follows. Section 2 explains the first part of the Q-Rapids cycle, namely data gathering and analysis. Section 3 introduces Q-Rapids process related aspects, remarkably process metrics. Section 4 provides details on the strategic dashboard fed by the result of the analysis. Section 5 presents the results of the evaluation conducted so far. Then, Section 6 summarizes some lessons learned and Section 7 finalizes the paper with the conclusions and related work

2 Data Gathering and Analysis

The ultimate goal of data gathering and analysis is to gain relevant knowledge about software quality (in particular at runtime) from the available software data, including development and runtime data. To achieve this goal, several tasks must be accomplished. Figure 2 presents the Cross Industry Standard Process for Data Mining, CRISP-DM [1], which Q-Rapids applied to guide analysis of software quality.

In the *business understanding* phase, the research goals of the project and business expectations of the project partners were translated into the specific objectives of data analysis. One of the analysis goals was to explore dependency between quality of software during development and its runtime quality. The development quality was represented by properties of software artifacts and development environment, in particular code, whereas runtime quality was represented by software misbehavior during testing and operation (in particular user crash reports).

The *data understanding* phase aimed at identifying sources of relevant software quality data available at application project partners and gaining first insights into the data to better understand its meaning and potential usefulness for achieving project objectives. In addition, this phase included an initial analysis of data quality as a critical success factor for the analysis. To cope with various structures of the data at involved project partners, that data from the source systems was imported as documents into a distributed storage system supported by ElasticSearch (ES) and Kibana. The major advantages of this solution include a powerful search functionality of ES and interactive visualizations offered by Kibana. Thanks to these features, initial insights into data and its quality can be made without much effort. The main sources of data included the software code repositories (e.g., Git, SVN), issue tracking systems (e.g., Jira, Mantis, Redmine), structural properties of software code (e.g., SonarQube) and runtime issue reports (e.g., Hockeyapp).

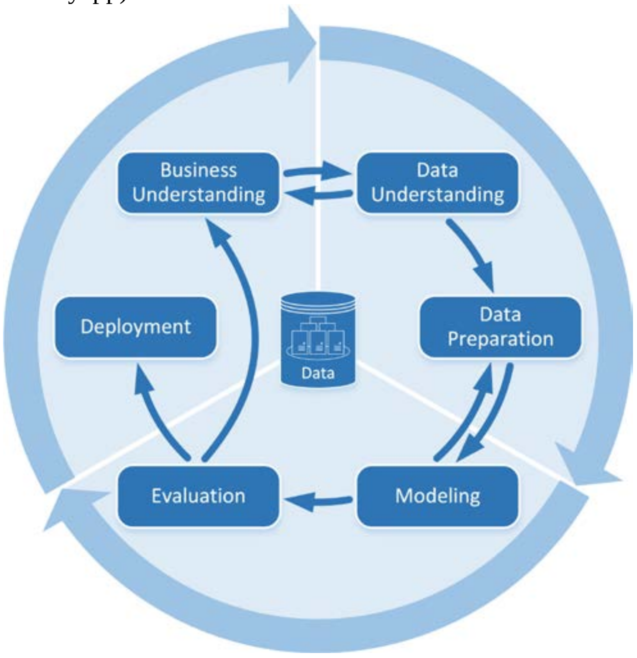

Fig. 2. CRISP-DM model used in Q-Rapids to guide the analysis of software quality

The *data preparation* phase focused on handling data quality deficits and preparing the data for specific data analyses. Data preparation tasks included: (1) integrating data stored in different source systems and structures, (2) handling data quality deficits, such as incomplete, inconsistent, and incorrect data, and (3) transforming data into the format acceptable for specific methods and tools used during the analysis. The data preparation phase is in practice the most challenging and the most expensive phase of the entire data analysis cycle; Q-Rapids was not different in this matter. Table 1 summarizes the most relevant challenges and the associated lessons we learned in the context of Q-Rapids. Summarizing, Q-Rapids replicates the experiences gained in several other projects regarding relatively low usefulness of software engineering data for quantitative analysis of software quality. Analysis of dependencies between development- and runtime-quality of software requires, on the one hand, complete and consistent data on potentially relevant quality factors (product-, process- and context characteristics); on the other hand, analysis of quality dependencies requires the data from different sources can be connected to each other, i.e., be integrated. Unfortunately, primary objectives of data collection do typically not include quality analysis and modeling. For example, the primary objective of issue tracking is to record and monitor progress of software issues, not to learn from issues, especially in connection with other software development aspects such as properties of software artifacts issues refer to or properties of the environment in which an issue occurred. In other words, quantitative cause-effect analysis does typically not belong to primary objectives of data collection.

Table 1. Example challenges of data preparation and analysis in Q-Rapids

Challenge	Recommendation
Availability/Accessibility: The required quality data is not available because appropriate data collection tools or/and processes are missing or systems were data is stored cannot be accessed. E.g., available code measurement tool does not cover all important base code metrics. For instance, although SonarQube provides data on violations of rules that are based upon specific base metrics it does not provide raw data for these metrics.	Select and set up data collection tool based on the explicitly defined business and analysis objectives, and potentially relevant data required for achieving these objectives. Data analysis should always start with business and data understanding (first two phases of the CRISP-DM model).
Completeness: Issue and change tracking data are incomplete, e.g., documented code changes are not associated to any issue.	Predefine orthogonal issue categories; use them consistently to classify issues; ensure every change can be associated to an issue.
Correctness: Actual type (nature) of change is not documented and cannot be recognized automatically. In particular, file rename or movement is recognized as deletion and addition of an entire file.	Label actual amount of change (e.g., in terms of its labor cost) to distinguish between changes that are or are not significant from the perspective of their potential impact on software quality.
Consistency: Summary changes for multiple issues of different type are documented in single change tracking entry. So, the exact amount of change per issue type is unknown.	Collection of already aggregated data should be avoided. Raw data on possibly atomic level should be supported by data collection tools (e.g., issue and change records).
Consistency: Inconsistent temporal granularity of different data sets, e.g., code changes and measurement data are recorded per commit (e.g., several times per day) but test and usage issues are recorded once a week or on an irregularly basis.	Associated data should be collected on a consistent granularity level, i.e., one entry per issue or issue type (e.g., new feature development, bug fix) to support data integration and cause-effect quality analyses.
Precision: Data is collected on a granularity level inappropriate for accomplishing analysis objectives. An example are source code measurements, collected on a file level. In such case, fine-granular changes on class- and function-level can compensate each other within a file and be thus not visible in the corresponding measurement data.	Based on explicitly defined business and analysis goals, specify precision (granularity) requirements for the necessary data. Set up tools that support collecting data with the required precision.
Redundancy: Issue tracking data contain duplicated entries.	Data collection and reporting tool should support real-time checks for duplicated, incorrect and inconsistent data entries.

The *modeling* phase various analysis and visualization techniques are applied on the prepared data to explore and model software quality dependencies represented by the data. Example analysis may investigate the probability (or frequency) of software bugs of runtime issues (user issue reports) in association with structural properties of software code and amount and type of software changes along its evolution. Figure 3 illustrates example quality analysis. Input data are gathered from multiple sources and include software code quality metrics, which represent specific product factors, and product quality data in terms of bug issues found in the software. The analysis provides two outcomes: relevancy of individual metrics as predictors of product quality and a quality model that captures quantitatively dependencies between the most relevant metrics and the product quality.

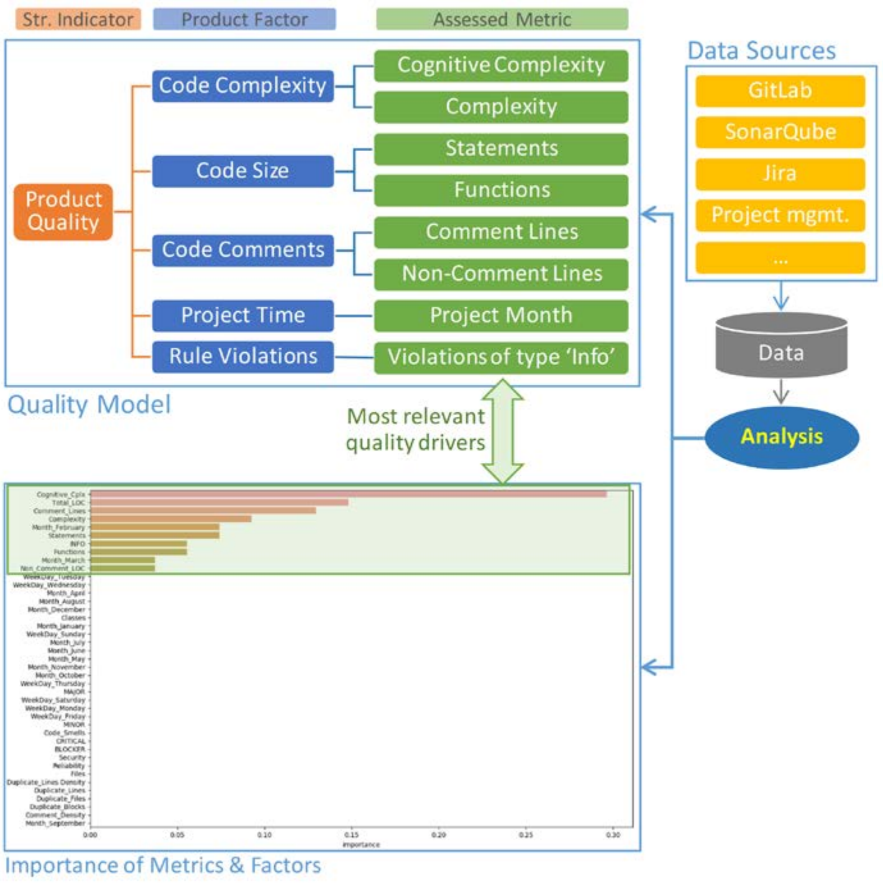

Fig. 3. Example of data-driven quality model

In the *evaluation* phase the outcomes of the data analysis phases, incl. data understanding, preparation, and modeling, are assessed from the perspective of business expectations defined in the very first business understanding phase. In our case these where to overall Q-Rapids research objectives and the specific business objectives of project partners whose software project data were analyzed in the project.

Finally, during the *deployment* phase, models (e.g., quality forecasting) will be integrated into the Q-Rapids tool to provide information to the strategies dashboard (e.g., predicted evolution of software quality or indication of the most relevant factors influencing software quality). For example, prediction models created in Python might be deployed as REST API service. As soon as new project data are available in ElasticSearch, Q-Rapids dashboard will call the Quality Model API with the required input data (metrics) and receive the answer regarding the predicted quality (forecasted number of bug issues to be found).

3 The Q-Rapids Software Development Process

Q-Rapids provides solutions for quality management in the context of Agile and Rapid Software Development. Under the umbrella of the Agile Manifesto, agile software development methods, such as Scrum and XP, are already the most popular software development approaches in industry [2]. Indeed, the tendency is towards reducing development cycles more and more to achieve a continuous software development flow, using principles from Lean thinking (e.g. lean software development [3]) and methods such as Kanban [4]. This is commonly known as Rapid software development [5,6]. However, faster and more frequent release cycles should not compromise software quality [7,8]. Indeed, quality is essential to be able to satisfy customers, which is, certainly, the ultimate goal of Agile and Rapid Software Development processes.

The literature evidences that this is not often the case, though. For example, technical debt (TD) has become a popular concept in Agile software development owing to the specific characteristics of ASD that make it prone to incurring TD [9]. Moreover, in a software development approach driven by functionality, the way in which QRs should be managed is unclear [10]. In this sense, the technical solutions provided by Q-Rapids (the Q-Rapids data gathering and analysis engine - Section 3, and the Q-Rapids dashboard - Section 5) aim to complement Agile and Rapid software development processes by incorporating three key process characteristics: *quality awareness*, *data-driven decision making* and *rapid (lightweight) development*.

3.1 Quality Awareness

In their current form, Agile methods such as Scrum, are mostly driven by functional requirements. Functional requirements are usually specified as user stories in a product backlog and prioritized using a customer perspective. This approach tends to naturally favor functional requirements over QRs [10]. As a result, quality aspects such as system security, performance or usability are often compromised [7]. The Q-Rapids software development process provides support for:

- *Continuous product quality assessment and monitoring*. Quality, which is modeled through a company specific quality model, is continuously monitored at different organizational levels. Quality related metrics provide a (almost) real time understanding on product quality to individual developers and development teams. Developers can check quality status at any time using the Q-Rapids and Kibana dashboards. This information also feeds discussions in agile ceremonies such as daily stand-up meetings or weekly (or bi-weekly) sprint meetings. In this way, quality aspects are an important part of these ceremonies, making quality a primary concern, and not an afterthought. Moreover, product quality is also continuously assessed by strategic decision makers in release planning and review meetings (e.g. once per month). Through quality-related key strategic indicators, decision makers at business level (e.g. program and product managers) can understand the implications that their decisions will have upon product and process quality.

- *Incremental elicitation of QRs*: the Q-Rapids software development process provides incremental and semi-automatic elicitation of QRs based on the continuous analysis of quality data [11]. Thus, product owners and development teams get support when defining and prioritizing quality related backlog items. QRs are explicitly included in product backlogs, decreasing the risk to overlook them during sprint planning meetings. Moreover, Q-Rapids supports the tasks of refining and improving QRs as the development progresses, using practices such as backlog grooming and sprint planning.
- *Continuous process quality assessment and monitoring*: besides product quality, Q-Rapids also offers support to continuously monitor the status of the process. Through a complete set of process metrics, Q-Rapids supports ceremonies such as Agile retrospectives, in which the way of working is discussed. Development teams can use Q-Rapids to analyze trends and process metrics values. The hard evidence reported by Q-Rapids motivates the team to find problems in order to resolve them and improve their way of working. For example, Q-Rapids can be used to identify process bottlenecks or improve estimation capabilities. Indeed, the Q-Rapids solutions and visualization of process metrics fill the current gap on tools related to processes in Agile and Rapid software development. Most existing tools focus on product quality or on continuous integration, without measures for the process (e.g. SonarQube). Basically, GitLab Time Tracker is one of the few competing solutions that could be used to analyze the process. However, Q-Rapids proposes a wider set of calculated process metrics, better visualization as well as enhanced analysis capabilities.

3.2 Evidence-based, Data-driven Software Development Process

Agile methods, such as Scrum, are founded on an evidence-based management style. Instead of making long-term predictions, Agile methods embrace a learning culture in which evidence drives decisions. However, to make accurate decisions, evidence must be reliable as well. Software analytics play a key role in this context. Agile's incremental development and extensive use of automation produce enormous amount of data that, properly used, can guide more accurate decisions. Quality related decisions in the Q-Rapids software development process are based on the insights provided by the Q-Rapids data gathering and analysis engine, which collects data from different systems such as software code repositories (e.g., Git, SVN), issue tracking systems (e.g., Jira, Mantis, Redmine), structural properties of software code (e.g., SonarQube) and runtime issue reports (e.g., Hockeyapp).

3.3 Rapid Software Development Process

A key aspect of the Q-Rapids software development process is that it supports quality management in a light-weight manner. Deploying the solution may be heavy at the beginning (e.g. defining the quality model, searching for data sources, customizing/defining connectors, etc.). However, once installed, quality support is smoothly integrated into existing agile practices such as sprint planning and review meetings, daily-stand up meetings and sprint retrospectives. Still, some extra practices and roles are needed to maintain the system up and running and to ensure that data is reliable and properly collected and analysed (e.g. data engineer).

The Q-Rapids software development process is being developed in close collaboration with the four companies participating in the consortium. It includes the use of Q-Rapids solutions in software development practices, such as coding and testing, and product management practices, such as sprint planning. It also includes the related supporting processes needed to make sure that the Q-Rapids machinery is up and running.

4 Strategic Decision Making Dashboard

The main goal of the strategic dashboard is twofold: (a) aggregating the gathered and analysed data into strategic indicators (SIs) and (b) providing extra analysis techniques that support decision-makers in their decisions.

Based on the quality factors resulting from the analysed data (see Section 2), the process metrics (see Section 3), and in collaboration with the use cases, we defined the following strategic indicators: *Blocking* (assessing when there is some problem that can alter the regular process flow, identifying potential blocking situations) [12], *Product Quality* (assessing the source code quality), *Process Performance* (assessing the fulfillment of the development process efficiently) [13], and *On-Time Delivery* (assessing the capability of fulfilling the issues planned for a specific release meeting internal and external delivery schedules) [14]. Although there is a generic definition for these strategic indicators, they must be customised in each use case to adapt them to the specific needs. For instance, for *Process Performance*, we have definitions from using two quality factors (*Testing Performance* and *Issues Velocity*) to five (*Testing Performance*, *Issues Velocity*, *Development Speed*, and *Realized Requirements*). Fig 4 includes the generic definition for the strategic indicators.

The list of strategic indicators has been extended by some use cases to support their specific scenarios. E.g., Softeam defined *Product Readiness* and *Quality Feedback Loop*, and Nokia defined *Operational Quality* and *Hardware Reliability*. In the case of Softeam, it is worth mentioning that the *Quality Feedback Loop* is a strategic indicator devoted to monitor the QRs generated by the dashboard (see below).

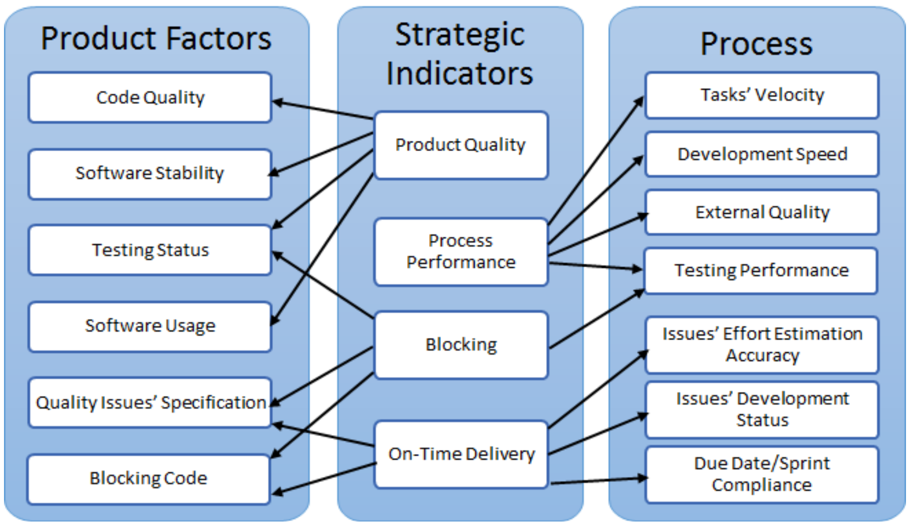

Fig. 4. Strategic Indicators

A brief description of the main features of the dashboard are as follows.

Quality assessment visualization. The dashboard includes several views to analyse the status of the SIs, i.e. indicators meant to support decision-makers to analyse the achievement of their strategic goals, such as product quality, customer satisfaction, or process performance. These SIs are defined as an aggregation of quality factors resulting from the data analysis, and these factors as an aggregation of quality metrics. The dashboard allows the decision-maker to navigate through these aggregations to have a deeper understanding of the assessment. Fig 5 depicts the different kinds of charts and the navigational path.

SI assessment. The dashboard provides two strategies to compute SIs: a quantitative approach based on computing the average of the quality factors, and a qualitative approach involving experts and historical data to define a Bayesian Network model [15]. Fig 5 shows the BN model for the *Product Quality* strategic indicator, impacted by *Code Quality*, *Stability*, and *Testing Status* quality factors. The probabilities for quality factors are computed using historical data, and for the strategic indicators we use domain experts.

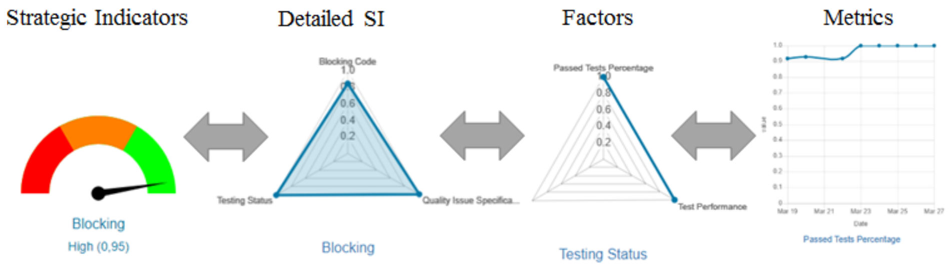

Fig. 5. Quality Assessment Navigation

Prediction. The dashboard provides several forecasting techniques that, applied over the SIs, allow decision-makers to analyse trends and behavioral patterns. Among others, it supports PROPHET, ARIMA, ETS, and THETA forecasting techniques [16, 17].

What-if analysis. Decision-makers can simulate some scenarios in order to see how different simulated values on metrics and factors would affect the assessment of their strategic indicators.

QR candidates. When the assessment values are below a given threshold, an alert is automatically raised and the dashboard identifies QR candidates from a QR patterns catalogue that, when implemented, would solve the alert [11]. For instance, if the dashboard receive an alert because the *testing performance* factor (impacting the *process performance* strategic indicator) assessment is below the defined threshold, the dashboard would suggest to consider the following QRs: (QR1) *the commit response time should be at least X%,* and (QR2) *the error correction should be at least X%*. The decision-maker can simulate the impact of each QR on the strategic indicators (see Fig. 6 for QR2). Then, the decision-maker can export the QR to the tool managing the backlog (e.g. Jira, OpenProject).

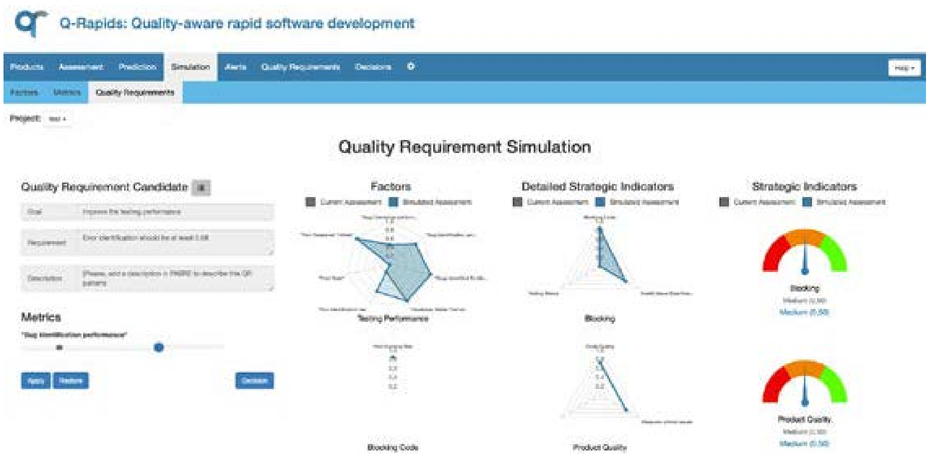

Fig. 6. QR simulation view

5 Evaluation of the Q-Rapids solution

The aforementioned Q-Rapids components (the tool support for data gathering and analysis, the software development process model, and the strategic dashboard), are being transferred and evaluated in industry. The integration of these components comprise the Q-Rapids solution, which consists of both tool support and its corresponding process model. During the technology transfer of Q-Rapids, three releases have been deployed by the four industry partners in the Q-Rapids project (Bittium, ITTI, Nokia, and Softeam[1]) within their specific development environment, where practitioners have

[1] https://www.q-rapids.eu/consortium

given feedback of the Q-Rapids solution, and used it within pilot projects for several months (since November 2018). It is worth mentioning that these companies have different profiles (one large corporation, two large/medium companies, one SME) and produce different types of systems (e.g., from modeling tools to telecommunication software).

The technology transfer and evaluation of the Q-Rapids solution follows a multi-staged process aligned with and supporting the iterative development process of the Q-Rapids components and integrated solution (see Table 2). The multi-stage evaluation process comprises two phases: formative and summative. First, the formative stage focused on supporting the evolution of concepts and ideas mainly of research work. Thus, we evaluated the first prototype and the intermediate version focusing on single components and functionalities of the Q-Rapids solution in controlled environments. We finished the formative stage with a static validation (i.e., presentation to prospective users) [18]. After the formative stage, the ongoing summative stage consists of the real use of the integrated Q-Rapids solution in under real settings of four pilot projects (i.e., dynamic validation with practitioners on-site).

- *Formative evaluation on component level*. These components have been the incremental outputs of the scientific work packages, such as components implementing an expert-based or data-driven quality model for actionable analytics, company specific software development process models, and a strategic dashboard for supporting decision-making. This formative evaluation took place at developer sites for the first release of the Q-Rapids solution, and ended with a static validation presenting the component’s capabilities to prospective users. The formative evaluation focused on technical aspects (e.g., general feasibility, scalability, and appropriateness of the gathered and visualized data). It was helpful to identify interweaved improvements of the components being developed for next releases. Examples of identified and addressed suggestions for improvement from the industrial context have been: explicitly linking the strategic indicators, quality factors, and metrics with other information sources (e.g., source code, user stories, and list of issues) in order to better support the decision making process with the help of the strategic dashboard [19], include visualization of the raw data in the quality model to facilitate decision-making [20], give a practitioner attractive support to follow of the software development process model (e.g., available on an interactive website rather than long documents[2]), and simplify the Q-Rapids solution installation and configuration process with easy deployment options such dockers [21]. Despite these suggestions for improvement, initial results have been promising in pilot projects, since participants agree on the understandability and usefulness of the Q-Rapids solution components.
- *Summative evaluation of the third and final release of the integrated Q-Rapids solution*. The summative evaluation is focusing on the application of the integrated Q.Rapids solution under the realistic circumstances of four selected projects in which the integrated Q-Rapids solution is being used. The Q-Rapids solution is be-

[2] https://www.oulu.fi/q-rapids/

ing evaluated by its impact on the selected projects where it is being used. Preliminary results helped to characterize the value provided by the solution, since Q-Rapids users have experienced benefits such as including the semi-automated functionality of creating QRs, the improvement of product quality and process performance, and an increased awareness of product readiness. Furthermore, another goal is considering suggestions for the successful commercialisation of the solution, such as looking for bilateral collaborations with industrial partners out of the Q-Rapids consortium interested in the capabilities of the Q-Rapids solution, and making effective the installation process (which currently it is one of the main barriers for adoption).

Table 2. Phases of the evaluation and technology transfer of Q-Rapids.

Characteristic	Iteration 1	Iteration 2	Iteration 3
Q-Rapids solution release	Proof-of-concept	Consolidated version	Final
Evaluation phase	Formative evaluation	Formative evaluation	Summative evaluation
Object of study	Components of the Q-Rapids solution release	Components of the Q-Rapids solution release	Integrated Q-Rapids solution as a whole
Months within the project	From month 7 to month 15	From month 16 to month 24	From month 25 to month 33
Environment	Controlled environment	Static validation (i.e., presentation to prospective users)	Dynamic validation (i.e,. pilot project using the tool)

6 Lessons learned

In [21], we have presented the most relevant lessons learned during the project on the potential adoption of Q-Rapids by practitioners, based on the experiences of the companies in the consortium. Some of them follow:

- Incremental adoption approach. Companies are advised to start using Q-Rapids in a small product first in order to understand the solution and start to grow a base of tailored connectors and a quality model fit for purpose.
- Transparency in the organizational culture. The visibility of all quality-related issues managed in Q-Rapids provides confidence to decision-makers and other involved stakeholders.
- Single access point to quality assessment. One advantage that was not really foreseen in the conception of the project is the possibility to put together lots of indicators that are normally managed through several tools.
- Tailoring to product and projects. Quality is an elusive concept that may change in every single project, even in the same organization. It is important to tailor the quality model and strategic indicators to the needs in each context.
- Expert involvement. The Q-Rapids solution requires the participation of several experts in order to get the most, from developers to implement connectors up to data scientists to analyse the collected data.

7 Conclusions

In this paper we have presented the highlights of the Q-Rapids project. We have described the three major parts of the delivered solution (data gathering and analysis; software development process with Q-Rapids; strategic dashboard) and shown the evaluation done, as well as some lessons learned. More information is available in the project website, www.q-rapids.edu. Software components are available at https://github.com/q-rapids.

At this point of time, very close to the completion of the project, we can say that we have delivered a solution that fulfils most of the original objectives of the project. However, there are many improvements that we plan to address in the near future. The implementation of machine learning approaches to fine-tune and improve the strategic indicators definition in every organization is one of the most challenging extensions. Another important topic is the better definition of cost functions for the QR patterns, which would allow to make decisions in a more informed manner.

Acknowledgements

This work is a result of the Q-Rapids project, which has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 732253.

References

1. C. Shearer: The CRISP-DM model: the new blueprint for data mining. Journal of Data Warehousing, 5 (4), 2000.
2. P. Rodríguez, J. Markkula, M. Oivo, K. Turula: Survey on agile and lean usage in finnish software industry. In: Proceedings of the ACM-IEEE international symposium on Empirical software engineering and measurement (ESEM), 2012.
3. M. Poppendieck, T. Poppendieck: Lean Software Development: An Agile Toolkit. Addison-Wesley. 2003.
4. D.J. Anderson: Kanban: successful evolutionary change for your technology business. Blue Hole Press. 2010.
5. B. Fitzgerald, K. J. Stol: Continuous software engineering: A roadmap and agenda. Journal of Systems and Software, vol. 123, 2017.
6. P. Rodríguez *et al.*: Continuous deployment of software intensive products and services: A systematic mapping study. Journal of Systems and Software, vol. 123, 2017.
7. B. Ramesh, L. Cao, R. Baskerville: Agile requirements engineering practices and challenges: an empirical study. Information Systems Journal, 20 (5), 2010.
8. L. Guzmán, M. Oriol, P. Rodríguez, X. Franch, A. Jedlitschka, M. Oivo: How Can Quality Awareness Support Rapid Software Development? – A Research Preview. In: Proceedings of Requirements Engineering: Foundation for Software Quality (REFSQ), 2017.
9. W. N. Behutiye, P. Rodríguez, M. Oivo, A. Tosun: Analyzing the concept of technical debt in the context of agile software development: A systematic literature review. Information and Software Technology, vol. 82, 2017.

10. W. Behutiye, P. Karhapää, L. López, X. Burgués, S. Martínez, A.M. Vollmer, P. Rodríguez, X. Franch, M. Oivo: Management of quality requirements in agile and rapid software development: a systematic mapping study. Submitted to IST.
11. X. Franch, C. Gómez, A. Jedlitschka, L. López, S. Martínez-Fernández, M. Oriol, J. Partanen: Data-Driven Elicitation, Assessment and Documentation of Quality Requirements in Agile Software Development. In: Proceedings of the International Conference on Advanced Information Systems Engineering (CAiSE), 2018.
12. X. Franch, C. P. Ayala, L. López, S. Martínez-Fernández, P. Rodríguez, C. Gómez, A. Jedlitschka, M. Oivo, J. Partanen, T. Raty, V. Rytivaara: Data-driven Requirements Engineering in Agile Projects: The Q-Rapids Approach. In: Proceedings of the International Workshop on Just-In-Time Requirements (JIT-RE), 2017.
13. P. Ram, P. Rodríguez, M. Oivo: Software Process Measurement and Related Challenges in Agile Software Development: A Multiple Case Study. In: Proceedings of the International Conference on Product-Focused Software Process Improvement (PROFES), 2018
14. M. Manzano, C. Gómez, C. Ayala, S. Martínez-Fernández, P. Ram, P. Rodríguez, M. Oriol: Definition of the On-Time Delivery Indicator in Rapid Software Development. In: International Workshop on Quality Requirements in Agile Projects (QuaRAP@RE), 2018
15. M. Manzano, E. Mendes, C. Gómez, C. Ayala, X. Franch: Using Bayesian Networks to estimate Strategic Indicators in the context of Rapid Software Development. In: Proceedings of the International Conference on Predictive Models and Data Analytics in Software Engineering (PROMISE), 2018.
16. S. J. Taylor, B. Letham: Forecasting at scale. The American Statistician, 72(1), 2006.
17. R. J. Hyndman, Y. Khandakar: Automatic Time Series Forecasting: The forecast Package for R. Journal of Statistical Software, 27(3), 2008.
18. T. Gorschek, P. Garre, S. Larsson, C. Wohlin: A model for technology transfer in practice. IEEE software, 23(6), 2006.
19. L. López *et al.*: Q-Rapids Tool Prototype: Supporting Decision-Makers in Managing Quality in Rapid Software Development. In: Proceedings of CAISE Forum, 2018.
20. S. Martinez-Fernandez, A. Jedlitschka, L. Guzman, A. M. Vollmer: A Quality Model for Actionable Analytics in Rapid Software Development. In: Proceedings of the Euromicro Conference on Software Engineering and Advanced Applications (SEAA), 2018.
21. S. Martínez-Fernández, A. M. Vollmer, A. Jedlitschka, X. Franch, L. López, P. Ram, P. Rodríguez, S. Aaramaa, A. Bagnato, M. Choras, J. Partanen: Continuously assessing and improving software quality with software analytics tools: a case study. IEEE Access, vol. 7, 2019.