QaSD: A Quality-aware Strategic Dashboard for supporting Decision makers in Agile Software Development

L. López[a*], M. Manzano[a], C. Gómez[a], M. Oriol[a], C. Farré[a], X. Franch[a], S. Martínez-Fernández[a], A. M. Vollmer[b]
[a]Universitat Politècnica de Catalunya, Spain
[b]Fraunhofer Institute for Experimental Software Engineering, Germany

Abstract. Software and data analytics solutions support improving development processes and the quality of the software produced in Agile Software Development (ASD). However, decision makers in software teams (e.g., product owner, project manager) are demanding powerful tools providing evidence data that support their strategic decision-making processes. In this paper, we present and provide access to QaSD, a Quality-aware Strategic Dashboard supporting decision makers in ASD. The dashboard allows decision makers to define high-level strategic indicators (e.g., customer satisfaction, process performance) related to software quality and to measure, explore, simulate and forecast the values of those indicators in order to explain and justify their decisions. Moreover, we also provide the results of a conducted evaluation of the dashboard quality in a real environment that evaluated the QaSD as usable, easy to use, with good navigation, and reliable.

Keywords: dashboard; software analytics; decision-maker; agile; forecasting; what-if analysis

1 Introduction

In recent decades, the software industry has embraced Agile Software Development (ASD) as a way to accelerate software delivery and to manage a fast-changing environment [1]. The agile way of working generates enormous amounts of data stored into different software repositories (e.g., continuous integration systems, project management tools, issue trackers). These data may be exploited using software analytics tools (e.g., SonarQube[1]) and data analytics or business intelligence solutions (e.g., Tableau[2]) to support decision makers (e.g., product owners, project managers, etc.) for improving development processes and the quality of the software produced [2], increasing the satisfaction of their customers and gaining market dominance.

In spite of the existence of such software and data analytics solutions, decision makers in ASD still need additional tool-support providing answers that they can understand more easily [3] and making use of the data in informed decisions [4]. As Godfrey stated, tools should provide answers to complex questions about development, such as "are we on schedule for next week's release?" [5]. In the same vein, Herbsleb proposed the inclusion of data mining techniques to create "powerful exploratory visualization environments that will let non-statisticians, such as developers and managers, understand the history, current state, and likely outcomes of large, complex projects" [5]. In the same line of thought, Svensson et al. show that practitioners require support for more general decision-making, as for example in requirements elicitation and requirements prioritization decisions [6].

To this aim, in this paper, we present QaSD, a modular, configurable and extensible data and expert-driven quality-aware strategic dashboard for supporting decision makers in ASD to improve their software development processes and the quality of the software produced. Concretely, the dashboard allows decision makers to define their own quality-related strategic indicators (SIs) (e.g., customer satisfaction, process performance, product quality), and bind those high-level indicators to quality factors (QFs) related to development and usage (e.g., development speed, testing status). The dashboard graphically visualizes those assessed SIs to show the current status of software products and development processes, and simulates different values of the SIs setting specific values for QFs. Besides, the dashboard supports forecasting for the values of the SIs in a time frame, in order to predict and anticipate, with traceability support, future issues in the software development process.

To guarantee the feasibility and use of QaSD, we designed the dashboard avoiding the common pitfalls in dashboard design reported by Few [7]. Moreover, we conducted a case study across four companies, involving eighteen practitioners, to evaluate the dashboard quality.

[1] https://www.sonarqube.org/
[2] https://www.tableau.com/
* corresponding autor: mailto:mllopez@essi.upc.edu

2 Background and Related Work

2.1 Background

The definition and assessment of SIs in QaSD follows a layered structure imposed by the Q-Rapids quality model [8]. This model defines SIs, QFs and metrics. SIs (e.g., Process Performance) are indicators representing the level of achievement of aspects that companies consider relevant to their software products and decision-making processes. SIs are computed from QFs (e.g., Tasks' Velocity, Development Performance) that are indicators assessing a concrete product or process quality-related aspect. At their turn, QFs are computed from metrics, which measure a specific characteristic (e.g., number of issues completely specified). For example, Fig. 1shows the layered structure of the *Process Performance* SI definition for a specific company with its corresponding QFs and metrics [9].

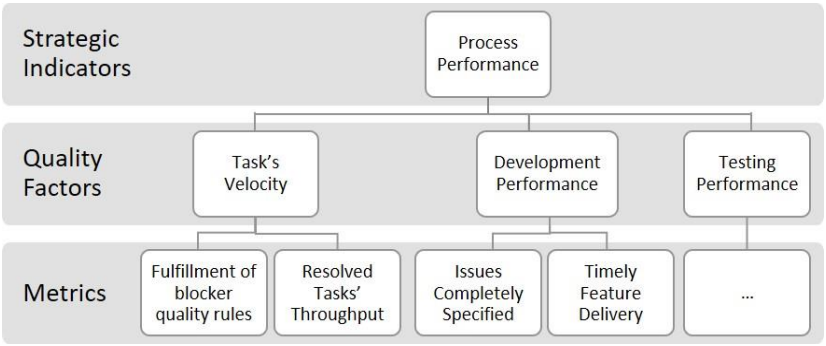

Fig. 1. Process Performance SI definition

2.2 Related Work

Dashboards are commonly used in data and software analytics solutions providing visual information to the decision makers about the status of their indicators.

Data analytics tools provide rich data visualizations that can be used by decision makers to support their decisions. According to Gartner (2017), Tableau, Microsoft Power BI[3], and Qlik[4] are market leaders in the field of data analytics tools [10]. The main functionalities of these tools are: data preparation, data discovery, and interactive dashboards for data visualization.

In the field of software engineering, software analytics tools provide data analytics features and dashboards to support software quality assessment, e.g. SonarQube, Kiuwan[5], and Bitergia[6]. These tools commonly focus on concrete quality factors. For example, SonarQube focuses on continuous code quality based on static code analysis; Kiuwan products focus on the detection of code security vulnerabilities; and Bitergia provides actionable and customizable dashboards for analyzing Open Source Software (OSS). In the context of continuous delivery, some tools supporting DevOps such as TaskTop[7], New Relic[8] and Datadog[9] are available in the market.

We can find research efforts for software analytics tools as well. Buse et. al. conducted a survey to elicit information needs for software analytics tools, such as supporting many different types of artifacts and many indicators [11]. Multiple researchers have explored building these tools on top of the well-known concept of quality models. Ulan et. al. demonstrated the practicality and usefulness of using joint probabilities and visualization for the quality assessment of software using metrics [12]. Haindl et. al. extended Quamoco to specify feature-dependent non-functional requirements and constraints for their validation [13].

Considering the above software analytics tools and related research works, the novelty of the QaSD dashboard is twofold: (a) providing the capability of defining any kind of indicator and a homogeneous visualization for them to facilitate decision-making processes; and (b) integrating forecasting and what-if analysis functionalities in a single dashboard to allow decision makers exploring the alternatives before taking decisions.

[3] https://powerbi.microsoft.com
[4] https://www.qlik.com
[5] https://www.kiuwan.com/
[6] https://bitergia.com/
[7] https://www.tasktop.com/
[8] https://newrelic.com/
[9] https://www.datadoghq.com/

3 *Software Framework*

In this section, we explain QaSD architecture and its functionality. Its code and documentation are available on GitHub[10][11]. Each component is released under its own OSS license, most of them are released under Apache 2.0, two under GPL 3.0, and one under LGPL 3.0. Appendix includes details about OSS license for QaSD components.

3.1 Dashboard Architecture

The dashboard has been designed as a modular, configurable and extensible tool. It is composed of several decoupled components capable of computing SIs from QFs, rendering the SIs assessment values through a graphical user interface and executing the supported techniques (simulation and prediction). Fig. 2 shows the top-level architecture of the dashboard whose components are described below.

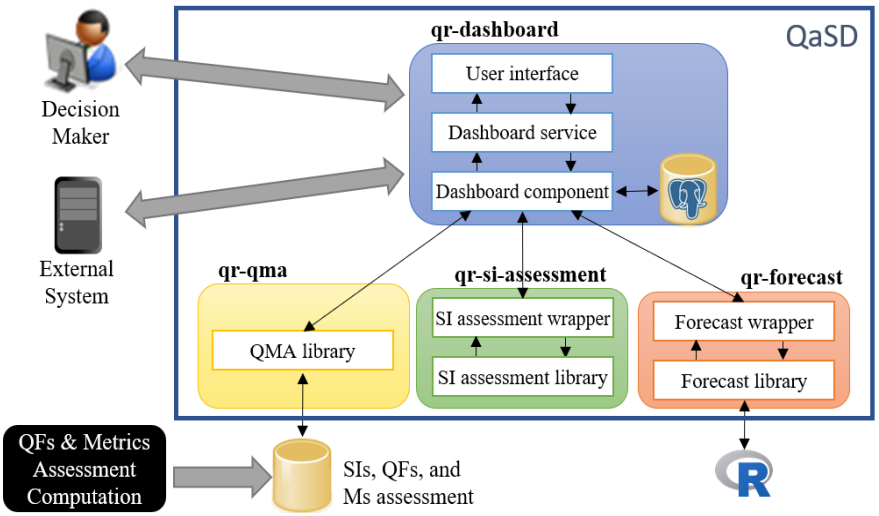

Fig. 2. Dashboard architecture

- *qr-dashboard* is a web application that orchestrates the functionalities provided by the other components and delivers them either to the decision makers through the *User Interface* or to external systems through a RESTful service provided by the *Dashboard service*. Additionally, *qr-dashboard* performs the user access control, registers the definition of SIs, assesses their values and performs simulations when the quantitative approach (see Section 3.2, functionality *Configuring SIs*) is selected.
- *qr-qma* is a software component containing the *QMA (Quality Model Assessment) library* that makes transparent to the *qr-dashboard* the distributed data sink technology used to store metrics, QFs and SIs assessments. Metrics and QFs are provided by companies whereas SIs assessments are computed by the dashboard. This component reads these values and writes SIs assessments. The available implementation of this component uses Elasticsearch[12] as data sink technology. In order to compute and visualize the SIs assessments, QFs and metrics assessments should be provided in concrete Elasticsearch indexes with concrete metadata (details are available in the documentation[13]).
- *qr-si-assessment* is a software component composed of the *SI assessment library* and the *SI assessment wrapper*. The library is responsible for the assessment of the SIs using BNs (qualitative approach). The *SI assessment wrapper* exposes the functionality provided by the *SI assessment library* as RESTful services to be used for the *qr-dashboard*.
- *qr-forecast* is a software component that includes the *Forecast library* and the corresponding *Forecast wrapper*. The library accesses an R server to compute the predicted assessment of SIs using different forecasting methods. The Forecast wrapper provides a set of RESTful services wrapping the Forecast library to be used for the *qr-dashboard*.

[10] https://github.com/q-rapids/qrapids-dashboard
[11] https://github.com/q-rapids/qrapids-dashboard/wiki
[12] https://www.elastic.co
[13] https://github.com/q-rapids/qrapids-dashboard/wiki/Elasticsearch-Indexes

3.2 Dashboard Functionalities

QaSD has been designed to support decision makers in the context of managing quality in ASD, visualising SIs assessment and providing concrete functionalities to support decision-making processes. In the following, we describe the main functionalities of the dashboard:

- *Configuring SIs.* Allows decision makers to define their own SIs, selecting the QFs used for their assessment and the approach to compute them. Two approaches are provided: (1) a quantitative approach, using a formula to compute SIs from the QFs assessment values; and (2) a qualitative approach (when it is difficult to define a formula), using estimation models based on Bayesian Networks (BNs) incorporating historical data and expert knowledge [14][15].
- *SI Assessment*. Computes the SIs values based on the configuration defined in the previous functionality. The dashboard provides a homogeneous visualization for the strategic indicators. To achieve this homogenization, the assessment is normalized, using expert knowledge, to have values in the range from 0 to 1, where 0 corresponds to the worse quality assessment and 1 to the best one.
- *Exploring SIs*. Visualizes the assessment of SIs, QFs and metrics in a graphical and textual way and provides navigation from SIs to metrics to make explicit the rationale of the assessment. The decision maker may visualize the current or historical assessments of the SIs. QaSD includes four views to show the quality model assessment: two views showing SIs assessments (*Strategic Indicator* and *Detailed Strategic Indicators*), one showing QFs assessments (*Factors*), and one showing metrics assessments (*Metrics*).
- *SI Assessment Forecasting*. Computes and visualizes the SIs forecasted values for a time frame. Several prediction methods may be selected and applied using the historical data of the QFs [16].
- *What-if analysis*. Allows decision makers to evaluate different scenarios based on the impact of QFs and metrics into the SIs. Specifically, the decision makers may set up specific values for QFs or metrics and obtain which would be the assessed value for the affected SIs.

4 Illustrative Example

This section describes some scenarios where QaSD can be used by decision makers. Assume that a company wants to assess the quality of a specific software product and its software development process. QaSD is deployed and three SIs are configured according to their needs: *Product Quality* provides insights about the quality of the developed code and *Process Performance* and *Blocking* assess the quality of its process. The SIs are defined through the dashboard user interface (see **Error! Reference source not found.**), for example *Process Performance* SI is defined as the aggregation of *Task's Velocity*, *Development Performance*, and *Testing Performance* QFs.

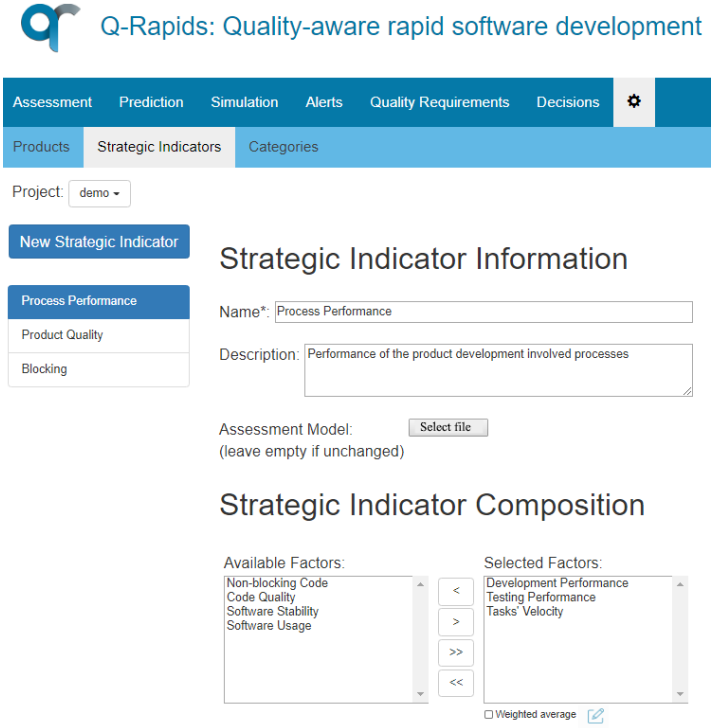

Fig. 3. Configuring SI

Scenario A: Quality Assessment

Let us assume that the release deadline for the software product is approaching. The project manager uses the dashboard to analyse the current status of the development. According to the general assessment, visualized in the dashboard main view (Fig. 4), the quality of the product and blocking tasks are in good shape (*Product Quality* and *Blocking* SIs in green[14]), but *Process Performance* SI (in the orange area) indicates that they could have some problems delivering on time.

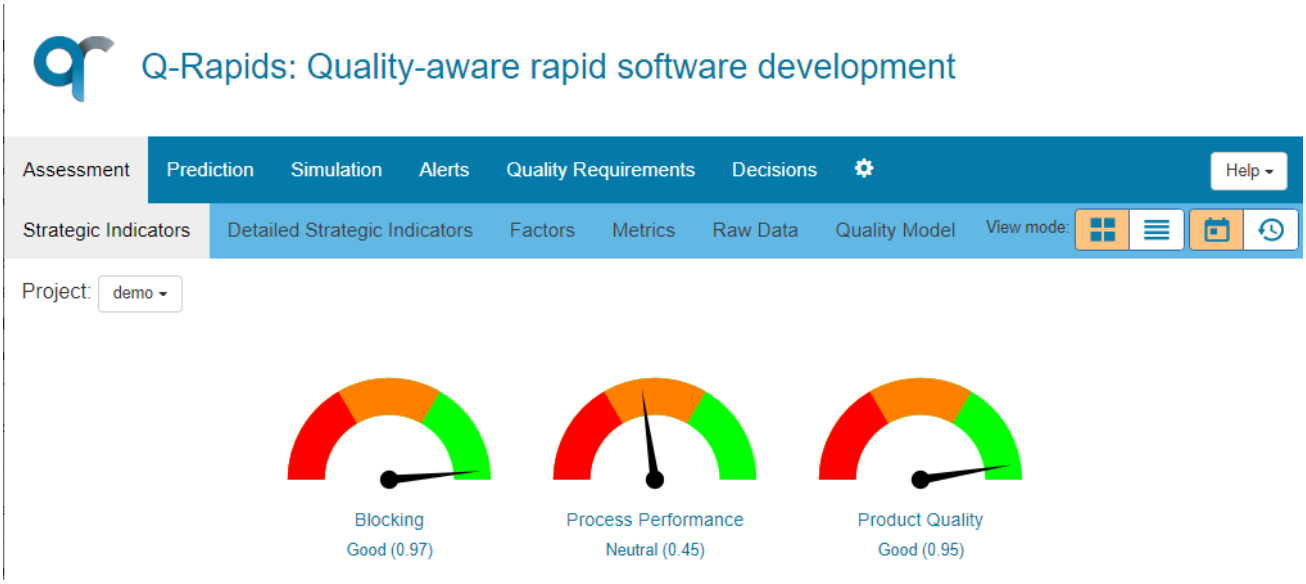

Fig. 4. Strategic Dashboard main view

The project manager navigates through the different views provided by the dashboard (Fig. 5shows fragments of those views) to identify the source of the problem. Clicking on the *Process Performance* SI name, the dashboard provides the Detailed SI view where he/she identifies that *Tasks' Velocity* QF assessment is almost 0, meaning that something is happening with tasks development. Clicking on this QF's name, he/she realises that the metrics used for its computation *(Resolved Tasks' Throughput* and *Fulfillment of blocker quality rules*) have both assessment values close to 0. The project manager decides to talk to the scrum master to identify which is the concrete problem. Using the Metrics view, they realize that *Fulfillment of blocker quality rules* has low values for all the analysed period and *Resolved Task's Throughput* is decreasing. The scrum master discovers and explains to the project manager that the low values for *Fulfillment of blocker quality rules* are due to a technical problem related to the quality rules definition, but he/she does not know the reason why *Resolved Task's Throughput* is going down. The scrum master will discuss this issue with the development team in the following scrum daily meeting to find a solution.

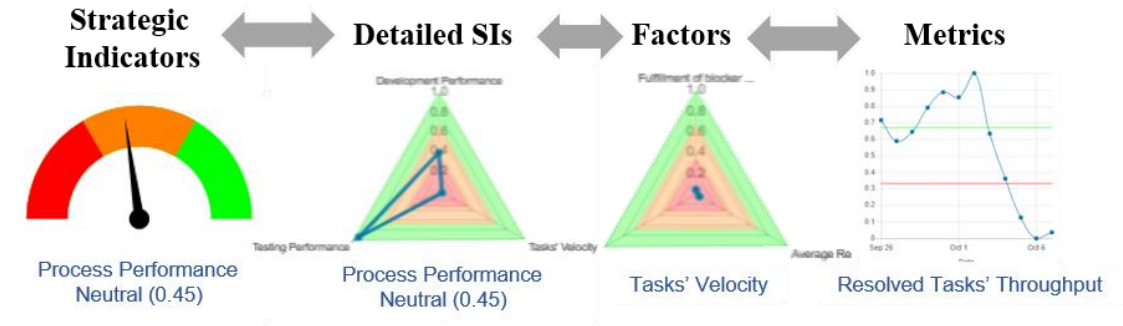

Fig. 5. Navigation Schema

Fig. 5 shows the navigation schema among the available views and the different kinds of charts used by the project manager and the scrum master to analyse the *Process Performance* SI assessment. Specifically, from left to right, the visualization of the current assessment for *Process Performance* SI (gauge chart showing the SI assessment value), Detailed SI (radar chart, QFs for the SI assessment), Factors (radar chart, metrics for the QF assessment), and historical data view for metrics (line chart, metrics assessments). The historical data views are available in all views. The dashboard also allows to visualise the data textually (in a table form).

[14] The colors and the number of areas are customizable

Scenario B: Prioritisation

Let us assume that the company wants to release a new version of the software product at the end of the following iteration but it does not know if the software product quality will be enough to release the product at that time. The project manager uses the forecasting feature provided by the dashboard to see the predicted values of the *Product Quality* SI. If those values show a positive trend, decision makers may decide to release the new version of the software product as it was planned. However, if the predicted values show that the expected product quality will not be achieved, decision makers may decide either to delay the release or, even, take actions to revert the trend, as for instance, to prioritize the quality-related tasks for the following iteration.

In the latter case, if the project manager wants to prioritize concrete quality-related tasks, he/she can use the what-if analysis (*simulation* feature) to identify which QFs or metrics should be improved to maximise que assessment of the *Product Quality* SI. Comparing different simulation scenarios, the project manager can prioritise those tasks that favours the achievement of those scenarios. Fig. 6 shows the results of one simulation scenario where the project manager changes the values using sliders on the left to set-up the scenario to be considered (i.e. concrete values for some metrics). The result is shown on the right side, the first two columns visualise how the new metric values would impact on the QFs and the last column visualise how it would impact on the SIs assessments. In the results panel, the current assessment is shown in grey, and the simulated assessment in blue. In the concrete example shown in the figure, the simulated alternative would improve *Product Quality* SI in 4%, improving also the other SIs (*Blocking*: 3%, *Process Performance* 24%).

Fig. 6. Simulation: what if a metric changes?

The strategic dashboard user's guide is available in GitHub[15], which includes links to some video tutorials.

5 *Implementation and Empirical Results*

In this section we provide details regarding the implementation of the dashboard, followed by an empirical evaluation that demonstrates the feasibility of the tool.

5.1 *Implementation*

The dashboard has been implemented in Java (1.8+) using:

- Gradle as a build automation system
- Travis CI for continuous integration.
- SonarCloud to assess the quality of the code[16].

[15] https://github.com/q-rapids/qrapids-dashboard/wiki/User-Guide

[16] https://sonarcloud.io/dashboard?id=q-rapids_qrapids-dashboard

The implementation has been documented using Spring REST Docs. Furthermore, to ensure that the documentation and the code are always in sync, all RESTful services have been documented using a Test-Driven Documentation approach[17].

Below we describe the implementation details of the dashboard components:

- *qr-dashboard* uses the Spring boot and AngularJS frameworks. It includes an external javascript library (Chart.js) to draw the graphical charts.
- *qr-qma* is implemented as a JAR library. It uses the Elasticsearch Java API to access the data sink (e.g. metrics, QFs).
- *qr-si-assessment* is implemented as a JAR library integrated in a RESTful service with Spring Boot. To compute the BNs, it uses the UnBBayes external library.
- *qr-forecast* is implemented as a JAR library integrated in a RESTful service with Spring Boot. It uses the Rserve and REngine to connect to the R server and compute the forecasting.

5.2 Empirical Evaluation

In order to explore the perceptions of practitioners on QaSD, we conducted an evaluation of the dashboard in four companies of different sizes (1 SME, 2 mid-caps and 1 corporative) and domains (model-based software development, telecommunications network, distributed systems in public safety, and risk analysis systems) during October and November 2018. Eighteen practitioners from those four companies participated in the evaluation, including six project managers, four managers or leaders, three developers, and two product owners (three people did not specify their role). They were introduced to the dashboard through a short demo describing its main functionalities (see Section 3.2). After the demo, the participants were asked to perform individual tasks and to answer a questionnaire based on their experience of using the dashboard.

In the following, there are the tasks requested to the participants for evaluating the different dashboard functionalities:

- Task I (*Exploring SIs*): Analyze the evolution of the quality of your product. If your project has suffered some quality issues during its evolution, identify these quality issues.
- Task II (*SI Assessment Forecasting*)*:* Analyze the prediction of the quality of your product and identify if your project is going to suffer some quality issues during the following days.
- Task III (*What-if analysis*)*:* (task III.1) Analyze the impact of *using* your resources to improve this quality (referring to the identified quality issues in the forecasting task); (task III.2) Analyze the impact of *not using* your resources to improve this quality, taking the chance that this factor will get the worst value possible.

The questionnaire was designed to evaluate six key quality aspects for the dashboard based on standardized constructs from [17][18][19][20], using Likert-scale rated questions (from "1: strongly disagree" to "5: strongly agree", including the option "I don't know"). Table 1 summarizes the results of the participants' answers.

Table 1. Results of the empirical evaluation

Participant's perception on...	**N**	**Mdn**	**Mode**	**Min**	**Max**
Usability	17	4	4	2	5
Ease-of-use	18	4	4	2,5	5
Navigation	18	4	4	2	5
Visualization	18	3,75	4	1,5	5
Efficiency	18	3,25	4	2	4,5
Reliability	16	4	4	2	4

Most participants ranked the dashboard as very usable, easy to use, with a good navigation and reliable. They also ranked the visualization reasonably high (e.g., "it has a clear structure", "it provides a good visual management view"), although some participants identified visualization issues that could be improved, e.g., one participant pointed out: "Spider

[17] https://github.com/q-rapids/qrapids-dashboard/wiki/Development-Guidelines

charts are not adapted to present two or less dimensions". We addressed such comments and improved the visualization of the dashboard by solving the issues identified in the evaluation (e.g. by using a triangular form to show two or less dimensions instead of spider charts).

Participants considered the dashboard slightly efficient, although some of them mentioned that the performance of the tool could be improved. To address such issues, we improved the performance of the most time-consuming functionalities of the dashboard, e.g., the forecasting functionality was improved by means of pre-fitting the forecasting models.

One common benefit reported by the users is the convenience of having a dashboard that can be used as an entry point for quality assessment. The aggregation of data, provided from several tools, gave them the possibility of defining heterogeneous strategic indicators like process performance and product readiness. The participants also highlighted the traceability of changes over time, i.e., identification of change points in order to better understand their processes. They also report that the dashboard also provides transparency and increases the awareness of the organization quality assessment.

The most significant validity threats in this evaluation are: (a) the research vias in the evaluation session (internal validity), a strict evaluation protocol was designed to minimize the thread; (b) the experience of the users using the tool, some of them had been using the tool in the previous months (internal validity); and (c) that this evaluation is based on the context of the concrete four companies. Our goal was to collect feedback from practitioners based on realistic usage. Even so we characterized the environment as realistically as possible, we cannot guarantee the generalizability of the results to other industrial settings (external validity).

The detailed information of the evaluation protocol and more detailed results for two companies can be found in [21] and the evaluation material is publicly available at https://fordatis.fraunhofer.de/handle/fordatis/123. During the formative phase of the tool, we conducted evaluations that provide us with some challenges and lessons learnt that agree with this final evaluation [8].

6 Conclusions

In this paper, we have presented QaSD, a quality-aware strategic dashboard supporting decision makers in ASD. The dashboard has been developed in the context of the Q-Rapids project, an H2020 European funded project that aims to support the software industry to increase the productivity of software development teams whilst ensuring appropriate levels of quality. Specifically, the dashboard allows decision makers to configure high-level strategic indicators (SIs), and link those SIs to their quality factors and metrics. Moreover, QaSD groups, in a single tool, functionalities to assess, explore, and forecast SIs quality assessments, complemented with the simulation of different scenarios to see how those scenarios would affect the SIs quality assessments.

An evaluation of the dashboard quality has been performed in pilot projects of four companies, where practitioners found QaSD usable, easy to use, with good navigation, and reliable. The participants highlighted that it could be used as an entry point for quality assessment, allowing the definition of heterogeneous SIs. The traceability of changes over time (historical views) allows the practitioners to identify quality changes in order to better understand their processes. It also provides transparency and increases the awareness of the organization quality assessment process. These benefits represent valuable insights for QaSD potential adopters and the software tool development industry.

As future work, we plan to incorporate new features to suggest mitigation actions when failure SIs predictions are detected. We are also interested in the improvement of the configuration capabilities of the quality model, e.g., allowing to configure factors hierarchically (i.e., using factors to compute other factors).

Acknowledgements

This work was supported by Q-Rapids (Quality-Aware Rapid Software Development) and VISDOM (Visual diagnosis for DevOps software development) projects. Q-Rapids was funded by the European Union's Horizon 2020 research and innovation programme under grant agreement nº 732253. VISDOM is funded by the Research, Development and Innovation cluster programme ITEA (project nº 17038). We thank all members of Bittium, ITTI, Nokia, and Softeam who participated in the evaluation of the dashboard. We also thank Alexandra Volkova, Aleix Balletbó, Guillem Bonet, and Oriol Martínez as main developers.

References

[1] VersionOne, 13th annual state of agile survey, http://stateofagile.versionone.com/, (2019) last accessed 4/9/2019.

[2] D. Zhang, S. Han, Y. Dang, J. G. Lou, H. Zhang, and T. Xie. Software analytics in practice. IEEE Softw., vol. 30, no. 5, pp. 30–37, 2013.

[3] V. Basili. Aligning Organizations Through Measurement: TheGQM+Strategies Approach. Springer, 2014.

[4] C. Matthies and G. Hesse. Towards using data to inform decisions in agile software development: views of available data. Int. Conference on Software Technologies, 2019.

[5] M. W. Godfrey, A. E. Hassan, J. Herbsleb, G. C. Murphy, M. Robillard, P. Devanbu, A. Mockus, D. E. Perry, and D. Notkin. Future of mining software archives: A roundtable. IEEE Software, vol. 26, n. 1, pp. 67 –70, 2009.

[6] R.B. Svensson, R. Feldt and R. Torkar. The unfulfilled potential of data-driven decision making in agile software development. Agile Processes in Software Engineering and Extreme Programming, pp. 69-85, 2019.

[7] S. Few. Information Dashboard Design: The Effective Visual Communication of Data. O'Reilly Media, 2006.

[8] S. Martínez-Fernández, A. M. Vollmer, A. Jedlitschka, X. Franch, L. López, P. Ram, P. Rodríguez, S. Aaramaa, A. Bagnato, M. Choras, and J. Partanen. Continuously assessing and improving software quality with software analytics tools: a case study. IEEE Access, vol. 7, pp. 68219-68239, 2019.

[9] P. Ram, P. Rodríguez and M. Oivo. Software Process Measurement and Related Challenges in Agile Software Development: A Multiple Case Study. Int. Conference on Product-Focused Software Process Improvement (PROFES), pp. 272-287, 2018.

[10] Magic Quadrant for Business Intelligence and Analytics Platforms. https://www.gartner.com/doc/3611117/magic-quadrant-business-intelligence-analytics, last accessed 1/4/2019.

[11] R.P.L Buse, T. Zimmermann: Information Needs for Software Development Analytics. Int. Conference on Software Engineering (ICSE). pp. 987–996, 2012.

[12] M. Ulan, S. Hönel, R.M. Martins, M. Ericsson, W. Löwe, A. Wingkvist, A. Kerren: Quality models inside out: Interactive visualization of software metrics by means of joint probabilities. IEEE Working Conference on Software Visualization (VISSOFT), pp. 65-75, 2018.

[13] P. Haindl, R. Plösch, C. Körner: An Extension of the QUAMOCO Quality Model to Specify and Evaluate Feature-Dependent Non-Functional Requirements. Euromicro Conference on Software Engineering and Advanced Applications (SEAA), pp. 19-28, 2019.

[14] M. Manzano, E, Mendes, C, Gómez, C. Ayala, and X. Franch. Using Bayesian Networks to estimate Strategic Indicators in the context of Rapid Software Development. Int. Conference on Predictive Models and Data Analytics in Software Engineering, pp. 52-55, 2018.

[15] E. Mendes, M. Perkusich, V. Freitas, and J. Nunes. Using Bayesian Network to estimate the value of decisions within the context of Value-Based Software Engineering. Int. Conference on Evaluation and Assessment in Software Engineering, pp. 90–100, 2018.

[16] M. Manzano, C. Ayala, C. Gómez, and L. López, A Software Service Supporting Software Quality Forecasting. Int. Workshop on Data-driven System Quality Assurance, 2019.

[17] V. Venkatesh and H. Bala. Technology acceptance model 3 and a research agenda on interventions. Decision Sciences, vol. 39, no. 2, pp. 273–315, 2008.

[18] V. McKinney, K. Yoon, and F. M. Zahedi. The measurement of web-customer satisfaction: An expectation and disconfirmation approach. Information Systems Research, vol. 13, no. 3, pp. 296–315, 2002.

[19] L. Goel, N. A. Johnson, I. Junglas, and B. Ives. From space to place: Predicting users' intentions to return to virtual worlds. In MIS quarterly, vol. 35, no. 3, pp. 749–772, 2011.

[20] P. Xu and B. Ramesh, Impact of knowledge support on the performance of software process tailoring. Journal of Management Information Systems, vol. 25, no. 3, pp. 277–314, 2008.

[21] A. M. Vollmer, S. Martínez-Fernández, A. Bagnato, J. Partanen, L. López, P. Rodríguez. Practical experiences and value of applying software analytics to manage quality. In Int. Symposium on Empirical Software Engineering and Measurement (ESEM), 2019, pp: 1-6.

Appendix. Required metadata

A.1 Current executable software version

Table A.1 contains the executable details for the reported version of the main component of the dashboard (qr-dashboard), the details for the other components are available at GitHub[18]. An executable release using docker is also available at GitHub. Such release includes demonstration data and it is ready to use, requiring no set-up or editing of configuration files.

Table A.1 – Executable Software metadata qr-dashboard

Nr	Software metadata description	
S1	Current software version	1.5
S2	Permanent link to executables of this version	**Docker**: https://github.com/q-rapids/qrapids-dashboard_demo-docker **WAR files:** https://github.com/q-rapids/qrapids-dashboard/releases/tag/v.1.5
S3	Legal Software License	Apache License 2.0
S4	Computing platform/Operating System	Linux, Microsoft Windows, OS X
S5	Installation requirements & dependencies	Web Server (e.g., Apache Tomcat), Oracle Java JRE, PostgreSQL, Elasticsearch
S6	If available Link to user manual	https://github.com/q-rapids/qrapids-dashboard/wiki
S7	Support email for questions	llopez@essi.upc.edu

A.2 Current code version

Table A.2 contains the code details for the reported version of the main component of the dashboard (qr-dashboard), the details for the other components are available at GitHub[18].

Table A.2 – Code metadata qr-dashboard

Nr	Code metadata description	
C1	Current Code version	1.5
C2	Permanent link to code / repository used of this code version	https://github.com/q-rapids/qrapids-dashboard
C3	Legal Code License	Apache License 2.0
C4	Code Versioning system used	git (GitHub)
C5	Software Code Language used	Java, Javascript
C6	Compilation requirements, Operating environments & dependencies	Java JDK, Gradle, Chart.js, Elasticsearch java API, Spring Boot, AngularJS
C7	If available Link to developer documentation / manual	https://github.com/q-rapids/qrapids-dashboard/wiki
C8	Support email for questions	llopez@essi.upc.edu

[18] https://github.com/q-rapids/qrapids-dashboard/wiki/Components-Version