Multidisciplinary | Rapid Review | Open Access Journal

Date of publication xxxx 00, 0000, date of current version xxxx 00, 0000.

Digital Object Identifier 10.1109/ACCESS.2017.Doi Number

Continuously assessing and improving software quality with software analytics tools: a case study

Silverio Martínez-Fernández[1], Anna Maria Vollmer[1], Andreas Jedlitschka[1], Xavier Franch[2], Lidia López[2], Prabhat Ram[3], Pilar Rodríguez[3], Sanja Aaramaa[4], Alessandra Bagnato[5], Michał Choraś[6-7], and Jari Partanen[8]

[1]Fraunhofer IESE, Kaiserslautern, Germany
[2]UPC-BarcelonaTech, Barcelona, Spain
[3]University of Oulu, Oulu, Finland
[4]Nokia, Oulu, Finland
[5]Softeam, Paris, France
[6]ITTI, Poznan, Poland
[7]UTP Univ. of Science and Technology, Bydgoszcz, Poland
[8]Bittium, Oulu, Finland

Corresponding author: Silverio Martínez-Fernández (email: Silverio.Martinez@iese.fraunhofer.de).

This work was supported by the European Union's Horizon 2020 research and innovation programme under grant agreement nº 732253 (Q-Rapids: Quality-Aware Rapid Software Development).

ABSTRACT In the last decade, modern data analytics technologies have enabled the creation of software analytics tools offering real-time visualization of various aspects related to software development and usage. These tools seem to be particularly attractive for companies doing agile software development. However, the information provided by the available tools is neither aggregated nor connected to higher quality goals. At the same time, assessing and improving software quality has also been a key target for the software engineering community, yielding several proposals for standards and software quality models. Integrating such quality models into software analytics tools could close the gap by providing the connection to higher quality goals. This study aims at understanding whether the integration of quality models into software analytics tools provides understandable, reliable, useful, and relevant information at the right level of detail about the quality of a process or product, and whether practitioners intend to use it. Over the course of more than one year, the four companies involved in this case study deployed such a tool to assess and improve software quality in several projects. We used standardized measurement instruments to elicit the perception of 22 practitioners regarding their use of the tool. We complemented the findings with debriefing sessions held at the companies. In addition, we discussed challenges and lessons learned with four practitioners leading the use of the tool. Quantitative and qualitative analyses provided positive results; i.e., the practitioners' perception with regard to the tool's understandability, reliability, usefulness, and relevance was positive. Individual statements support the statistical findings and constructive feedback can be used for future improvements. We conclude that potential for future adoption of quality models within software analytics tools definitely exists and encourage other practitioners to use the presented seven challenges and seven lessons learned and adopt them in their companies.

INDEX TERMS agile software development, case study, quality model, software analytics, software analytics tool, software quality

I. INTRODUCTION

Nowadays, a company's ability to innovate is increasingly driven by software. Digital technologies play a key role in the transformation of many industrial companies [1], especially in sectors like the automotive industry, where software (together with electronics) is responsible for over 90% of all innovations [2], [3]. In this context, software quality makes the difference and is an essential competitive factor for company success.

Companies use modern scalable data ingestion technologies (e.g., Apache Kafka[1], ActiveMQ[2]) together with data visualization and analytics technologies (e.g., Tableau[3], Microsoft Power BI[4]) to learn more about their businesses [4]. These technologies have enabled the concept of the real-time enterprise, which uses up-to-date information and acts on events as they happen [5]. This is also the case in software engineering, where **software analytics** aims at data-driven software development based on software, process, and usage data [6]–[9]. Software analytics has particular potential in the context of modern software development processes such as **Agile Software Development (ASD)** due to the incremental nature of these processes, which produce continuous sources of data (e.g., continuous integration system and customer feedback). More importantly, we can observe an increased interest in software analytics by different players, from large companies like Microsoft and Google [10], [11] to SMEs and startups offering software analytics services (e.g., Tasktop[5], Kovair[6], Kiuwan[7]) and even research projects (e.g., GrimoireLab[8], Q-Rapids[9], CodeFeedr[10]).

As reported by Forrester, companies are interested in connecting "an organization's business to its software delivery capability" by getting "a view into planning, health indicators, and analytics, helping them collaborate more effectively to reduce waste and focus on work that delivers value to the customer and the business" [12]. This is where a research gap still exists: While modern software analytics tools outperform traditional tools when it comes to gathering and visualizing data, they still fall short of using this data to create quality-related strategic indicators [13].

According to this state of the practice, we focus on the aforementioned need to connect an organization's business to its software delivery capability in the context of ASD by continuously assessing and improving quality-related strategic indicators (e.g., product quality, product readiness, and process performance) in software analytics tools. We propose using the well-known concept of **Quality Model (QM)** [13]–[16] in software analytics tools as the instrument to bridge the gap between low-level quality concepts related to development and usage and high-level quality-related strategic indicators. Therefore, **the goal of this research is to understand practitioners' perception regarding the integration of QMs in software analytics tools in ASD companies in order to effectively assess and improve quality-related strategic indicators.**

Since we address the perception of practitioners, our aim was to conduct an empirical study in industry. To this end, we needed a tool implementing the concept we wanted to evaluate. Given that no such tool is freely available on the market, we opted for the Q-Rapids tool. The Q-Rapids tool, developed as part of the Q-Rapids project [17], offers software analytics capabilities that integrate QMs to assess and improve software quality in the context of ASD. Its main functionalities are: (a) real-time gathering of various types of data related to the development and usage of a software system, which are the input for a QM; (b) real-time modeling of this data in terms of a QM in order to reason about aggregated quality-related strategic indicators; (c) presenting this QM and data to decision makers in a multi-dimensional and navigational dashboard for use during ASD events, such as sprint planning or daily stand-up meetings.

In order to understand the practitioners' view regarding our goal, we conducted **a case study across four companies, involving 26 practitioners, to investigate the following key aspects related to the Q-Rapids tool**:

- Its understandability, reliability, and usefulness to assess software quality;
- Its level of detail, relevance, perceived usefulness, and behavioral intention to improve software quality;
- Its challenges and lessons learned, as seen by adopters in realistic settings.

This paper is structured as follows. Section II presents the background and related work. Section III presents the object of study in this case study across four companies: the integration of a QM within a software analytics tool (i.e., the Q-Rapids tool). Moreover, it provides details on how the tool can be used to assess and improve software quality. Section IV describes the research methodology of our case study. Section V presents the results with respect to the participants' perceptions on exploring the tool to assess and improve software quality. Section VI discusses challenges and lessons learned as seen by adopters in ASD settings using the Q-Rapids tool. Section VII reports the limitations of this work. Finally, Section VIII concludes the article and presents future work.

II. BACKGROUND AND RELATED WORK

The objective of this section is twofold: (a) to provide a background on QMs, software analytics, and ASD; and (b) to discuss related work on software analytics tools in ASD companies.

[1] https://kafka.apache.org/
[2] https://activemq.apache.org/
[3] https://www.tableau.com/
[4] https://powerbi.microsoft.com
[5] https://www.tasktop.com/
[6] https://www.kovair.com/
[7] https://www.kiuwan.com/
[8] https://chaoss.github.io/grimoirelab/
[9] https://www.q-rapids.eu/
[10] http://codefeedr.org/

A. BACKGROUND

Quality is defined by ISO 8042 [18] as "the totality of characteristics of an entity that bear on its ability to satisfy stated and implied needs". This definition is too abstract to be operationalized directly, and is one of the reasons why there has been a multitude of software **QMs** proposed in the last four decades (e.g., [19]–[23]) that refine high-level concepts of "quality" (like reliability or efficiency) down to the level of metrics (as number of bugs or response time). One popular example widely adopted in industry is the ISO/IEC 25010 standard [14], which determines the quality aspects to be taken into account when evaluating the properties of a software product. Two more recent examples well known in industry are Quamoco and SQALE [24]. Quamoco [16] is a QM integrating abstract quality aspects and concrete quality measurements. SQALE [25] computes technical debt indexes based on the violations of quality rules in the source code. Other works propose refactoring models to improve a particular quality aspect of the existing model or code, and different extensions to the traditional concept of quality (such as incorporation of non-technical criteria into ISO/IEC 9126-1 QM [26]).

Software analytics is defined as "analytics on software data for managers and software engineers with the aim of empowering software development individuals and teams to gain and share insight from their data to make better decisions" [6]. In this context, analytics results should include some actionable knowledge that can be used to improve software quality. Software analytics is used in various scenarios to assess concrete problems, e.g., use of process data to predict overall project effort [27], use of security data to identify indicators for software vulnerabilities [28], or classification of app reviews [29]. In this context, **software analytics tools** provide features for specifically visualizing software development aspects. In our view, some notable examples are SonarQube[11], Kiuwan, Bitergia[12], and Tasktop, all of which have been discussed in related work.

ASD relies on short feedback cycles as a way to provide flexibility and rapid adaptation to market fluctuations. In this context, decisions are also made more frequently. For instance, product releases and the related decisions take place in short intervals, instead of months/years as was the case with traditional software development approaches such as the waterfall model. Popular agile practices such as continuous integration [30] provide a tight connection to development to ensure errors are detected and fixed as soon as possible. The current tendency to shorten feedback cycles even further (e.g., continuously getting feedback from customers and being able to react on that) enhances the potential for software analytics. Continuous software engineering [31], which has its roots in ASD, represents a holistic approach to providing tight connections among all software development activities, including not only integration but also aspects such as business and development (BizDez) and development and operations (DevOps).

Our work aims to further understand the challenges limiting industrial uptake of QM-based software analytics tools in ASD companies, as seen by practitioners. In our research, we build on previous research on software quality modeling, integrating a QM into a software analytics tool to assess and improve software quality (see Section III).

B. RELATED WORK

Software analytics plays a major role in ASD and continuous software engineering since, properly used, the enormous amounts of data from different systems (e.g., continuous integration system, management tools, etc.) have proved increasingly useful for guiding (almost) real-time decision-making [32]–[34]. Indeed, companies like Microsoft are hiring data scientists for their software teams [35]. ASD and continuous software engineering have created numerous new opportunities for observing user behavior and monitoring how systems and services are being used; for identifying unexpected patterns and runtime issues; for monitoring system quality attributes; and for collecting real-time data to feed both business and technical planning [36], [37]. The main objective is to constantly monitor and measure both business indicators and infrastructure-related metrics in order to facilitate and improve business and technical decision-making [38]. In the specific case of software quality, continuous quality monitoring enables transparency to foresee, trace, and understand important aspects of product quality in real time [38], [39]. Support for product quality management is particularly relevant in ASD, since it tends to overlook quality aspects in favor of product functionality [17]. Although the literature reports on promising advances regarding the use of analytics in continuous software engineering [32]–[34], many challenges still exist, with the lack of software analytics tools being among the most relevant ones [33].

When we look for software analytics tools typically used in ASD, we find a large number of commercial and academic software analytics tools that are available on the market. Examples of commercial tools focusing on concrete quality aspect are SonarQube, Kiuwan, Bitergia, New Relic[13], Datadog[14], and Taktop. SonarQube focuses on continuous code quality based on static code analysis, including assessment of code smells, bugs, and vulnerabilities. Kiuwan products focus on the detection of code security vulnerabilities, also offering a tool specifically for Open Source Software (OSS) risk analysis. Bitergia provides actionable and customizable dashboards for analyzing OSS. New Relic allows developers to install monitoring agents and gather real-time insights from users such as software failures

[11] https://www.sonarqube.org/
[12] https://bitergia.com/
[13] https://newrelic.com/
[14] https://www.datadoghq.com/

and performance improvements. Datadog monitors cloud-scale applications; provides monitoring of servers, databases, tools, and services; and supports the full DevOps stack. Tasktop aims to integrate and coordinate value streams across the DevOps stack. In addition to these commercial software analytics tools, it is worth mentioning value stream management tools for capturing, visualizing, and analyzing critical indicators related to software product development [12]. Some of these tools even strongly support the creation of new dashboards from data and advanced data analytics (e.g., machine learning). With respect to tools in academia, we find CodeFeedr [40], Perceval [41], and SQuAVisiT [42]. CodeFeedr is a real-time query engine. Perceval performs automatic and incremental data gathering from almost any tool related to open source development, which can be used in Bitergia dashboards. SQuAVisiT is a generic framework supporting maintainability assessment.

Based on the aforementioned software analytics tools, we can see that the data analytics trend has had a great impact on software engineering as well (i.e., software analytics), particularly in the short feedback cycles of ASD and continuous software engineering. However, application of QMs in the software analytics tools used in industry is not widespread, despite the need for "a view into planning, health indicators, and analytics, helping them collaborate more effectively to reduce waste and focus on work that delivers value to the customer and the business" [12]. A notable exception is the SQALE QM within SonarQube. However, its limitation is that its main functionality of static analysis of the source code is limited to a single data source (i.e., the source code). We can conclude that the aforementioned software analytics tools do not offer full capabilities, including an integrated QM, to provide quality-related strategic indicators by using software analytics results. Therefore, at the beginning of the Q-Rapids project, we decided to build the Q-Rapids tool, which integrates highly customizable (instead of pre-defined) QMs.

Some researchers have investigated the success factors as well as the needs and challenges related to the application of software analytics in industry [6], [43]. Huijgens et al. conducted an empirical study to identify success and obstruction factors regarding the implementation of software analytics in the context of continuous delivery as a service (e.g., defining and communicating the aims upfront, standardizing data at an early stage, and building efficient visualizations). Buse and Zimmermann proposed several guidelines for analytics tools in software development (e.g., ease-of-use and measurement of many artifacts using many indicators). In our work, we focus on the key aspects needed for practitioners to adopt QMs integrated into software analytics tools in the context of ASD.

The novelty of our work is twofold: (a) Based on input from practitioners, it explores the integration of QMs into software analytics tools in ASD companies, leveraging software, process, and usage data; and (b) it provides an understanding of what is needed in the aforementioned tools to enable them to be widely accepted in industry as well as challenges and lessons learned from their adoption in four ASD companies.

III. INTEGRATING A QUALITY MODEL INTO A SOFTWARE ANALYTICS TOOL

This section includes: (a) the description of the QM we propose in our work; (b) the Q-Rapids software analytics tool integrating the aforementioned QM; and (c) how the Q-Rapids tool can be used to assess and improve software quality in ASD.

A. THE Q-RAPIDS QUALITY MODEL

Both academic and industry partners of the Q-Rapids project have iteratively created a QM for software analytics, whose main characteristic is that it offers tool-supported customization (integration into software analytics tools) to define strategic indicators related to the quality of software product and development processes based on company needs. This QM aims at enabling decision makers to improve identified quality deficiencies.

In the following, we will present the elements of the QM and how these elements are computed from automatically ingested raw data. For details on the initial creation and previous iterations, the reader is referred to [44], [45].

Following common approaches, we defined the QM for the Q-Rapids tool as a hierarchical structure composed of five types of entities, each of them serving a well-defined purpose (see Figure 1):

- Strategic Indicator - Quality-related aspect that a company considers relevant for its decision-making processes. It represents an organization-wide goal, e.g., product quality or process performance. A strategic indicator can be influenced by numerous product/process factors.
- Product/Process Factor - The attributes of parts of a product/process that are concrete enough to be measured [16], e.g., code quality, issue velocity, delivery performance. A product/process factor may be composed of numerous assessed metrics.
- Assessed Metric - Concrete description of how a specific product/process factor is quantified for a specific context, e.g., code duplication, test success.
- Raw Data - Data that comes from different data sources without undergoing any modification, e.g., files, unit tests, pending issues. Typically, this data cannot be decomposed further into smaller units.
- Data Source - Each of the different tools the companies use for extracting raw data, e.g., SonarQube, Jira, Jenkins, Gerrit.

Figure 1 shows an excerpt of the elements of the Q-Rapids QM. A detailed QM (i.e., with definitions of all elements and formulas of the assessed metrics) is available in the Appendix.

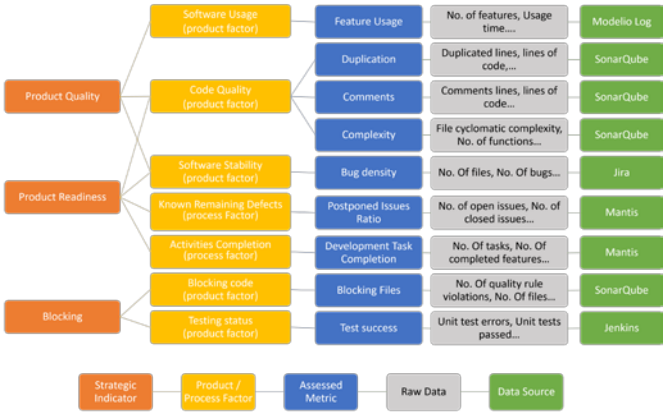


FIGURE 1. **Excerpt of the generic Q-Rapids QM (details in Appendix).**

To enable assessment of continuously updated strategic indicators, there is a bottom-up calculation process, starting from the data sources, which is detailed in [44].

B. THE Q-RAPIDS SOFTWARE ANALYTICS TOOL

The Q-Rapids tool[15] provides continuous assessment, which could be real-time, of the strategic indicators to decision makers based on the Q-Rapids QM. Figure 2 shows an excerpt of the conceptual architecture of the Q-Rapids tool, depicting its modules related to data analytics capabilities and its data flow, which adopts the idea of the lambda architecture approach used for Big Data solutions [46]. The main modules of the tool are *Data Gathering*, *Data Modeling and Analysis*, and *Strategic Decision Making*.

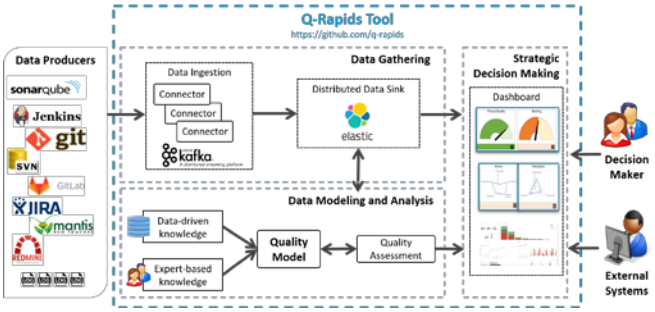


FIGURE 2. **Q-Rapids software analytics tool architecture.**

The *Data Gathering* module is composed of different Apache Kafka connectors to gather data from heterogeneous external data sources, such as static code analysis (e.g., SonarQube), continuous integration tools (e.g., Jenkins), code repositories (e.g., SVN, Git, GitLab), issue tracking tools (e.g., Redmine, GitLab, JIRA, Mantis), and software usage logs.

The *Data Modeling and Analysis* module uses the gathered data and the QM definition to assess software quality. The QM is highly customizable to support any company's needs. For instance, users of the Q-Rapids tool can define new strategic indicators, product/process factors, assessed metrics, and their relationships, as well as the frequency of execution of the QM assessment (e.g., daily, hourly). Another example: Since the QM works as a plug-in, it can also be defined from data mining and analysis.

The *Strategic Decision Making* module is responsible for the end-user interface and provides two different types of dashboards: (a) the *Strategic Dashboard*, providing homogeneous visualization for assessing the higher-level elements of the QM; and (b) the *Raw Data Dashboards*, providing specific visualizations for the data gathered from the source tools.

On the one hand, the *Strategic Dashboard* supports decision makers in their decisions related to the assessed quality of their products (e.g., does our product have the quality to be released?). It uses the same kind of charts for visualizing the assessment of the three most abstract levels of the QM (strategic indicators, factors, and assessed metrics). These most abstract levels (strategic indicators, factors, and assessed metrics) work like "traffic lights", i.e., red stands for risky quality, orange for neutral, and green for good quality. These generic visualizations unify the quality visualization and hide the heterogeneity of the data and source tools. This also allows navigating through the different elements, which provides traceability and helps to understand the assessment.

On the other hand, the *Raw Data Dashboards* allow decision makers to take concrete actions to address a particular issue and improve quality (e.g., which concrete files are too complex?). They are customized for concrete QM elements, e.g., the metric *Blocking files* used in the *Blocking* strategic indicator. Therefore, they allow the user to link the quality issue (e.g., *Non-Blocking files*) to the concrete elements from the data source tools (concrete files violating the quality rules).

C. THE Q-RAPIDS TOOL FOR ASSESSING AND IMPROVING SOFTWARE QUALITY

This section describes two scenarios where the Q-Rapids tool can be used by decision makers to assess or improve the quality of their products. We follow the definitions of "assessing" and "improving" software quality specified by Kläs et al. [23]. Assessment refers to the process in which "a concept is quantified, measured and compared to defined evaluation criteria to check the fulfillment of the criteria" [23]. Improvement refers to the process in which "relevant factors (i.e., variation factors) influencing the concept of interest are known. Therefore, the concept of interest can be improved by improving its influencing factors" [23].

1) ASSESSMENT SCENARIO WITH Q-RAPIDS

The (infamous) ACME company needs to deliver the next release of one of their higher-quality products, *NeverLate*, on time. Therefore, Bob, the product manager in charge of

[15] The tool source code and documentation are available at: https://github.com/q-rapids

NeverLate, decides to use the Q-Rapids tool in order to assess and monitor *Blocking* situations[16].

At the beginning of the sprint, Bob receives an alert because the *Blocking* strategic indicator has dipped below some predefined threshold. The Q-Rapids tools visualizes this event by changing the traffic light related to this indicator from green (good quality) to orange (neutral) in the *Strategic Indicators View* (Figure 3, needle in the radar chart on the left). He delves deeper into the QM to analyze the situation. Taking up the QM, the *Blocking* strategic indicator is impacted by two product factors: *Blocking code* and *Testing status*. The *Detailed Strategic Indicators View* reveals that the *Testing status* assessment is good, while the *Blocking code* assessment is not so good (value around 0.5 on the radar chart's corresponding axis). Bob explores the assessed metrics impacting this factor by using the *Factors View*, where the *Non-blocking files* metric has a low assessed value (below 0.5 on the corresponding axis of the radar chart). Finally, using the *Metrics Historical View* (the line chart on the right), Bob verifies that the *Non-Blocking files* metric has been deteriorating since the last sprint. Based on this *assessment*, Bob decides to *improve* the software quality by tackling the *Blocking files* problem.

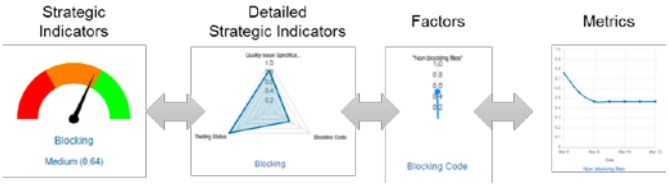


FIGURE 3. **Strategic Dashboard navigation schema.**

2) IMPROVEMENT SCENARIO WITH Q-RAPIDS

Following the *Blocking* problem above, Bob calls a meeting with Jane, a senior developer working on *NeverLate*, to discuss concrete actions to improve the *Blocking files* problem. Using the concrete *Raw Data Dashboard* corresponding to *Non-blocking files*, Joe and Jane learn that in the last sprint, the classes of a specific directory were changed many times by the same developer, Brandon (Figure 4, list of blocker and critical issues at the bottom). Moreover, the classes contain five blocker quality rule violations regarding code smells. Raw data visualization offers actionable analytics to refactor the classes of the problematic directory, clearly indicating which classes have been heavily modified and explaining the violated quality rules. Consequently, Bob could take the concrete action of adding a new issue to the backlog so that Brandon can solve these problems. Table I gives a summary of the actions that can be taken to improve software quality based on *Blocking*. It also indicates at which point in time during the ASD process the actions can be taken.

[16] "Blocking", as defined and customized by the companies of our case study, refers to situations that negatively influence the regular progress of the software development.

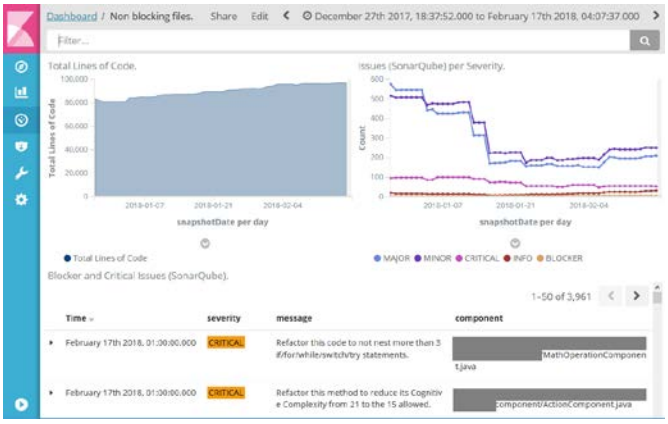


FIGURE 4. ***Raw Data Dashboard*** **for blocking and critical files.**

TABLE I
EXAMPLES OF ACTIONS TO IMPROVE SOFTWARE QUALITY FOR THE "BLOCKING" STRATEGIC INDICATOR

Factors	Action Points	When in ASD?
Blocking code	Resolving blocker quality rule violations or refactoring highly changed files (e.g., God objects or configuration files)	Refactoring-related tasks are included in the product backlog and selected during sprint planning. Developers continuously perform refactoring.
Testing status	Improving tests that do not detect critical bugs during development. Improving the performance of the test pipeline.	Test suite improvement tasks are included in the product backlog during sprint planning and assigned to testers based on their priority.

IV. EVALUATION METHODOLOGY

This section reports the methodology of our case study.

A. RESEARCH SETTING

Our research context consisted of four pilot projects in the four companies participating in the Q-Rapids project. As reported in Table II, the companies ranged from small and medium-sized enterprises with up to ten products on the market to very large corporations with more than 1,000 products on the market. They develop products using ASD for various application domains such as telecommunications, security, military, transport, health, and public administration.

The companies' pilot projects had three essential things in common that made it possible to run a case study [47] across them: They had an ASD (Scrum-like) way of working; the stakeholders (i.e., product owners, managers, and developers) were interested in having tool support to assess and improve software quality in their setting; and the Q-Rapids tool presented in Section III was deployed and used by these stakeholders in such settings.

TABLE II
SETTING OF THE FOUR COMPANIES' PILOT PROJECTS

Setting	Company 1	Company 2	Company 3	Company 4
Id	UC1	UC2	UC3	UC4
Product	Tool for model-based software development	Distributed system in telecommunication networks	Distributed system in public safety	Risk analysis system
Context	Single long lifetime software product line	Multiple product lines	Multiple product lines	Multiple software products
Product size	[KLOC] > 1000	500 – 1000 [KLOC]	1 – 100 [MLOC]	[KLOC] > 200
# of releases	Up to 100	13	16	1
1st release	1991	2011	2013	2017
ASD since	2006	2011	2005	2017
Programming language	Java, Eclipse RCP, JEE for Web tools	C	Java, C/C++	Java, JavaScript, HTML5/CSS3

In the following, we will briefly describe the context of the four use cases. One common characteristic is that the four companies had already used Q-Rapids in earlier pilot projects than the ones evaluated in this case study (i.e., UC1, UC2, UC3, and UC4 from Table II).

UC1. Company 1 used Q-Rapids in its main product Modelio 3.8 and in Modelio NG (UC1), which is a customization for a customer. Modelio is the latest generation of a 25-year-old product line of a model-driven tool suite dedicated to expressing and managing requirements, modeling software architectures, building accurate UML models, generating a full range of documentation, and automating application code production for several languages. In UC1, Company 1 used Q-Rapids with the aim of improving the quality of Modelio NG by leveraging QRs during the development phase of new versions of the product line. This involved the early detection of anomalies, which helped to improve their ability to release the product on time by reducing the number of anomalies discovered during the pre-release validation phase. Company 1 used its experts' knowledge to identify new strategic indicators not provided directly by the Q-Rapids tool.

UC2. Company 2 deployed Q-Rapids in two different contexts, applying two different strategies. For case one, Company 2 used a kind of research-oriented approach, where the research partners facilitated or collaborated on multiple sessions to define the QM and the strategic indicators based on high-level strategic goals, to derive factors and metrics, and to identify relevant data sources. This was done to realize a proof of concept. Company 2 assumed that it had learned the necessary methods and deployment process in such detail that they could replicate it on their own in another use case (i.e., UC2). In UC2, Company 2 decided to use their expert knowledge to identify interesting strategic indicators not provided by other tools to be presented from different viewpoints by the Q-Rapids tool. The reported challenges and lessons learned are based mainly on the deployment experience with UC2.

UC3. Company 3 implemented the solution in two contexts. For the first context, the focus was on the use of a proof of concept of the Q-Rapids solution by a production test software team in the public safety domain. The research partners facilitated or collaborated on multiple sessions to determine the strategic indicators as well as process factors and metrics. This included the company's internal development of the software lifecycle development process as well as the identification of necessary data sources and alignment of the development tool chain with the Q-Rapids solution. For the second context, identified as UC3, Company 3 expanded the approach to a multi-context information tool development project related to product development, manufacturing, and maintenance. In comparison to the first context, UC3 involved several teams. The identified improvement issues regarding development tool chains, metrics, and the Q-Rapids solution from the first context were very useful for UC3.

UC4. Company 4 first (during the proof of concept phase) used the solution in almost finished projects in the maintenance phase, and in the software part of a research (non-commercial) project, where more experiments and a research-oriented approach were possible. After those initial tests, Company 4 decided to use the solution in practice to measure quality- and process-related metrics in the largest, most active, and most important commercial software product deployment project, here referred to as UC4 (where the solution is still being used by the Product Owner).

B. RESEARCH QUESTIONS

As stated in Section I, our research goal was to understand practitioners' perception of the integration of QMs into software analytics tools in ASD companies in order to effectively assess and improve quality-related strategic indicators. We split this research goal into three Research Questions (RQs):

- RQ1. What is the practitioners' perception regarding the integration of QMs into software analytics tools when *assessing* software quality in ASD?
- RQ2. What is the practitioners' perception regarding the integration of QMs into software analytics tools when *improving* software quality in ASD?
- RQ3. What are the challenges and lessons learned that practitioners face when integrating QMs into software analytics tools for *assessing* and *improving* software quality in ASD?

With RQ1, we investigated the *assessment* scenario presented in Section III, while RQ2 focused on the *improvement* scenario presented. Transversal to the assessment and improvement scenarios, RQ3 was used to investigate the challenges and lessons learned as seen by

adopters in realistic settings. RQ3 was addressed after RQ1 and RQ2 (see details in Section IV.C).

For the assessment and improvement scenarios (i.e., RQ1 and RQ2), we specified relevant sub-RQs for practitioners (see Table III). First, we considered that a QM within a software analytics tool for assessing software quality needs to be *understandable*, *reliable*, and *useful* in order to be used by practitioners, based on our experiences with the proof of concept of the Q-Rapids project (see [48]). Second, we considered that a QM within a software analytics tool for improving software quality has to contain the *right level of detail;* it has to provide *relevant* and *perceived useful* actions; and practitioners should *intend* to use it.

Table III shows the mapping between the sub-RQs and the constructs used to address them. Examples of constructs are 'Perceived usefulness' and 'Behavioral intention' [49]. The table has four columns: (a) an identifier for the sub-RQ; (b) the sub-RQ; (c) the construct that the sub-RQ is dealing with; and (d) the literature used to define the construct.

TABLE III
SUB-RESEARCH QUESTIONS. EACH SUB-RESEARCH QUESTION IS MAPPED TO A CONSTRUCT.

Id.	Sub-research question	Construct	Reference of the construct [a]
RQ1.1	Do practitioners find QMs within a software analytics tool understandable when using them to assess software quality?	Understand-ability	McKinney et al. 2002: [50]
RQ1.2	Do practitioners find QMs within a software analytics tool reliable when using them to assess software quality?	Reliability	McKinney et al. 2002: [50]
RQ1.3	Do practitioners find QMs within a software analytics tool useful when using them to assess software quality?	Usefulness	McKinney et al. 2002: [50]
RQ2.1	Do practitioners find QMs within a software analytics tool traceable when using them to improve software quality?	Right level of detail	Goodhue and Thompson 1995: [51]
RQ2.2	Can practitioners take relevant actions when using QMs within a software analytics tool to improve software quality?	Relevance	Lee and Strong, 2003: [52]
RQ2.3	Can practitioners take usable actions when using QMs within a software analytics tool to improve software quality?	Perceived usefulness	Venkatesh and Bala, 2008: [49]
RQ2.4	Do practitioners intend to use a QM within a software analytics tool to improve software quality?	Behavioral intention	Venkatesh and Bala, 2008: [49]

[a] All the constructs from the selected references have already been validated in practice.

1) HYPOTHESIS REGARDING THE INTEGRATION OF QMs WITHIN SOFTWARE ANALYTICS TOOLS

The main hypothesis of this work is that QMs within software analytics tools can be used by practitioners to assess and improve software quality. This main hypothesis was refined into seven sub-hypothesis, one for each sub-RQ:

- Practitioners are able to *understand* a QM within a software analytics tool to assess software quality (H1: understandability).
- Practitioners find that using a QM within a software analytics tool is *reliable* for assessing software quality (H2: reliability).
- Practitioners find that using a QM within a software analytics tool is *useful for* assessing software quality (H3: usefulness).
- Practitioners find the traceability between abstract quality aspects and raw data of a QM within a software analytics tool to be at the *right level of detail* to improve software quality (H4: right level of detail).
- Practitioners are able to take *relevant actions* to improve software quality based on the information provided about a QM presented in a software analytics tool (H5: relevance).
- Practitioners are able to take *perceived useful actions* to improve software quality based on the information provided about a QM presented within a software analytics tool (H6: perceived usefulness).
- Practitioners *intend to use* a QM within a software analytics tool to improve software quality rather than using a human approach (H7: behavioral intention).

C. RESEARCH DESIGN AND SAMPLING

This study was initiated by the Q-Rapids project members. The authors were organized into two teams. The first seven authors constituted the researcher team. The researcher team was composed of the leading team (first three authors from Fraunhofer IESE) and another four authors from two universities (Technical University of Catalonia and University of Oulu). The last four authors, from the four companies, constituted the practitioner team. Their responsibility was to use Q-Rapids in the setting described in Section IV.A. In addition, there were 22 other participants in the study from the four companies. In the following, we will briefly discuss the research design, sampling, and execution.

Design: The first two authors created an evaluation guideline with two objectives: (a) to provide experimenters and observers with a detailed description of the evaluation procedures and guidance for managing potential confounding factors; and (b) to ensure equal treatment between different evaluation steps independent of the experimenter. These guidelines included the design of our evaluation, with two key researcher roles: an experimenter and at least one observer. On the day of the evaluation, the researcher team performed the following steps at the premises of the four companies:

(1) The experimenter explained the evaluation goals and the procedure to the participants and asked them to sign the informed consent.

(2) The experimenter performed a live demo in the company setting to introduce the software quality assessment and improvement scenarios of the Q-Rapids tool. Showing the same live demo at all four companies served to ensure equal treatment and to reduce experimenter bias. At the end of the training, the experimenter asked the participants to clarify any doubts about the use of Q-Rapids tool before advancing to the next step in the evaluation.

(3) The participants individually explored the Q-Rapids tool by working on assigned tasks. The experimenter encouraged them to use a form to write down positive and negative aspects of the Q-Rapids tool. This served to get a better understanding of the participants' perceptions regarding the Q-Rapids tool.

(4) The experimenter collected responses to our sub-RQs by using a structured questionnaire based on the constructs of Table III, answered individually. These constructs had already been validated in practice in terms of item reliability score of the construct or Cronbach's alpha, which are objective measures of reliability [53]. Reliability is concerned with the ability of an instrument to measure consistently [53]. All selected constructs were reported to have an item reliability score greater than 0.8. Each construct included up to four items to be rated using a five-point rating scale from 1 "strongly disagree" to 5 "strongly agree" (where 3 was "neutral") and an additional "I don't know" option. We instantiated the selected questions according to the purpose and content of the assessment and improvement usage scenarios of the Q-Rapids tool. This served to collect the practitioners' perception on these two scenarios.

(5) The experimenter asked the participants about the strengths of the Q-Rapids tool and any suggestions for improvements during debriefing sessions with all participants. In these sessions, the participants individually used cards to record the results, posted them on the wall, and presented them to the rest of the group. Then the experimenter, together with the participants, created clusters of the cards and asked the participants to prioritize strengths and weaknesses by individual voting. The goal was to complement the data from the questionnaire and understand why the participants reported those perceptions.

(6) In parallel to the five sequential steps above, at least one observer documented the progress of each activity using a predefined observation protocol. The observer kept records of the participants' comments and questions on the Q-Rapids tool and of any deviations from the prescribed evaluation procedures. This activity was intended to facilitate later analysis of possible threats to validity, such as experimenter bias. There were different experimenters and observers during each company's evaluation.

The instruments used to support these steps will be described in Section IV.D.

Finally, after the evaluation and once the research team had finalized the analysis, the research team presented the results to the practitioner team, who were the 'champions' of applying Q-Rapids in the four companies. The goal was twofold: to validate the results and to discuss RQ3. With respect to RQ3, we performed the following two additional actions:

- We asked the four UC champions to summarize their challenges and the lessons they learned regarding the use of the Q-Rapids tool in their companies. We instructed them to provide this description in free text in order to avoid any unnecessary constraint on their communication.
- The research team consolidated the responses by categorizing them while respecting the provided text as much as possible. The result was presented to the UC champions, who provided their feedback (in the form of comments to the proposed text and new insights with additional information), yielding the final input for answering RQ3, reported in Section VI.

Population and Sampling: There were three types of target users of the Q-Rapids tool: product owners, managers, and developers. The researchers informed the companies about the target sample and the companies' champions proposed a list of participants based on their suitability and contacted them. Then we drew a convenient sampling including product owners, managers, and developers of the companies involved [54]. At the time of the evaluation, the participants were team members of the pilot project selected in each company for the evaluation of the Q-Rapids tool.

In total, 22 persons from the four companies participated in the evaluation conducted for RQ1 and RQ2 (see Table IV). Among these participants were two product owners, seven project managers, seven managers (including quality and technical managers), three developers, and three participants who did not indicate their role in the demographics part of the questionnaire, but who belonged to one of these categories. All participants had at least eight months of work experience in their companies (Mdn = 10 years, Min = 0.75 years, Max = 32 years) and at least nine months of work experience in their current role (Mdn = 5 years, Min = 0.8 years, Max = 30 years) at the time of this evaluation.

Execution: Between July 2018 and October 2018, we deployed the Q-Rapids tool in each company. This was the second major release of the Q-Rapids tool, whose first prototype had already been deployed in January 2018. Therefore, data collected by the Q-Rapids tool reached back to January 2018. In parallel, the first two authors trained the experimenters and observers responsible for performing the evaluation at each company. Then the experimenters and observers executed this study following the aforementioned procedures between mid-October and November 2018. We scheduled each evaluation session for up to 7 hours including breaks, taking into consideration the availability of the participants.

TABLE IV
DEMOGRAPHIC INFORMATION OF PARTICIPANTS

Participants	UC1	UC2	UC3	UC4
Number	4	10	6	2
Roles	1 Developer, 3 Project Managers	2 Developers, 3 Managers, 2 Project Managers, 3 Others	4 Managers, 2 Project Managers	2 Product Owners
Work experience in the company	13.5 years (1 year - 30 years)	15.3 years (2 years - 32 years)	9.25 years (9 months - 19 years)	6.25 years (2.5 years – 10 years)
Work experience in the current role	10.5 years (1 year - 30 years)	3.6 years (3 months - 15 years)	7.05 years (2 years - 15 years)	3.5 years (2 years - 5 years)

D. DATA COLLECTION INSTRUMENTS

To support the aforementioned evaluation design, the researchers used the following six instruments (available in the Appendix) during the evaluation:

1. Slides with evaluation goals, agenda, and the informed consent and demographics forms.
2. Scripts for a live demo of the Q-Rapids tool assessment and improvement scenarios to give researchers a common artifact to show to the users of the Q-Rapids tool (similar to the examples in Section III). In addition, a document comprising the specification of the QM (e.g., strategic indicators).
3. The task description, consisting of specific scenarios for assessing and improving the software quality of the products in which the participants were involved on a daily basis.
4. A questionnaire, to collect the practitioners' perception regarding the use of the Q-Rapids tool, using the constructs of Table III.
5. Presentation and moderation guidelines for the researchers to conduct a debriefing session about the strengths of the current tool and any suggestions for improvement.
6. The observation protocol, where the observers recorded start time, attendees, end time, activities/events/deviations/disruptions, timestamp, and memos.

E. DATA ANALYSIS

The experimenter and the observer transcribed the participants' answers (regarding the tasks, the questionnaires, and the cards from the debriefing sessions) and the observation protocol into three standardized Excel templates. This served to keep the data analysis consistent among the four companies.

We first carried out within-case analyses of the quantitative and qualitative data collected for each company. Then we compared, correlated, and integrated the results among the companies (cross-case analyses) [55].

Quantitative analysis. We report descriptive statistics including the sample size (N), minimum (Min), maximum (Max), median (Mdn), and modal value (Mode) for the quantitative analyses. If a participant answered more than half of the questions of one construct with "I don't know" (see the questionnaire in the fourth item of Section IV.D), we did not include his/her rating of this construct for our quantitative analysis. This happened on four occasions.

We performed a *One-Sample Wilcoxon Signed Ranks Test*[17] [56], as it is suitable for testing hypotheses with small samples. This served to test whether the participants significantly rated the QM more positively or more negatively, i.e., to check whether or not the answers are significantly lower or greater than a selected middle point in the five-point rating scale data of the questionnaire, i.e., H_0: median (X) = θ where θ was set to 3 (the neutral point). If we were able to reject the null hypothesis (i.e., $p < \alpha$ with $\alpha = 0.05$), we checked for the standardized test statistic (T*) from which we derived whether the result was positive (i.e., Mdn (X)>θ if T*>0) or negative (Mdn (X)<θ if T*<0). Therefore, we also report the significance levels *p* and the standardized test statistics of the *One-Sample Wilcoxon Signed Ranks Test* (see Table V). We used IBM SPSS Statistics 19 (including IBM SPSS Exact Tests for analyzing small samples) and set the confidence level of the test at 95% (i.e., $\alpha = 0.05$).

Qualitative analysis. We used thematic analysis [57] to analyze the participants' feedback on the Q-Rapids tool. At least two researchers derived themes inductively by coding and interpreting all observation protocols, independent of each other. Then these researchers compared their results and resolved any deviations. Moreover, we performed several peer review meetings, which included all experimenters, observers, and analysts, to review the interpretations of the elicited qualitative data. This served to keep the qualitative analyses grounded on the collected evidence and ensured rigor.

V. RESULTS

In this section, we will present our results according to the RQs. The results are combined from the quantitative analysis (see descriptive analytics and hypothesis testing in Table V) and the qualitative analysis (thematic analysis).

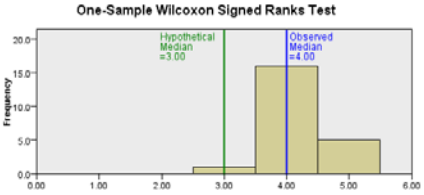
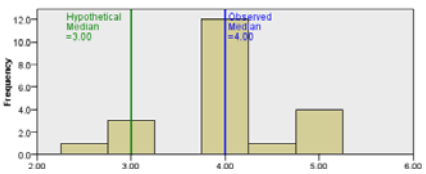
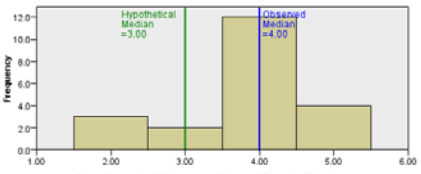
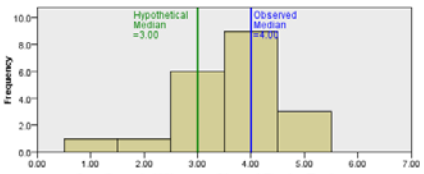
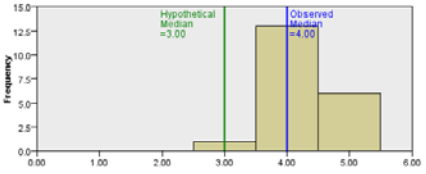
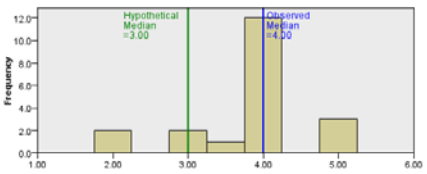
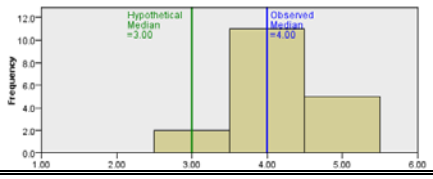
A. RQ1: WHAT IS THE PRACTITIONERS' PERCEPTION REGARDING THE INTEGRATION OF QMs INTO SOFTWARE ANALYTICS TOOLS WHEN ASSESSING SOFTWARE QUALITY IN ASD?

FINDING 1 - THE PARTICIPANTS UNDERSTOOD THE QM WHILE ASSESSING SOFTWARE QUALITY.

[17] https://www.ibm.com/support/knowledgecenter/en/SSLVMB_22.0.0/com.ibm.spss.statistics.algorithms/alg_nonparametric_onesample_wilcoxon.htm

All participants claimed that the strategic indicators, product factors, and process factors provided in the QM were understandable (see Table V: *N=20, Min = 3, Max = 5, Mdn = 4, Mode = 4, p = 0.000,* T* = 4.167). For our hypothesis *H_1: understandability*, the *One-Sample Wilcoxon Signed Ranks Test* revealed that the participants rated the understandability of the QM significantly higher than the neutral point (Mdn(x)= 3). Therefore, the null hypothesis *H_{01}*: *understandability*, i.e., that the participants' perception is neutral towards the *understandability* of a QM within a software analytics tool for assessing software quality, can be rejected.

TABLE V
QUANTITATIVE RESULTS OF THE EVALUATION OF THE QM WITHIN A SOFTWARE ANALYTICS TOOL FOR ASSESSING AND IMPROVING SOFTWARE QUALITY

RQ	Hypotheses	N	Min	Max	Mdn	Mode	One-Sample Wilcoxon Signed Ranks Test	*p*	T*
RQ1.1	H1: Understandability	20	3	5	4	4		.000	4.167
RQ1.2	H2: Reliability	21	2,5	5	4	4		.000	3.816
RQ1.3	H3: Usefulness	21	2	5	4	4		.002	3.038
RQ2.1	H4: Right level of detail	20	1	5	4	4		.023	2.266
RQ2.2	H5: Relevance	20	3	5	4	4		.000	3.943
RQ2.3	H6: Perceived usefulness	20	2	5	4	4		.002	3.160
RQ2.4	H7: Behavioral Intention	18	3	5	4	4		.000	3.666

Each item of the structured questionnaire was rated using a five-point response scale from 1 "strongly disagree" to 5 "strongly agree" and included the option "I don't know".

One-Sample Wilcoxon Signed Rank Test: H_0: Mdn(X) = θ where θ was set to the neutral point (3), with α = 0.05

The participants gave us the following main reasons why a QM within a software analytics tool is understandable (and how understandability can be improved):

- *Clear structure for users.* This was mentioned explicitly by six participants from two use cases; e.g., *"I understand and like it [the QM view] simplified" [UC 3]; "[It has a] clear structure" [UC 2]; "It looks simple!" [UC 3]; "[It] provides [a] 'common language' from analysis (Product Owners) to action (Developers)" [UC 3]*.
- *Appropriate visualizations.* 15 participants from all use cases emphasized appropriate visualization and provided examples such as getting information where users expect it to be; e.g., for the Q-Rapids tool, to have the factor calculation rationale in the detailed strategic indicators view (*"At the moment, the "Detailed strategic indicators" [view] leads directly to the factors. It would be nice to be able to see directly what metrics the strategic indicator is associated with." [UC 2]*).
- *Support for customized settings*, e.g., having explanations and descriptions or support for setting up appropriate thresholds. Another participant associated the support for setting up appropriate thresholds with the understandability of the QM and asked *"What are the proper/default values for thresholds?" [UC 4]*.
- *Including descriptions.* Three participants from two use cases emphasized explanations of elements of QMs. For the Q-Rapids tool, for example, they stated that "*the descriptions of metrics need to be better. The factors that are made of the metrics should also have better descriptions*" *[UC 3]*, e.g., by "*including descriptions of formulas,* e.g., *average of factors" [UC 3]*. These are additional features that would foster understandability.

FINDING 2 - THE PARTICIPANTS FOUND THE QM ELEMENTS RELIABLE FOR ASSESSING SOFTWARE QUALITY.

The computed QM elements were perceived as *reliable* by the participants (see Table V). The *One-Sample Wilcoxon Signed Ranks Test* showed that they considered the reliability of the QM elements as positive, as the null hypothesis H_{02}*: reliability*, i.e., that the participants' perception is neutral towards the *reliability* of a QM within a software analytics tool for assessing software quality, was rejected by this test.

The reasons given by the participants for why they found the QM reliable or how reliability can be further improved can be categorized as follows:

- *Stable computations*. This was explicitly mentioned by one participant: *"Most of the calculation of [the] current model is quite stable now and seems to follow the model" [UC 3]*. Furthermore, two participants from one use case explained that the existing QM calculates values that can be used to understand the quality of the software system (*"When [the QM] is producing "weird" values, I can know something is wrong with the system" [UC 3]*).
- *Computational transparency*, as mentioned by six participants from two use cases: *"In the strategic indicator view, in order to trust data it is important to identify who made the threshold changes if there have been some changes." [UC 3]*. Still, one participant mentioned that transparency should be improved: *"It is aggregated data from other tools; I don't know how exactly it is computed." [UC 4]*.
- *Updated computations,* e.g., a*dditional information* about the last time the data was updated. One participant stated, *"If Jenkins was updated 5 min ago [and this is shown], I know I can trust it." [UC 3]*.
- *Possibility to customize the QM*, e.g., by adjusting the weights of the QM elements relations. This was pointed out by three participants from three use cases and had an influence on their perceptions regarding reliability. One participant suggested improving the tool support for these customizations: *"Specifying weights for strategic indicators seem a bit weak." [UC 2]*.
- *Support for exceptions/false positives*. Another participant explained this with the following example: *"Some files have to be complex in order to provide their functionality. It should be possible to mark them as such, so they don't affect the metrics and show our perception." [UC 4]*.

FINDING 3 - THE PARTICIPANTS FOUND THE QM ELEMENTS USEFUL FOR ASSESSING SOFTWARE QUALITY.

The participants rated the QM elements as *useful* (see Table V). As the *One-Sample Wilcoxon Signed Ranks Test* showed that the null hypothesis with respect to the rating of the participants can be rejected, we can reject our null hypothesis H_{03}*: usefulness*, i.e., that the participants' perception is neutral towards the *usefulness* of the QM elements within a software analytics tool for assessing software quality.

The following categories of reasons were provided by the participants:

- *Gathering and combining data from heterogeneous sources.* This was stated by seven participants from three use cases; e.g., *"Merging information from different sources is useful." [UC 4]*, *"[The QM provides a] general view and one entry for many related tools (SonarQube, Project Manager, etc.). [There is a] combination of many entries" [UC 1]*,

"Good combination to direct tools used" [UC 3], *"Good synthesis from a wide set of data" [UC 1]*.

- *Inclusion of product and process quality*. This is also one of the strengths that impact the usefulness of the QM for assessing software quality (mentioned explicitly by one participant): *"It is good to see that you included both aspects. Because from the product owner perspective, the process perspective is maybe even more interesting. And there are less tools to assess the process. Indeed, there are a lot of tools managing the product quality but there are less tools for the process." [UC 4]*. Furthermore, the possibility to show information about the product or the process separately was mentioned by two participants from two use cases. *"It would be good to have on the highest level a distinction between product quality and process quality." [UC 4]*. For the Q-Rapids tool, *"it is not clear [at the moment] which ones of the factors belong to the category product and which to the process category" [UC 2]*.
- *Different abstraction levels* of information for a variety of stakeholders. This was emphasized by three participants from two use cases, e.g., *"Knowing which files are increasing the complexity is crucial for the developers […] usually the product owner is not that much interested in these details." [UC 4]*.
- *Real-time raw data dashboards*. Three participants from two use cases also perceived the real-time raw data dashboards as useful because a *"better level of information improves reaction time and agility" [UC 1]*. One of these participants further suggested adding raw data dashboards that consider multiple data sources, i.e., enabling raw data aggregation: *"[There is] no raw data aggregation: e.g. separate ElasticSearch documents for GitLab data and SonarQube data" [UC 4]*. These suggestions could help to further increase the usefulness of the QM elements.
- *Appropriate terminology* in visualizations. Furthermore, two participants from two use cases claimed that appropriate visualizations further improve the usefulness of the QM elements. This includes process-specific (i.e., agile) terminology like time tags (e.g., sprints, milestones) in order to ensure compliant use of terminology.

B. RQ2: WHAT IS THE PRACTITIONERS' PERCEPTION REGARDING INTEGRATING QMs INTO SOFTWARE ANALYTICS TOOLS WHEN IMPROVING SOFTWARE QUALITY IN ASD?

FINDING 4 - THE PARTICIPANTS FOUND THE TRACEABILITY BETWEEN ABSTRACT QUALITY ASPECTS AND THE RAW DATA OF THE QM TO BE AT THE RIGHT LEVEL OF DETAIL TO IMPROVE SOFTWARE QUALITY.

The participants assessed the *right level of detail* of the QM needed to drill down from abstract levels of the quality model (e.g., strategic indicators) to the raw data (see Table V). Although three participants from two use cases considered traceability as strongly positive, one participant rated the traceability of the QM as strongly negative. The *One-Sample Wilcoxon Signed Ranks Test* showed that the participants rated traceability significantly positively, although the significance of this finding is the lowest in comparison to the rest of findings ($p = 0.023$). Therefore, our null hypothesis H_{04}*: right level of detail*, i.e., that the practitioners' perception is neutral regarding the *right level of detail* to trace between abstract quality aspects and raw data of a QM within a software analytics tool to improve software quality, can be rejected.

The participants provided the following reasons regarding positive traceability:

- *Drill-down for in-depth knowledge.* Nine participants from two use cases considered it positive that the QM aggregates raw data into abstract quality indicators that are traceable, as *"we need to drill down and be able to trust" [UC 3]*. Meaning that *"the traceability [from strategic indicators to raw data] is very useful." [UC 1]*. One participant further explained: *"We have different levels and you drill down if you miss some information" [UC 3]*. Increased traceability can be achieved if it is possible to *"drill down directly to the metrics from the visualization of the strategic indicators view. [This] would be nice*." [UC 2]. Improving the connection from assessed metrics to raw data dashboards was also mentioned in another use case, e.g., *"from strategic indicators to raw data in the raw data dashboard seems now to work [for a subset] very nice" [UC 3]*.
- *Holistic QM view*. Five persons from two use cases stated the need for a holistic view of the complete QM. They further elaborated it, e.g., *"I need to see the whole hierarchy [as a whole]" [UC 3]; "I would like to have a direct link from the assessed metrics in the QM to the correlated raw data in the raw data dashboard" [UC 4]; "easier access to the data visualized with the raw data dashboard from the implemented QM [is required] for a final product" [UC 4]*
- *Traceability to source tools*. One participant stated that traceability to the source tools (i.e., data producers such as GitLab or SonarQube) is another aspect of the QM's traceability; e.g., *"including the URL in the metrics textual views" [UC 1]*. This traceability would also provide data to allow users to understand the rationale behind the computation.

FINDING 5 - THE PARTICIPANTS TOOK RELEVANT ACTIONS BASED ON THE QM FOR IMPROVING SOFTWARE QUALITY.

The participants assessed the actions that can be taken based on the information provided by the QM as *relevant* (see Table V). The *One-Sample Wilcoxon Signed Ranks Test* supports this finding as the null hypothesis H_{05}*: relevance*, i.e., that the participants' perception is neutral towards taking *relevant actions* to improve software quality based on the information provided by a QM presented in a software analytics tool, can be rejected.

The participants provided the following explanations:

- *The QM supports explicit insights to solve quality problems.* 18 participants from the four companies stated this. Among the reasons, we can find *"it [the QM] allow us to navigate into concrete real metric in order to make right decisions." [UC 1]*. For example, *"quality would improve by fixing the 5 errors" [UC 2]* and the QM supports to *"concentrate to speed up clarifying the real bugs" [UC 3]*. One of these participants further elaborated: *"Knowing which files are increasing complexity is crucial for the developers to know; this can be done by the raw data dashboard because it is great at pinpointing the issues in the code. Usually [the] product owner is not that much interested in these details but for taking actions, the developers need to know where the problem is." [UC 4]*.
- *Providing precise information to be able to act on it.* Participants indicated the importance of actionable information, and also indicated the case in which it was not precise enough to be able to act on it; e.g., *"It is possible that [the] raw data dashboard provides enough information in some cases and for some metrics. But for example, there is no data for finding the root cause for test success related information." [UC 2]*. In addition, *"this raw data dashboard will not provide me possible solutions. For example, you can reduce the bug density by two methods. Either you don't include these identified bugs in the current or next planned Sprint [i.e. postpone it] or the second way how to handle bug density is to hire more […] developers to increase the resources." [UC 4]*.

FINDING 6 - THE PARTICIPANTS TOOK PERCEIVED USEFUL ACTIONS BASED ON THE QM FOR IMPROVING SOFTWARE QUALITY.

The participants perceived the actions that can be taken based on the information provided by the QM as *useful* for improving software quality (see Table V). This finding is supported by the rejected null hypothesis of the *One-Sample Wilcoxon Signed Ranks Test*, which showed that the participants gave positive ratings. Therefore, our null hypothesis H_{06}*: perceived usefulness*, i.e., that the participants' perception is neutral towards taking *perceived useful actions* for improving software quality based on the information provided by a QM presented in a software analytics tool, can be rejected.

The reasons given by the participants can be categorized as follows:

- *Useful for improving quality deficiencies.* As the participants perceived that the QM provides explicit insights to solve quality problems, they explained why the actions taken based upon the QM could support them in practice; e.g., *"I can use [the QM] to identify problems" [UC 4]; "better information related to the status of the project allows us to take more accurate decisions during meeting plan" [UC 1]*. The information provided by the QM enables *"better detection of quality issues which arrives to the customer" [UC 1]*. Moreover, one participant was concerned that *"the quality of the commits need to be reviewed because of high trend of error identification." [UC 3]*. Another person claimed: *"I don't have needed information how to use this in [practice] to improve quality but I guess this could be useful and can be used to improve quality" [UC 3]*, for example by *"[putting] more people to work on issues" [UC 3]*.
- *Early detection of quality issues.* Two participants from two use cases claimed that the real-time information improves reaction time and agility as *"the more information about the environment [is there,] the more possible adjustments for the project [are feasible]" [UC 4]*. Furthermore, one participant explained that *"early detection of issues [implies issues that are consequently] more simple to fix" [UC 1]*.
- *Combination of different data providers in a single tool.* Furthermore, the combination of many data providers in a single tool is useful for their work. One participant summarized this with: *"Giving developers additional tools increases their work time. It takes more effort and time, more overload that they don't have. Having everything in one single tool is easier to convince them." [UC 4]*. However, convincing practitioners to use a tool is still challenging; e.g., *"But we had also challenges to convince them to use even SonarQube." [UC 4]*.

FINDING 7 - THE PARTICIPANTS INTEND TO USE THE QM TO IMPROVE SOFTWARE QUALITY.

With our final question, we asked the participants whether they would use the QM within a software analytics tool to improve software quality or not; i.e., whether they would prefer manual/ad-hoc assessment and improvement without tool support. The participants had the intention to further use the implemented QM (see Table V). For our hypothesis H_{07}*: behavioral intention*, again the *One-Sample*

Wilcoxon Signed Ranks Test revealed that their null hypothesis H_{07}: *behavioral intention*, i.e., that the participants' perception is neutral towards *intending to use* a QM within a software analytics tool to improve software quality rather than using a human approach, can be rejected.

The participants gave us the following main reasons:

- *Raw data dashboard*. One of the participants explicitly highlighted this twice: *"I would include it [the raw data visualization] in my daily work." [UC 4]; "In the current state I would use Kibana [the raw data dashboard]." [UC4]*.
- *QM assesses quality at every lifecycle point*. This was mentioned by nine participants from four use cases because the Q-Rapids tool *"would allow to evaluate the impact of new features and the quality of the project at every lifecycle point" [UC 1]*. One of these participants continued: *"It is based on concrete metrics which allow to measure and evaluate [...] the quality of our product during the development life cycle; [and] has a significant progress in the software development industry." [UC 1]*.
- *Customization of the QM*. Three participants from one use case emphasized the possibility to customize the QM, e.g., to add new strategic indicators, as a strength of using the QM. *"[There is the] possibility to add new strategic indicators" [UC 3]* and *"the feature of removing metrics from the graph" [UC 3]*. *"We love it!" [UC 3]*. This allows practitioners to customize the QM according to their needs, which is an explicit prerequisite of using the QM for one participant, who stated that *"if the QM is improved to provide me more detail, I'd probably use it." [UC 4]*. In general, additional customizations would increase the behavioral intention, as stated by three participants from three use cases. For instance, they suggested for the Q-Rapids tool to specify a *"different mechanism to assess weights for the quality factors while assessing the strategic indicators" [UC 2]* because one of them additionally asked: *"Is there not a way to define weights for the strategic indicators' factors?" [UC 1]*.

VI. DISCUSSION: CHALLENGES AND LESSONS LEARNED

This section presents the discussion about RQ3: *What are the challenges and lessons learned that practitioners face when integrating QMs into software analytics tools for assessing and improving software quality in ASD?*

As a final step, based on their own experience, the practitioner team of this study reflected on the challenges and lessons learned when using the Q-Rapids tool to assess and improve software quality. As mentioned in Section IV, they provided written feedback, which was processed by means of a thematic analysis [57]. The resulting topics are presented below, with a distinction made between challenges and lessons learned.

A. CHALLENGES REGARDING THE USE OF THE Q-RAPIDS TOOL

Challenge #1. Need for tailoring to the company. A software analytics tool like the Q-Rapids tool requires a comprehensive understanding of the company's software development or lifecycle process to be able to extract a solid QM as required by the tool. Although the tool can be used from the beginning with common strategic indicators, it requires extra effort to completely adapt its QM to the needs of each company's business.

Challenge #2. Need for a shared language. One aspect experienced by companies to date is the need to use a language already known and understood by the potential users. Here, the QM (names for metrics, factors, strategic indicators) has to be clear and based on the practical language known and used by product owners and developers. Of course, this will improve with time, although it also requires a customization process for each company (the language and terms might not be standardized across all software houses in Europe).

Challenge #3. Need to be informative for the user. A challenge with tools such as the Q-Rapids tool is how to interpret the information that is rendered. From an end-user's perspective, they need to understand whether a certain value of an indicator/factor is good or bad, and how it relates to the previous stage in the project or to other projects (a benchmark needs to be established and known to the users, and the 'delta' needs to be shown clearly).

Challenge #4. Need for integration with other tools. Currently, in many software development companies and software houses, teams are already using some common tools (such as Jira, GitLab, and SonarQube) to manage the development process and the quality of their code and products. Such tools are now embedded into developers' way of working, and into the processes used in companies. Therefore, when introducing the Q-Rapids tool, it has to be clear when it is going to be used to assess and improve software quality, e.g., before checking the status of the project in GitLab or Jira or afterwards. It needs to be clarified when exactly in the agile (or any other) process this new tool should be used.

Challenge #5. Need for transparency and more clarity on actions. While automatic data gathering from data sources can be seen as an advantage, some end-users (i.e., product owners) may have a need to understand from which raw data the factors and indicators are computed and how. For instance, for one of the companies, this was the case for the *product quality* indicator: What data is aggregated to arrive at a single value? One of the crucial aspects for future adoption is the clarity regarding possible decisions and courses of actions. The Q-Rapids tool should

be able to propose a catalog of suggested decisions (as in a typical decision support system), but the user should also be aware (either by training, through practice, or supported by a clear manual) of which decisions can be made and why. One of the possible obstacles to successful adoption would be offering solutions without any clear indication and information of how the tool can be used, for what decisions, etc. – the potential user would be lost and discouraged from further usage. Results should be connected to the process (e.g., allowing users to visualize the assessment for the last sprint, including the next milestone, as an informative field to provide some context for better understand the results).

Challenge #6. Simplify tool installation. A general comment was that the deployment of the Q-Rapids tool in the companies was complex and cumbersome. However, a closer look at this statement shows two sides of the coin. On the one hand, although the tool included a wide variety of connectors for the most popular data sources in the software industry, companies needed additional ones, which was not trivial in terms of knowledge required. One company reported that choosing the appropriate factors, defining the metrics, and finding the best data sources for this purpose turned out to be extremely challenging due to several factors (e.g., access to relevant databases). This raised the issue of which competencies are needed in the process, such us knowledge on data protection regulations (e.g., EU GDPR). On the other hand, adding new strategic indicators appears to be much easier. To sum up, software analytics tools should offer easy deployment options such as dockers, and facilitate as much as possible the setting up of the tool (e.g., user interface for the configuration of the different elements to decrease the technological knowledge required).

Challenge #7. Need for an efficient tool configuration process. Some situations make the adoption of the Q-Rapids tool (and other software analytics tools) challenging from a technological perspective. The first of these is the configuration of the connectors. One constraint is that connector configuration parameters (such as user name, password and end-point) need to be set up per project, even though the data sources are the same across projects (e.g., data sources may have multiple deployments). In other words, for each data source in each case, the connectors for mining the data as well as for the feedback loop back to the backlog must be customized. To increase the feasibility of any software analytics tool, it is necessary to provide configurable connectors via a file or user interface, as this decreases the need for specific coding skills and decreases the overall adoption time. Second, some of the techniques may require additional tooling to address the subsequent challenges regarding technological integration and licenses (e.g., tools for Bayesian networks provide limited facilities for such scaling, so any business-relevant purpose would require purchasing a license for the tool).

B. LESSONS LEARNED FROM THE USE OF THE Q-RAPIDS TOOL

Lesson learned #1. Embrace an incremental adoption approach. It usually takes some time for any new solution to be fully embedded into a company's software development style, and any new process is usually based on some *play&try* in practice sessions. We recommend adopting the Q-Rapids tool in a company for a smaller-scale product or solution being developed in order to enable extension of the solution to a wider number of products later on. This will allow adjusting the QM as well as potentially tailoring the data connectors used.

Lesson learned #2. Boost transparency as a business value. Incremental adoption also helps to get an overview and enable fruitful discussions among developers and related stakeholders of the product regarding valuable indicators. For instance, one of the companies reported that, after the experiences with two solutions using the Q-Rapids tool, the user teams clearly recognized the added value provided by the good visibility of the company-defined strategic indicators and quality factors through real-time aggregated and transparent data.

Lesson learned #3. Use strategic indicators as an asset for monitoring progress. Strategic indicators are useful not only for knowing current values but also for understanding the progress of a project. One of the companies stated explicitly that Q-Rapids end-users need to know how a particular indicator's value relates to the previous stage in the project or to other projects (a benchmark needs to be established and known to users, and the 'delta' needs to be shown clearly).

Lesson learned #4. Have a single access point to software-quality-related data. With the Q-Rapids tool, the project manager no longer needs to know which tools provide which metrics and to connect to them in order to understand the various measurement processes and to link and synchronize results from different tools. The various indicators give the project manager a very easy and fast way to analyze and understand a potentially complex situation. This aspect is completely built into the Q-Rapids QM and its associated dashboards where the various aspects collected by the different software development lifecycle metrics are analyzed.

Lesson learned #5. Use QMs integrated into software analytics tools as a communication space. The Q-Rapids tool provides effective means for different product stakeholders to discuss and interact on the basis of a common dashboard. The various metrics and results are given on a factual basis and facilitate the interaction between the various team members and the project manager in the decision-making process. Having different levels of abstraction and views (e.g., *Strategic Dashboard*, *Raw Data Dashboards*) opens usability up to a wider range of users (different roles are interested in different levels of abstraction).

Lesson learned #6. Enable tailoring to a product or project. Even within the scope of a single company, the QM might not be exactly the same. In a normal case, many of the metrics will persist; however, deviations over the products being developed or ongoing projects may occur. These variations need to be identified and incorporated into the QM (and then into the tool) as some kind of parameter. In this direction, one of the companies reported that some Q-Rapids end-users would like the opportunity to play with the model itself (plug or unplug some input data) and to assign weights used in the aggregation process. After their first experience in a pilot project, another company decided to use their expert knowledge to identify interesting strategic indicators that are not provided by other tools, or which would at least be presented from a different viewpoint by the Q-Rapids tool. In another company, the person in charge of R&D decided that it was worth measuring the general management quality of two projects and independently added a new indicator to measure that. In a matter of seconds, the customization was achieved on the company's common Q-Rapids website, and this new indicator was then created for all monitored projects. Therefore, we learned that support for customizing the QMs in software analytics tools is key to assessing and improving software quality, which is one of the characteristics of the Q-Rapids tool.

Lesson learned #7. Involving experts. Putting a product like the Q-Rapids tool into action requires knowledge from software engineers, especially technology experts and data scientists. The companies pointed out some particular situations. First, to set up the connectors, a lot of expertise is needed: end-users of the tool who can identify the relevant data content from the UI; software engineers who can assess the quality of raw data with respect to the functionality of the connector; and data scientists for verifying the relation between UI fields and the actual structure of the database. Second, some of the techniques used by the analytics tool, such the option of Bayesian networks for creating strategic indicators [58], require specific competence. Last, implementation also requires knowledge of particular technologies that a company might not have, e.g. Kafka. To sum up, the role of a specialist setting up the Q-Rapids tool environment will be crucial for industry adoption. If no specialist is available, it may be necessary to hire someone or outsource the development in order to implement the solution.

VII. THREATS TO VALIDITY

As with any empirical study, there might be limitations to our research method and findings. This section discusses possible threats to validity in terms of construct, conclusion, internal, and external validity [47] and emphasizes the mitigation actions applied.

Construct validity refers to whether *the employed measures appropriately reflect the constructs they represent.* We used seven validated, reliable instruments from the literature to correctly operationalize our evaluation goals and the identified quality aspects. Clearly defining the constructs reduced the risk of mono-operation bias. We also included open questions and comment fields to further collect the participants' opinion. The open feedback sessions enabled us to further elaborate the practical relevance of the strengths and drawbacks of the Q-Rapids tool with all the participants. The use of quantitative and qualitative measures and observations reduced mono-method bias. Furthermore, we aimed at creating a safe environment, encouraging the participants to highlight any negative aspects and make suggestions for the improvement of the Q-Rapids tool. Finally, some of our results (particularly the challenges) could be caused by a not optimal Q-Rapids tool implementation rather than by the integration of QMs into software analytics tools. Still, these results are useful for others to learn how to build such an infrastructure in realistic settings.

Conclusion validity refers to whether *correct conclusions are drawn from (correct) statistical analysis.* To ensure the reliability of this evaluation, the measurement plan and procedure, including the creation of instruments for the implementation and execution, were documented in detail. For instance, the instruments were reviewed by the complete researcher team and by one member of the practitioner team, and during the analysis we involved researchers who had not been involved in the creation of the Q-Rapids tool. In this way, we mitigated risks such as using poorly designed instruments or fishing for results during the analysis, which would have led to a subjective analysis. Another point deserving attention was the training we provided to the experimenters and observers to enable all of them to apply the treatment in a similar way. For instance, the participants in the different companies received the same training and Q-Rapids tool version. During the task sessions, the participants performed concrete tasks aimed at making them use the main features of the Q-Rapids tool. In order to compare the results among the companies and due to time constraints, these tasks had to be the same for all cases. Thus, some might not have been optimal for all the different projects involved. Furthermore, this version of the Q-Rapids tool included real project data from the four use cases. However, not all available data could be used on the day of the evaluation, and at least one of the selected projects had finished before. Therefore, the data had not changed during the previous weeks. This might have had an influence on the answers of the participants regarding the practical experience they gained with the tool for their work. Thus, the results can only be interpreted as an indication of appropriateness. Furthermore, we were aware that we would only get a small sample size (i.e., 22 participants) and looked for appropriate statistical tests.

Internal validity refers to whether *observed relationships are due to cause-effect relationships, and whether it is possible to explain unequivocally the changes in the dependent variable due to the influence of the independent variables.*

We evaluated the Q-Rapids tool by drawing a convenient sample of product owners, managers, and developers working in several application domains and software development environments. One limitation of our work is that we were not able to get a random sample of participants in the pilot projects of the companies.

In addition, we defined an evaluation protocol in advance, which included a specific description of our planned procedure and the order of using the materials, i.e., a script of the demonstrations to the participants, the tasks, the questionnaire, and an explanation with all the steps that had to be performed by the experimenter and the observer. We distributed all the materials to the researchers as well as to the use case representatives (who did not participate) to collect feedback in order to further improve our design. After all the partners had agreed on the final version of the evaluation guidelines, we executed the evaluation accordingly. This should mitigate the fact that we needed to split the work of conducting the evaluation among different researchers and partners. Some of the eight researchers who conducted the evaluation were involved in developing the Q-Rapids tool modules. To minimize that bias, we made sure that in each case there were at least two researchers present, one acting as the moderator/experimenter and one as the observer, to emphasize that the participants could speak freely. After inspecting and analyzing all the collected data, including the observation protocol, we could not identify any influence by these researchers on the participants' perceptions.

External validity refers to whether *findings of the study can be generalized.* Our results are tied to the context of the companies involved in the Q-Rapids project. Our goal was to better understand practitioners' perception. We characterized the environment as realistically as possible and studied the suitability of our sampling (see Section IV). This case study could be the input for further meta-analysis aimed at generalizing the results (e.g., [59]).

VIII. CONCLUSIONS

Agile software development companies need tools to continuously assess and improve software quality. For this study, we conducted a case study across four European ASD companies to explore practitioners' perceptions on the integration of QMs into software analytics tools. We aimed at exploring key aspects needed for the widespread adoption of these tools in industry, namely understandability, reliability, usefulness, right level of detail, relevance, perceived usefulness, and behavioral intention.

Twenty-two practitioners involved in the study agreed that they understood the integration of QMs into software analytics tools and found the QM elements reliable and useful for assessing software quality. In addition, the practitioners found the QMs to be at the right level of detail and the actions taken from the QMs to be relevant and perceived useful, and they intend to use QMs within software analytics tools. These findings are complemented with reasons for how these qualities were achieved in the four companies and for how they can be improved further.

Taking into account that the introduction of a new tool and its measurable impact usually takes several months to years, the results are very promising. For future adoption, we reported seven challenges and seven lessons learned from four practitioners leading the application of QMs within software analytics tools in their ASD companies. One hope we associate with our work is that it will not only contribute directly to the existing body of knowledge on the use of QMs within software analytics tools in industry – which currently is still weak – but that it will also encourage other practitioners to join us in exploring this promising solution for the problem of software quality assessment and improvement in industry.

Future work will consist of further supporting practitioners in meeting software quality challenges:

- We observed that the definition of QMs varies among companies. In addition to offering tool support for QMs as plug-ins in software analytics tools, ongoing work is aimed at generating QMs not only from expert knowledge but using more cost-effective data mining and analysis algorithms (e.g., identifying QM elements interrelations), and at creating a QM repository.
- Building a complete software analytics infrastructure in a company (e.g., all data available from a single point) requires time and should be done incrementally. Based on the current infrastructure in the four companies of this study, other software quality scenarios are being explored, such as prediction and simulation of quality, and semi-automatic generation of quality requirements.
- We believe that software analytics can provide greater value in ASD companies by mining usage data to understand the behavior of users, which is another direction for future work.
- Companies need clarity about the process for deploying and applying software analytics tools in their environments. A next step is a process describing software analytics tools deployment and customization for new companies.

APPENDIX

The appendix contains the instruments used during the evaluation: https://figshare.com/s/217851eed7ec1da7ab86

ACKNOWLEDGMENT

We thank all members of Bittium, ITTI, Nokia, and Softeam who participated in the evaluation of the Q-Rapids software analytics tool. We also thank Axel Wickenkamp for implementing the Q-Rapids quality model, Liliana Guzmán for her support for previous evaluation activities, and Sonnhild Namingha for proofreading this article.

REFERENCES

[1] M. Ardolino, M. Rapaccini, N. Saccani, P. Gaiardelli, G. Crespi, and C. Ruggeri, "The role of digital technologies for the service transformation of industrial companies," *Int. J. Prod. Res.*, vol. 56, no. 6, pp. 2116–2132, Mar. 2018.

[2] S. F. Wiesbaden, "AUTOSAR — The Worldwide Automotive Standard for E/E Systems," *ATZextra Worldw.*, vol. 18, no. 9, pp. 5–12, Oct. 2013.

[3] J. Bosch, "Speed, Data, and Ecosystems: The Future of Software Engineering," *IEEE Softw.*, vol. 33, no. 1, pp. 82–88, Jan. 2016.

[4] "Magic Quadrant for Business Intelligence and Analytics Platforms." [Online]. Available: https://www.gartner.com/doc/3611117/magic-quadrant-business-intelligence-analytics. [Accessed: 14-Apr-2019].

[5] C. Bussler and BIRTE (Conference), *Business intelligence for the real-time enterprises : first international workshop, BIRTE 2006, Seoul, Korea, September 11, 2006 : revised selected papers*. Springer, 2007.

[6] R. P. L. Buse and T. Zimmermann, "Information needs for software development analytics," in *2012 34th International Conference on Software Engineering (ICSE)*, 2012, pp. 987–996.

[7] 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, Sep. 2013.

[8] H. Gall, T. Menzies, L. Williams, and T. Zimmermann, "Software Development Analytics (Dagstuhl Seminar 14261)," *DROPS-IDN/4763*, vol. 4, no. 6, 2014.

[9] T. Menzies, "The Unreasonable Effectiveness of Software Analytics," *IEEE Softw.*, vol. 35, no. 2, pp. 96–98, Mar. 2018.

[10] J. Czerwonka, N. Nagappan, W. Schulte, and B. Murphy, "CODEMINE: Building a Software Development Data Analytics Platform at Microsoft," *IEEE Softw.*, vol. 30, no. 4, pp. 64–71, Jul. 2013.

[11] C. Sadowski, J. van Gogh, C. Jaspan, E. Soderberg, and C. Winter, "Tricorder: Building a Program Analysis Ecosystem," in *2015 IEEE/ACM 37th IEEE International Conference on Software Engineering*, 2015, pp. 598–608.

[12] C. Condo and B. Seguin, "The Forrester New Wave™: Value Stream Management Tools, Q3 2018," 2018. [Online]. Available: https://www.forrester.com/report/The+Forrester+New+Wave+Value+Stream+Management+Tools+Q3+2018/-/E-RES141538. [Accessed: 11-Mar-2019].

[13] V. Basili *et al.*, *Aligning Organizations Through Measurement*. Cham: Springer International Publishing, 2014.

[14] International Organization For Standardization Iso, "ISO/IEC 25010:2011," *Software Process: Improvement and Practice*, 2011. [Online]. Available: http://www.iso.org/iso/iso_catalogue/catalogue_tc/catalogue_detail.htm?csnumber=35733.

[15] M. Kläs, J. Heidrich, J. Münch, and A. Trendowicz, "CQML scheme: A classification scheme for comprehensive quality model landscapes," *Conf. Proc. EUROMICRO*, pp. 243–250, 2009.

[16] S. Wagner *et al.*, "Operationalised product quality models and assessment: The Quamoco approach," *Inf. Softw. Technol.*, vol. 62, pp. 101–123, Jun. 2015.

[17] L. Guzmán, M. Oriol, P. Rodríguez, X. Franch, A. Jedlitschka, and M. Oivo, "How Can Quality Awareness Support Rapid Software Development? – A Research Preview," Springer, Cham, 2017, pp. 167–173.

[18] "ISO 8402:1994 - Quality management and quality assurance -- Vocabulary." [Online]. Available: https://www.iso.org/standard/20115.html. [Accessed: 14-Apr-2019].

[19] B. W. Boehm, J. R. Brown, H. Kaspar, M. Lipow, G. Macleod, and M. Merrit, "Characteristics of Software Quality.," 1978.

[20] V. Basili and H. D. Rombach, "TAME: Integrating Measurement into Software Environments," 1987.

[21] C. Ebert and R. Dumke, *Software measurement : establish, extract, evaluate, execute*. Springer, 2007.

[22] N. Fenton, J. Bieman, and J. Bieman, *Software Metrics*. CRC Press, 2014.

[23] M. Kläs, J. Heidrich, J. Münch, and A. Trendowicz, "CQML Scheme: A Classification Scheme for Comprehensive Quality Model Landscapes," in *2009 35th Euromicro Conference on Software Engineering and Advanced Applications*, 2009, pp. 243–250.

[24] C. Izurieta, I. Griffith, and C. Huvaere, "An Industry Perspective to Comparing the SQALE and Quamoco Software Quality Models," in *2017 ACM/IEEE International Symposium on Empirical Software Engineering and Measurement (ESEM)*, 2017, pp. 287–296.

[25] J.-L. Letouzey and T. Coq, "The SQALE Analysis Model: An Analysis Model Compliant with the Representation Condition for Assessing the Quality of Software Source Code," in *2010 Second International Conference on Advances in System Testing and Validation Lifecycle*, 2010, pp. 43–48.

[26] J. P. Carvallo, X. Franch, and C. Quer, "Managing Non-Technical Requirements in COTS Components Selection," in *14th IEEE International Requirements Engineering Conference (RE'06)*, 2006, pp. 323–326.

[27] E. Kocaguneli, T. Menzies, and J. W. Keung, "On the Value of Ensemble Effort Estimation," *IEEE Trans. Softw. Eng.*, vol. 38, no. 6, pp. 1403–1416, Nov. 2012.

[28] R. Wettel, M. Lanza, and R. Robbes, "Software systems as cities," in *Proceeding of the 33rd international conference on Software engineering - ICSE '11*, 2011, p. 551.

[29] W. Maalej, Z. Kurtanović, H. Nabil, and C. Stanik, "On the automatic classification of app reviews," *Requir. Eng.*, vol. 21, no. 3, pp. 311–331, Sep. 2016.

[30] D. Ståhl and J. Bosch, "Modeling continuous integration practice differences in industry software development," *J. Syst. Softw.*, vol. 87, pp. 48–59, Jan. 2014.

[31] B. Fitzgerald and K.-J. Stol, "Continuous software engineering: A roadmap and agenda," *J. Syst. Softw.*, vol. 123, pp. 176–189, Jan. 2017.

[32] A. Vetro, R. Dürre, M. Conoscenti, D. M. Fernández, and M. Jørgensen, "Combining Data Analytics with Team Feedback to Improve the Estimation Process in Agile Software Development," *oundations Comput. Decis. Sci.*, vol. 43, no. 4, pp. 305–334, 2018.

[33] M. Kuruba, P. Shenava, and J. James, "Real-time DevOps Analytics in Practice," in *QuASoQ*, 2018, p. 40.

[34] B. Snyder and B. Curtis, "Using Analytics to Guide Improvement during an Agile–DevOps Transformation," *IEEE Softw.*, vol. 35, no. 1, pp. 78–83, Jan. 2018.

[35] M. Kim, T. Zimmermann, R. DeLine, and A. Begel, "Data Scientists in Software Teams: State of the Art and Challenges," *IEEE Trans. Softw. Eng.*, vol. 44, no. 11, pp. 1024–1038, Nov. 2018.

[36] D. Cukier and Daniel, "DevOps patterns to scale web applications using cloud services," in *Proceedings of the 2013 companion publication for conference on Systems, programming, & applications: software for humanity - SPLASH '13*, 2013, pp. 143–152.

[37] D. G. Feitelson, E. Frachtenberg, and K. L. Beck, "Development and Deployment at Facebook," *IEEE Internet Comput.*, vol. 17, no. 4, pp. 8–17, Jul. 2013.

[38] P. Rodríguez *et al.*, "Continuous deployment of software intensive products and services: A systematic mapping study," *J. Syst. Softw.*, vol. 123, pp. 263–291, Jan. 2017.

[39] S. Neely and S. Stolt, "Continuous Delivery? Easy! Just Change Everything (Well, Maybe It Is Not That Easy)," in *2013 Agile Conference*, 2013, pp. 121–128.

[40] E. L. Vargas, J. Hejderup, M. Kechagia, M. Bruntink, and G. Gousios, "Enabling Real-Time Feedback in Software Engineering," in *ICSE*, 2018.
[41] S. Dueñas, V. Cosentino, G. Robles, and J. M. Gonzalez-Barahona, "Perceval," in *Proceedings of the 40th International Conference on Software Engineering Companion Proceeedings - ICSE '18*, 2018, pp. 1–4.
[42] M. van den Brand, S. Roubtsov, and A. Serebrenik, "SQuAVisiT: A Flexible Tool for Visual Software Analytics," in *2009 13th European Conference on Software Maintenance and Reengineering*, 2009, pp. 331–332.
[43] H. Huijgens, D. Spadini, D. Stevens, N. Visser, and A. van Deursen, "Software analytics in continuous delivery," in *Proceedings of the 12th ACM/IEEE International Symposium on Empirical Software Engineering and Measurement - ESEM '18*, 2018, pp. 1–10.
[44] S. Martínez-Fernández, A. Jedlitschka, L. Guzman, and A. M. Vollmer, "A Quality Model for Actionable Analytics in Rapid Software Development," in *2018 44th Euromicro Conference on Software Engineering and Advanced Applications (SEAA)*, 2018, pp. 370–377.
[45] P. Ram, P. Rodriguez, and M. Oivo, "Software Process Measurement and Related Challenges in Agile Software Development: A Multiple Case Study," Springer, Cham, 2018, pp. 272–287.
[46] N. Marz and J. Warren, *Big data: principles and best practices of scalable real-time data systems*. Manning Publications, 2013.
[47] P. Runeson and M. Höst, "Guidelines for conducting and reporting case study research in software engineering," *Empir. Softw. Eng.*, vol. 14, no. 2, pp. 131–164, Apr. 2009.
[48] L. López *et al.*, "Q-Rapids Tool Prototype: Supporting Decision-Makers in Managing Quality in Rapid Software Development," Springer, Cham, 2018, pp. 200–208.
[49] V. Venkatesh and H. Bala, "Technology Acceptance Model 3 and a Research Agenda on Interventions," *Decis. Sci.*, vol. 39, no. 2, pp. 273–315, May 2008.
[50] V. McKinney, K. Yoon, and F. "Mariam" Zahedi, "The Measurement of Web-Customer Satisfaction: An Expectation and Disconfirmation Approach," *Inf. Syst. Res.*, vol. 13, no. 3, pp. 296–315, Sep. 2002.
[51] D. L. Goodhue and R. L. Thompson, "Task-Technology Fit and Individual Performance," *MIS Q.*, vol. 19, no. 2, p. 213, Jun. 1995.
[52] Y. W. Lee and D. M. Strong, "Knowing-Why About Data Processes and Data Quality," *J. Manag. Inf. Syst.*, vol. 20, no. 3, pp. 13–39, Dec. 2003.
[53] M. Tavakol and R. Dennick, "Making sense of Cronbach's alpha," *Int. J. Med. Educ.*, vol. 2, pp. 53–55, Jun. 2011.
[54] J. Daniel, *Sampling Essentials: Practical Guidelines for Making Sampling Choices*. 2455 Teller Road, Thousand Oaks California 91320 United States : SAGE Publications, Inc., 2012.
[55] M. B. Miles, A. M. Huberman, and J. Saldaña, *Qualitative data analysis : a methods sourcebook*. .
[56] F. Wilcoxon, "Individual Comparisons by Ranking Methods," *Biometrics Bull.*, vol. 1, no. 6, p. 80, Dec. 1945.
[57] V. Braun and V. Clarke, "Using thematic analysis in psychology," *Qual. Res. Psychol.*, vol. 3, no. 2, pp. 77–101, Jan. 2006.
[58] 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," in *Proceedings of the 14th International Conference on Predictive Models and Data Analytics in Software Engineering - PROMISE'18*, 2018, pp. 52–55.
[59] S. Martínez-Fernández, P. S. Medeiros Dos Santos, C. P. Ayala, X. Franch, and G. H. Travassos, "Aggregating Empirical Evidence about the Benefits and Drawbacks of Software Reference Architectures," in *International Symposium on Empirical Software Engineering and Measurement*, 2015, vol. 2015–Novem, pp. 154–163.