Quality Practitioners' Differing Perspectives on Future Software Quality Assurance Practices

https://doi.org/10.3991/ijim.v15i24.20123

Altti Lagstedt[1,2], Amir Dirin[2,3(✉)], Päivi Williams[2]
[1]University of Turku, Turku, Finland
[2]Haaga-Helia University of Applied Sciences, Helsinki, Finland
[3]University of Helsinki, Helsinki, Finland
amir.dirin@haga-helia.fi

Abstract—Constant changes in business context and software development make it important to understand how software quality assurance (SQA) should respond. Examining SQA from supplier and client perspectives, this study explores how different groups of SQA practitioners perceive future needs. A survey (n = 93) conducted in fall 2017 explored the views of SQA organizations on future trends. The results indicate that SQA organizations differ slightly in their attitudes to quality categories, as do different groups of SQA practitioners. It is argued that these differences should be taken into account when developing and implementing future SQA strategy. It is further argued that the found basic enables SQA management, evaluation of new practices and allocation of resources to ensure that all quality categories remain balanced in the future.

Keywords—software quality assurance, SQA management and evaluation

1 Introduction

Software quality assurance (SQA) is changing [1, 2]—not because quality itself has changed but because the world has changed irreversibly. In particular, clients' business environments, business needs and software development methods are in constant flux. The scale and complexity of software have increased, and the Internet has become an important development environment. New development methods have emerged, solving old problems and posing new ones; at the same time, business models are changing rapidly as digitization becomes all-pervading. With the availability of new digital services in all areas of business, end users of these services are becoming increasingly selective about the services they elect to use [1, 3–5].

The digitization of services entails more direct interaction with the user, and customer engagement has become increasingly important [1, 3]. Information systems (IS) and software (SW) are now of strategic importance even in business areas not traditionally considered IT-oriented. In these circumstances, the quality of SW is crucial, and social media makes this transparent, as the global audience is immediately alerted

to any failure [2, 3]. Software quality assurance (SQA) is therefore crucial in meeting business goals such as reducing time-to-market and increasing security, performance and customer satisfaction [1, 5]. In addition, as software development (SWD) methods change constantly, quality assurance practices must also change, and traditional plan-driven ('waterfall') development has largely been replaced by change-driven methods like agile development [4]. However, change-driven methods also pose certain challenges, such as growing technology debt [6].

Traditional plan-driven SWD had serious limitations in relation to lack of business feedback and volatile business requirements, yielding solutions that were technically correct but useless for business purposes [7]. Change-driven (agile) methods present the opposite problem, as solutions that are business-relevant may have significant internal defects (e.g. fragmented architectural structure) [6, 8]. This issue can be described as a technical debt, defined here (following [9]) as an immature code that 'may work fine and be completely acceptable to the customer, but excess quantities will make a program unmasterable, leading to extreme specialization of programmers and finally an inflexible product' [9]. Technical debt is an undesirable side effect of agile development; if unnoticed, it is likely to grow during the development period [6, 9–11], and there is no single tool or method to prevent this [12, 13]. Repaying technical debt, which is an essential element of quality assurance, is likely to be more challenging for change-driven development than for the traditional plan-driven approach [10]. Additionally, change-driven development entails risk; if the product owner (client) has no clear vision, and priorities change constantly, or if there is no shared understanding of what is to be delivered, the scope of development becomes unclear, and quality assurance becomes challenging [8, 14]. All SW development methods present their own challenges [8], and it is important to identify quality assurance practices that align with prevailing methods in order to guard against loss of quality. SQA practices that attempt to address these challenges include agile and DevOps, with increasing automation of quality assurance and testing practices [1, 5]. In discussing these practices, it is important to note two broad criteria of software quality: suitability for desired use and faultlessness [15, 16]. As the former relates to client business needs and the latter to developer practices and methods, both of these perspectives should be taken into account when evaluating SQA practices.

As new technologies and methodologies evolve, successful organizations must engage with both current trends and future opportunities [1]. In this constantly changing environment, organizations employ a range of strategies and frameworks to manage change and to prevent chaos. Widely used SQA frameworks such as software quality standard ISO 25020 are grounded in software life cycle processes and plan-driven development. In moving from plan-driven to change-driven software development, it seemed important to assess whether the basic principles of exiting SQA frameworks can be used for classifying, discussing and developing future practices and trends [4], as the existing presumptions may no longer be the only valid ones. In particular, it seemed important to understand whether actual trends and practices align with the known software quality categories, and whether different groups of SQA practitioners share a common understanding of software quality categories.

Although the changes referred to above are almost overwhelming, there are few academic studies of future SQA practices. Many blogs and other non-academic publications have addressed such issues, but these are not usually empirically grounded. The purpose of the present research is to bridge this gap in the academic literature. We know that frameworks, such as ISO 25020, are constantly developed and evolving, but in this study, we concentrate to the basic principles which are not to be changed so often.

2 Software quality assurance

Among definitions of SQA, [17] still seems valid: 'a systematic, planned set of actions necessary to provide adequate confidence that the software development process or the maintenance process of a software system product conforms to established functional technical requirements as well as with the managerial requirements of keeping the schedule and operating within the budgetary confines'. By implication, SQA is not a final development step but an ongoing assessment mechanism that continues during subsequent maintenance [17]. [17] identified six components of SQA: pre-project quality, project life cycle quality, infrastructure error prevention and improvement, software quality management, standardization, certification and SQA assessment, and organizing for SQA (the human components). Contrary to this extensive approach, quality assurance is often seen in practice as unnecessary and unproductive because it does not add value directly in terms of code and features [18]. However, the extent of SQA in developers' daily work is an important determinant of success and is therefore worth evaluating.

The importance of good software quality metrics is well known (see for example [19]). SQA commonly employs a range of metrics as a subset of software development metrics. Not all of these relate directly to quality, but they may exert a considerable indirect influence—for example, in motivating and guiding developers [20]. Equally, some software metrics (e.g. velocity in change-driven development) may impact negatively on quality [14, 20], making it important to define such metrics clearly.

According to [15], two key factors determine software quality: 1) number of errors or bugs (i.e. faultlessness) and 2) suitability for the intended purpose—in other words, does it supply the features identified by users in the requirements specification? While this approach is also applied in evaluating project success (see for example [21]), it remains quite general but gives little indication of how software quality is to be achieved and measured in practice. In addition, when software quality is discussed, it is important to keep in mind that supplier and client have different business objectives; for the supplier, the project is itself the business while the client benefits only after the software is implemented and in use [22, 23]. This is one reason for clients' and suppliers' differing perspectives on software quality measurement; while clients are more interested in suitability, suppliers emphasise faultlessness. In this regard, software quality has both subjective and objective aspects; while software error rate can be objectively assessed through comprehensive testing, usability depends on business need, which is an ambiguous and volatile concept [19]. When entering into a contract, objective criteria are favoured, as these are tangible and impartially measurable [24].

2.1 Software quality classification

As the purpose of this study was to utilize a widely used SQA framework in evaluating and classifying the opinions of different SQA groups. For that purpose we selected globally published ISO 25020 as a baseline and extracted some basic principles of it to be considered. ISO 25010 provides a detailed definition of software quality in terms of 1) domain-specific functional properties (i.e. suitability for the desired use) and 2) quality properties that determine how well the software performs (i.e. faultlessness). Quality properties include functional suitability, reliability, performance efficiency, operability, security, compatibility, maintainability and portability [16].

Software quality is assigned to three categories: 1) quality in use, 2) external quality and 3) internal quality [25]. Internal quality is prerequisite for external quality, and appropriate external quality is prerequisite for quality in use [16]. In ISO 25020, quality in use refers to client business needs and to software usability, which can only be measured in a real-world operational environment when software is already in use [25]. This aligns with the view that project performance metrics (schedule, budget and deliverables) are insufficient to capture project value [21, 26–28]. As client satisfaction is an essential element of project success, this can only be measured after the development process is complete and the software has been in use for some time [21, 29]. During that time, changes in business needs or in the external environment may hamper evaluation, and the subjective nature of quality in use adds to this difficulty [19]. Nevertheless, quality in use is an essential element of software quality, as technically perfect software is of little value unless usable in the given business context.

External quality is a measure of how well the software performs in a test situation and how well it addresses the stated business requirements. This can be examined and measured during the testing and operational stages of the product life cycle [25].

Of the three, the lowest level is internal quality, which relates to the software's inner structure. Internal quality can be measured in small parts at different phases of development–for example, the quality of intermediate deliverables. Because internal quality relates to software architecture, structure and components, which can be explicitly measured, it is clearly objective in nature, and some internal quality measurements can be automated. However, as internal quality does not guarantee user satisfaction, other measures of quality are also needed [25].

As part of quality metrics, testing is the process of analysing an item of software to detect any differences between existing and required conditions. Testing is also used to evaluate software features [18]. For both developer and client, testing is generally the most visible part of quality assurance, traditionally at the end of the development project. However, this kind of testing is often unpredictable in terms of duration and effort. While some SQA models incorporate testing throughout the software development process (see for example [30]), SQA practices and frameworks are often linked to plan-driven development, each phase of which includes a corresponding SQA phase (see for example [30]). As this does not readily accommodate change-driven development, time-boxed agile projects demand a different approach to quality assurance [18].

In addition to the three categories of quality referred to above (internal, external and quality in use), ISO 25010 acknowledges that software quality is affected by the quality of development processes [16]. Standards and frameworks that assess the

quality (and maturity) of these processes include ISO 12207, ISO 15288 and CMMI. These frameworks assume that if development processes are sufficiently capable and mature, the software will be of uniform quality. However, these 'development maturity' frameworks are suited mainly to plan-driven development and pay less attention to change-driven development, which characterises the current mainstream [4]. For that reason, the present study focuses more on the known characteristics of known software development and quality classification than on software process maturity models. We argue here that the ISO 25020 classification is useful for studying SQA in rapidly changing development and business contexts because it is possible to situate changes and draw comparisons with earlier practices. While there are other approaches, ISO 25020 is a natural choice for present purposes, as it is an international standard widely known and used.

2.2 Software development methods

As early as 1989, [19] noted the importance of evaluating how new methods and tools affect software quality and argued that SQA professionals should be acquainted with the various software development methods in order to be able to prepare a viable quality plan for the method in question [17, 19]. Despite the emergence of multiple software development methods over the last six decades [31], no single method will suit all cases [32, 33]; any SWD method may struggle or fail (see for example [34]), and in the same way, SQA must be context-driven [2].

Earlier research classified SWD methods in myriad ways [35]–for example, on the basis of heaviness [36], flexibility [37] or objectives and orientation [38, 39]. In our view, these classifications are problematic for their overlaps and conceptual incoherence. For example, heavyweight and change-driven SWD methods are sometimes seen as opposites, but also heavyweight change-driven SD projects have been conducted [40]. For that reason, we decided to adopt a classification based on the control concept, distinguishing between plan-driven (waterfall) and change-driven (agile) SWD methods [14].

The central idea of plan-driven methods is that software can be fully planned before coding begins, and development can be divided into distinct phases [7]. Quality assurance usually plays a significant role at two points in this process; at the outset, it is important to ensure that the listed requirements correspond to real business needs, and at the end, it is important to thoroughly test the new software. This traditional plan-driven approach has used in many different frameworks (e.g. CMMI, SPICE) to assess and improve the quality of development processes. However, the plan-driven approach is known for its serious quality problems, and development projects commonly fail in both of the above respects. First, the requirements may fail to meet real business needs, either because they are poorly defined or because the business needs or environment have changed during the development period [24, 41]. Second, a fixed delivery date or budget may leave insufficient time or resources for comprehensive testing [7, 42]. As a result, errors may only become apparent after the software is deployed, leading to serious production problems. In general, the plan-driven approach is seen as inflexible and uncommunicative, with little client contact between requirements specification and acceptance testing (or implementation).

In plan-driven development, quality assurance often employs a V-model, involving a corresponding test practice for each development phase. The requirements specification is validated by an acceptance test; system specification and design are validated by a system integration test; and detailed design and coding are validated by a sub-system integration test and by module and unit tests [43, 44]. There are more detailed SQA models (see for example [30]), but the V-model and other plan-driven SQA models highlight the challenges of this form of development. Both development and quality assurance processes are inflexible, and while faultlessness of individual units can be tested almost immediately, system requirements (i.e. suitability for the desired use) can be validated only after building the whole system.

Change-driven methods seek to overcome the known limitations of plan-driven development by being iterative and incremental, so accommodating changing business requirements. By using client feedback from the previous step to determine objectives for the next, software can be shaped incrementally to address client needs. In addressing the known problems of plan-driven development in relation to changing business requirements and lack of communication between developer and client, change-driven methods are considered more customer-oriented. However, this approach also presents some challenges from a quality perspective, and SQA practices must be adjusted in change-driven environments [2]. Examining different QA approaches in agile development contexts, [18] found that agile methods focus more on low-level rather than release-level testing. This emphasises daily development and associated unit testing, with a test after each iteration (e.g. a sprint in Scrum). However, change-driven methods do not build project release-level tests, which are easily ignored, and it is difficult, for instance, to apply a V-model [18]. Although highly customer-oriented, change-driven methods neglect final level testing related to customer business needs [18]. Because it relies heavily on continuous communication between developer and client, change-driven development tends to neglect business suitability validation, which is left by default to the client. In addition, early and continuous software delivery with a rapid release cycle presents challenges for testing because the cycle imposes fixed deadlines and does not allow for extension of the testing period if defects exceed the estimated level. Additionally, there are obvious challenges for quality assurance when requirements change during later phases [18]. Change-driven development emphasises face-to-face communication, and business people and developers must work well together. For that reason, the documentation on which traditional testing is based may not necessarily exist. Instead, the most detailed and up-to-date information about the expected results often resides in the heads of developers and clients, presenting challenges for quality assurance and final validation [18].

Despite addressing low-level coding-related issues, including unit tests, change-driven methods are likely to cause significant problems in relation to the structure and architecture of the developed code. Because change-driven development makes it impossible to predefine the architecture and code structure, new increments may introduce new and unpredictable code requirements, leading to software conflicts. Cumulative inner problems of this kind are referred to as technical debt. Technical debt reduces the quality of the application and generates risks for further development, but also weakens the morale of developers, which in turn lowers the quality of development [45]. As early as 1992, [9] noted that technical debt is not an issue for change-driven development alone; poorly executed software development always incurs this debt,

which must be paid back to avoid accumulation and organizational stasis [9]. Nevertheless, change-driven development is seen to be especially prone to technical debt, making the issue more topical [6, 10, 11, 46]. According to. [10], this is because change-driven development emphasises working software rather than documentation; along with a lack of attention to software architecture, this means that testing and quality assurance practices are neglected in pursuit of fast deployment [10]. Additionally, each iteration introduces new features that the client considers important. As these are impossible to anticipate at the outset, the new features may conflict with the existing code and architecture, and even systematic testing of units and modules cannot guarantee that they will work together. In general, maximising agility increases the risk of trade-offs between fast deployment and poor development practices [10, 14].

Granted these difficulties, it can be argued that change-driven development practices related, for instance, to coding and refactoring may impact positively on management of technical debt [46, 47]. Additionally, change-driven development makes extensive use of the so-called DevOps approach (combining development and support operations), affording new opportunities for quality assurance [2, 3, 48]. DevOps is one way of compensating for the missing life-cycle dimension of change-driven methods. Because DevOps draws no real distinction between development and maintenance phases, which are harmonised in continuous deployment [1, 48], quality assurance must also be continuous. New deployments must be undetectable for the client, and techniques like task automation are largely supported as part of DevOps (although practitioners seem to differ in their interpretation of the term) [3, 48].

3 Research questions, methodology, design and implementation

3.1 Research questions

The research objectives of the present study were to assess 1) the extent to which the expectations of SQA suppliers and clients differ; 2) the extent to which the expectations of different SQA practitioners groups differ; and 3) is the future of SQA comprehensive and balanced, or do some areas receive more attention than others? To address these objectives, we formulated the following research question.

Do SQA groups differ in how they view the future of SQA?

This overarching question yielded the following sub-questions.

RQ1.1: How do predictions of future practice vary among SQA practitioner groups?
RQ1.2: Based on the study findings, what are the main requirements for SQA practitioners in the future?

3.2 Research methodology

The present study adopts a mixed-methods approach, which [49] defined as collecting, analysing, and interpreting qualitative and quantitative data simultaneously.

As the purpose of the present study was to acquire detailed information about the views of Comiq's clients and employees, it was appropriate to combine quantitative and qualitative approaches to data collection and analysis. A quantitative research strategy entails the collection of numerical data to investigate the relationship between theory and research, using a deductive, natural science approach based on an objectivist conception of social reality [50]. The use of qualitative data analysis at different points can enrich the findings by revealing unexpected points of view and adding meaning and depth to the interpretation of quantitative data [51].

To gather both forms of data, a questionnaire was considered appropriate, addressing both the target population's expectations and views in relation to quality assurance and clients' expectations in relation to SQA practitioners. Predictions of future SQA practices found in the relevant literature and reports were used as a baseline for the questionnaire, along with guidelines used by the case company. We also referred to non-academic guidelines [52, 53] that seem influential among SQA suppliers.

3.3 Research design

The selected SQA supplier organization was Comiq Ltd., which specialises in test management, test automation, technical testing, DevOps and requirements specification for large and demanding business-critical software development projects. Headquartered in Helsinki, Comiq currently employs around 70 SQA and DevOps experts and works with many of Finland's largest banks, retailers, insurers, telecoms and manufacturing companies.

All of Comiq's clients and practitioners have their own quality assurance methods and strategies. The aim of the study was to determine how supplier and clients differ in their perspectives on SQA. While Comiq has extensive knowledge of SQA practices and trends, their clients look at how these can be utilised for their own purposes. As well as comparing these two perspectives on trends and expectations, responses were also classified by practitioner group. The presumption was that professionals in different roles might be expected to emphasise different SQA actions, which should be taken into account in project design and implementation. The study did not include end users (others than in-house developers) and other client representatives because of their limited relevant knowledge, especially in cases where there is no in-house development [3].

We anticipated that SQA practitioners could be classified according to their work and responsibilities. Organizational decision-making generally happens at three levels—operational, tactical and strategic [14]—and SQA practitioners can be found at each of these levels. For the purposes of this study, we assigned the participating practitioners to six groups. 1. Testers work at the operational decision level; generally involved in software testing rather than development, their main task is to test what developers produce. 2. SQA professionals work mainly at the tactical level but may also be found at operational level. They have a broader perspective than testers, planning, supporting and consulting on software quality assurance, and are normally responsible for a specific area of SQA. 3. SQA managers have more administrative responsibilities. Working at the tactical level as project managers or equivalent, they are responsible

for overall software quality assurance. 4. SW developers work at the operational level and are mainly responsible for software development, with additional responsibility for SQA. 5. Other IT managers work at tactical or strategic level and are mainly responsible for the client organization's development projects, programmes or infrastructure, including SW development project managers and IM staff. 6. Business developers such as product or programme owners work at strategic level and are responsible for developing IS-related business processes. The boundaries between these decision-making levels are not always distinct [14], and with the advent of change-driven methods, development team responsibilities have extended to tactical and even strategic levels. In general, however, the above roles are clear and unambiguous.

The questionnaire was largely based on the World Quality Report [52] and the DevOps Maturity Matrix [53]. Although non-scientific, the World Quality Report provides a good overall picture of current trends in SQA. According to Comiq, these guidelines (especially the former) have a major influence on its business and on SQA business in general. To supplement and finalise the questionnaire, the existing SQA literature was also reviewed, and the personal views and experiences of one researcher with several years' experience of SQA and DevOps were taken into account. Furthermore, we extended the questionnaire with open-ended question. The questions were finalised in collaboration with Comiq management, who signed off on the published version.

3.4 Sample

A link to the online questionnaire was distributed via email to 350 people working in quality assurance-related tasks such as testing, as well as quality assurance practitioners or consultants, test managers and test leaders. All of the study participants had worked for Comiq or for their client companies.

Based on a Comiq customer register and with the assistance of Comiq management, the link to the questionnaire was sent to all relevant Comiq personnel (n = 60) and to a select group of Comiq clients (n = 290). The link was sent during the period 22.6.2017–30.6.2017, and the questionnaire remained available until 7.7.2017. During that time, 93 individuals completed the questionnaire.

3.5 Research implementation

To anonymise data collection, a browser-based survey tool (kyselynetti.com) facilitated participation without registration. The tool provided options for a range of question types, phrasing one's own questions, adding explanatory text and uploading images—in other words, it was possible to design the survey systematically, item by item. The tool also allowed the respondent to preview single questions and the entire questionnaire. As the target group consisted of software-oriented individuals, the online tool was considered a natural choice.

There were 15 questions in total: three background questions, four open questions, four multiple choice questions, and four 5-point Likert scale questions (1 = slightly important, 2 = moderately important, 3 = important, 4 = very important, and 0 = can't

say/not important). The four Likert scale questions included 22 items in total. It is important to mention that this study does not utilize data from all 15 questions but refers only to those data of immediate relevance (from three background questions, four open questions and three Likert scaled questions). The questions utilized in this study are presented in Table 1.

Table 1. The questions utilized in the study

No.	Question	Type
1.	What is the work description of your current work	Background
2.	Do you currently work for a company that specializes in testing and quality assurance?	Background
3.	How many years have you worked on testing and quality assurance?	Background
4.	How important do you see the following (SQA) features in the future?	Likert, 8 SQA features
5.	What (other) features do you expect from a quality assurance expert in the future?	Open
6.	How important do you see the following things in the future?	Likert, 8 SQA practices
7.	What do you think are the most important things for software development quality assurance in the future?	Open
8.	How important do you see the automation of the following types of testing in the future?	Likert, 5 test types
14.	What do you think are the biggest obstacles to the realization of the (SQA) target state?	Open
15.	What do you think is the most important action to improve software quality assurance?	Open

In the first phase of the analysis, we focused on background-related questions and the three selected Likert scale questions, which were transformed and calculated as emphasis points. We then analysed the four open questions. The three Likert scale questions related to future expectations about the attributes of QA experts, the future software development process and the likely importance of test automation. This was done by importing the data to Microsoft Excel worksheets and using Excel graphics and other features to create figures. The open questions were analysed by coding the answers. After formulating preliminary data coding categories from the literature as recommended by [54], we added new codes developed inductively during data analysis. The data from the open questions were used to supplement and enrich the answers to the Likert scale questions.

4 Results

This paper includes only those questionnaire items that helped to answer the research question. The first question we asked related to current position and job status. The data were collected from those with experience in quality assurance-related fields.

To classify respondents (Figure 1), we used the categories referred to in section 3.1. Most of the answers were provided by those in the first three groups (Testers, SQA Professionals, SQA Managers) (see Figure 1). When targeting clients, it was assumed that most of the respondents would have close links to software quality assurance.

We also asked about participants' experience of testing and quality assurance (summarised in Figure 2). Most (80%) of the respondents had been working in the industry for 5 years or more, which meant they had long experience of SQA and could be assumed to know about more than just the latest fad.

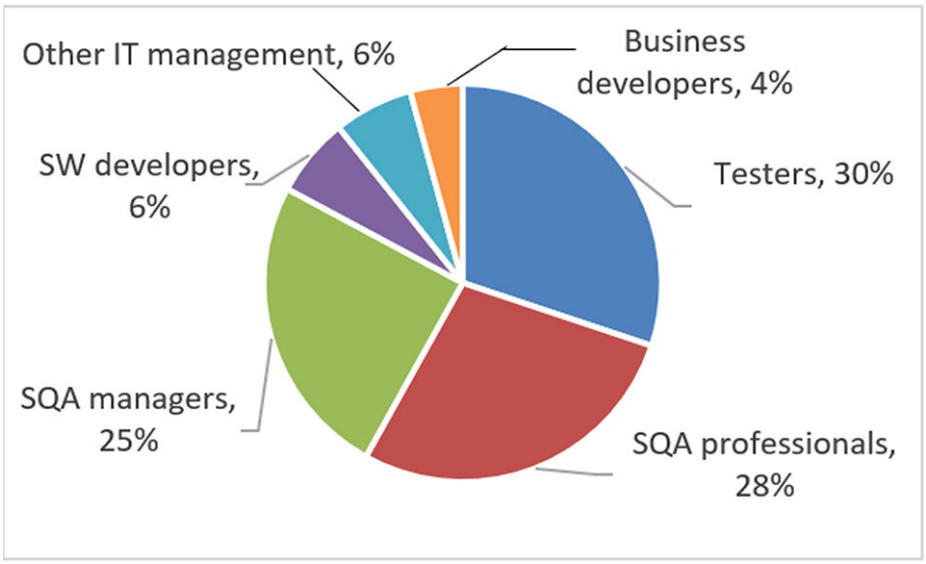

Fig. 1. Breakdown of practitioner groups

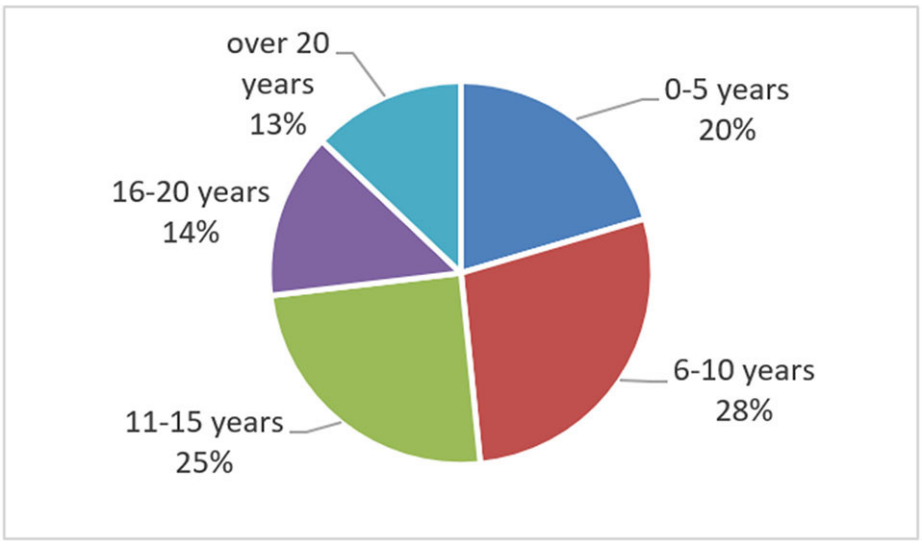

Fig. 2. Duration of participants' work experience of quality assurance-related tasks

The next three sets of questions related to the future of SQA. The first set of questions (i = 1) asked about the perceived future importance of SQA practitioner attributes (understanding the business, understanding the customer's needs, technical skills, test automation, quick defect correction process, reporting of testing progress, communication skills). The second set of questions (i = 2) asked about the perceived future importance of various SQA practices (functional requirements management, test environments are virtualized, version control supports testing, testing is automated at different levels, automated quality assurance is linked to requirements, new version can be released completely automatically, server environments are mainly virtualized, quality assurance can be monitored throughout the whole organization). The third set (i = 3) asked about the perceived future importance of various SQA-related automation practices (unit testing, integration testing, system testing, regression testing, acceptance testing). Question sets and individual evaluations of each practice are discussed in more detail in [55]; here, we focus on the question sets and comments collected from open questions.

The analysis established that all three evaluation question sets were considered equally important (even though they covered different numbers of practices). To ensure balance, preliminary emphasis points were first calculated for each set before summing to obtain emphasis points (p) for each case (i.e. a specific quality category and a specific SQA practitioner group; see (1) and Table 1). The emphasis point p is therefore a comparison value, indicating the emphasis placed on each quality category by the various practitioner groups. Emphasis points were calculated as

$$p = \sum_{i=1}^{3} \frac{a_i * 100}{n_i * x_i} \tag{1}$$

where a_i indicates how many respondents rated practices in that quality category as *important* or *very important*; n_i indicates the number of respondents in that practitioners group or organization; and x_i indicates the number of practices included in the quality category in question; *i* indicates the evaluation question set; and the constant value 100 is used only to facilitate comparison.

Emphasis points were calculated for both SQA client and supplier organizations, and for all SQA practitioner groups mentioned above. Emphasis points were calculated separately for each (ISO 25020 based) quality category in order to identify any differences of emphasis in the different groups. Table 1 identifies emphasis points (p) for SQA clients and SQA supplier (Comiq) in each quality category.

Table 2. Client and supplier emphasis points in quality categories

Quality Category	SQA Clients (*p*)	SQA Supplier (*p*)
Quality in use	214	235
External quality	265	260
Internal quality	237	252
Number of respondents (*n*)	56	37

Note: Scale: *p* = 0–300.

As Table 1 shows, the category quality in use, which relates to finished software quality assurance, received 214 emphasis points from SQA clients and 235 emphasis points from the SQA supplier (Comiq)—the lowest number for both organizations. External quality, which is often a prerequisite of quality in use, received 265 emphasis points from SQA clients and 260 emphasis points from the SQA supplier. Internal quality, which in turn is often a prerequisite for external quality, received 237 emphasis points from SQA clients and 252 emphasis points from the SQA supplier. Formula (1) was again applied to calculate emphasis points for the six SQA practitioner groups (Table 2).

Table 3. SQA practitioner group emphasis points in quality categories

Quality Category	Testers	SQA Professionals	SQA Managers	SW Developers	Other IT Management	Business Developers
Quality in use	215	241	208	227	233	216
External quality	264	264	264	271	242	255
Internal quality	243	237	244	269	216	258
Number of respondents (*n*)	28	26	23	6	6	4

Note: Scale: $p = 0–300$.

Table 2 shows the highest number of emphasis points for quality in use in the SQA Professionals group (p = 241) and the lowest number in the SQA Managers group (p = 208). Testers and Business Developers returned remarkably low emphasis point totals in this category (215 and 216, respectively), and SW Developers and Other IT Management returned lower emphasis points for quality in use than for other categories. In the external quality category, the SW Developers group returned the highest number of emphasis points (p = 271) while the lowest number was in the Other IT Management group (p = 242). The differences are small in this quality category, with strong emphasis across all SQA practitioner groups. In the internal quality category, the SW Developers group returned the highest number of emphasis points (p = 269) while the lowest number was returned by the Other IT Management group (p = 216).

Table 2 also confirms that the majority of participating SQA practitioners worked as quality assurance experts—for example, Testers, SQA Professionals, and SQA Management are the most strongly represented groups in almost all quality categories. In general, testers (n = 28) constitute the largest group.

5 Discussion

The quantitative study reported here sought to identify trends and expectations among software quality assurance practitioners on both supplier and client sides, as well as in different SQA practitioner groups. In a survey of Comiq personnel and clients

(n = 93), SQA practitioners' views of future practices were collected and analysed, and the perspectives identified were clustered in relation to the (ISO 25020 based) quality classification. The study findings answer the research questions as indicated below.

RQ1.1: How do predictions of future practice vary among SQA practitioner groups?

We looked at two broad groups: SQA organizations (client and supplier) and SQA practitioner groups (Testers, SQA Professionals, SQA Managers, Software Developers, Other IT Management and Business Developers). In relation to the quality classification, the results indicate some differences between the SQA supplier and their clients, as well as among the six SQA practitioner groups. Both SQA supplier and client emphasise external quality, as do all the SQA practitioner groups, indicating overall consensus in this regard. One predicted external quality trend in future SQA relates to test automation. Internal quality was also highly valued by SQA suppliers and quite highly by clients. To our surprise, quality in use was less highly valued by client representatives than by SQA suppliers. One possible explanation relates to the study sample; only SQA practitioners were included, and the results might have been different if the questionnaire had also addressed the end user perspective. It would be interesting to investigate whether a different approach to sampling would affect the results, or whether clients place less emphasis on quality in use because they consider it self-evident, or whether there are other reasons for these differences. This interesting finding should be followed up in future studies. However, based on these results, we can say that client-side SQA practitioners should take more serious account of quality in use (and associated SQA practices).

We found greater variation among practitioner groups than between suppliers and clients. Notably, only the Other IT Management group clearly valued internal software quality less than other categories. External quality was assigned fairly equal emphasis across all SQA practitioners groups. In contrast, the greatest fluctuation among groups related to quality in use, which was least appreciated by all groups other than SQA Professionals and Other IT Management. Quality in use also divided opinion among SQA practitioner groups; while Testers, SQA managers, SW developers and Business Developers placed little emphasis on quality in use, SQA Professionals and Other IT Management valued quality in use more than internal quality. It was unsurprising that Testers and SW developers placed less value on quality in use, as they are much closer in their daily work to the software code and its internal and external quality than to actual software use. It seems obvious that they concentrate on practices and trends related to their work; as one tester put it,

It is good for a quality assurance specialist to be willing to learn the technical background of the system because the system must work consistently. By understanding the technical logic of the system, it is possible to understand what should be tested.

A more surprising finding was that SQA Managers and Business Developers placed little value on quality in use. One possible explanation relates to budget; in the open questions, several respondents pointed out that SQA is not sufficiently appreciated, and that savings seem more important than comprehensive testing.

There are too few resources for testing, the same people have too many tasks in the different areas, testing is still often a side task with other work.

In practice, missing resources are primarily sought from cheaper countries. This leads to ever-increasing training circles. The testing professional is a testing professional, and companies should seek them rather than savings.

The diminished emphasis on quality in use aligns with earlier findings from the literature. Quality in use is subjective, which makes it difficult to measure neutrally [19]. It has to be measured in a realistic and operational system environment [25], and reliable measurements are possible only after development is complete [21, 29]. The attitudes of SQA managers and business developers should clearly be further investigated in future studies, especially on the business side of client organizations. In general, these findings emphasise the importance of comprehensive appreciation of all SQA categories: if some practitioner groups show less appreciation of certain categories, it will be difficult to develop balanced SQA practices—especially if some practices or categories are favoured and some are disregarded for external reasons such as budgeting. This issue clearly warrants discussion in supplier and client organizations.

We turn now to RQ1.3: Based on the study findings, what are the main requirements for SQA practitioners in the future? In this regard, there is clearly a need for extensive knowledge of different development methods and practices. The open questions highlight the impact of change in the business environment and in SW development methods, especially in relation to mobile application development. Mobile development is seen as more dynamic than traditional SW development, with an emphasis on faster development cycles and agile development practices. In the open questions, respondents noted that SQA practices—especially those suitable for agile development—must align with SW development methods:

Application development and testing closer together in agile development, a tester [should have a] role that primarily gives feedback on the application's quality and helps developers.

To ensure that alignment, SQA practitioners should have a good grasp of all the main SW development methods and, in particular, of the risks and pitfalls associated with each. It was also noted that developers need to take more responsibility for SQA; tight alignment between development method and SQA practice is considered especially important in agile development, where continuous deployment requires continuous SQA. Incorporating SQA into each development step makes it easier to cope with technical debt, which seems central to agile development quality. Test automation was commonly mentioned as the most important agile-related SQA practice.

Less paper and more automation.

Increasing automation; regression tests should be executed as automatically as possible, especially if the release interval is short.

Test automation makes it possible to accelerate both test cycles and development cycles and can also help to minimise technical debt; automating routine testing allows SQA practitioners more time to evaluate code structure.

Automating all possible repetitive work to free time from performing [mechanical] tests for actual intelligent testing.

Although widely supported, test automation also attracted some criticism. In particular, it was considered expensive and not suitable in all cases; at worst, it was seen to slow the pace of development because developing test cases was considered slower than manual testing.

There is a lot of hype about test automation, but in the early stages of development—when, for example, a user interface changes with every iteration—automating user interface testing can consume time unnecessarily.

In the same time as it takes to create test automation cases, it is possible to manually test a much wider range of system functions.

It was also considered important that business development should align with SW development—in other words, when using agile methods, business development cannot be based on a waterfall approach. Understanding the business context was seen as one important element of SQA.

In my opinion, a quality assurance expert needs very comprehensive knowledge—an ability to understand large complex technical systems, as well as business processes.

One notable problem was the lack of client-side knowledge of SQA and, in some cases, of SWD methods.

On the client side, there is no testing or release control over projects/systems. They have no testing expertise of their own but buy it per project, leading to varied methods and tools, with no continuity.

Returning, then, to our overall research question (Do SQA groups differ in how they view the future of SQA?), we found that SQA supplier and client organizations place differing emphases on quality categories when discussing future SQA trends and practices, as do the six SQA practitioner groups. Although relatively minor, these differences should be taken into account when developing and managing future SQA practices in both supplier and client organisations.

Our findings indicate that all three quality categories (internal quality, external quality and quality in use) are considered important for the future despite differences of emphasis. We also note the conviction that future SQA practices should balance quality categories—that is, all three categories should be taken into account when planning SQA practices. These findings are important for software development organizations in building or developing their SQA practices, as well as for SQA supplier organizations

like Comiq as well; their offering can be based on these quality categories and should be balanced. This is especially important in cases that emphasise specific internal or external quality assurance practices such as test automation.

Understanding different SWD methods was also considered important; although we did not ask specifically about this, it was mentioned several times. As SWD methods apparently determine the viability of SQA practices, practitioners need to understand the main differences and challenges. One example of how SWD method affects those challenges is technical debt, which is often associated with agile development. While techniques that prevent technical debt (such as automatic regression tests) were mentioned several times, it is important to remember that no technique can automatically eliminate it [12, 13]; good awareness is also needed of technical debt and its causes [56].

6 Conclusion

Changes in the business world and in software development entail changes in SQA practices. To understand future SQA needs, these changes must be understood from different perspectives. The present study examined the assumptions underlying SQA supplier and client perspectives, as well as the differing perspectives of SQA practitioners. Based on commonly known (ISO 25020 based) quality categories, the biggest differences between organizations and practitioner groups relate to quality in use. Although quality in use is subjective, and as such, difficult to evaluate reliably, it has remarkable importance in utilization of the software. Because of that, we recommend that quality in use SQA practices to be emphasized, especially among business developers, who are responsible for producing added business value with new software. We also see it important that in future studies business developers' attitudes to be studied more in detail.

Aligning SQA practices with SW development methods was seen important, and it is essential to understand not only different SQA practices, but also the features of different SW development methods. Especially in agile development there is need for continuous SQA practises, and the importance of test automation was emphasized. In addition to technical solutions like test automation, we also found that SQA practitioners and software developers may in future be expected to demonstrate broader overall knowledge, not only of software development and SQA practices, but also of business development and their alignment. The used quality categories proved to work well for this study, and we claim it remain a suitable framework for future SQA management as well.

For practitioners, it is important to take account of the different categories of quality (internal, external and quality in use) when developing SQA practices, and of the differing perspectives of practitioner groups. As SWD methods change constantly, SQA practitioners should also be acquainted with different SWD methods and associated challenges. For academics, there is a pressing need to address the lack of research on future SQA trends as identified here in light of the evident importance of this issue. While practitioner groups were found to differ in their attitude to SQA, these differences were relatively minor, and further research is recommended to validate these

findings and to extend their generalisability. It also seems important that future studies should investigate other perspectives and approaches to develop a more comprehensive picture of future SQA needs.

6.1 Validity and reliability

The present study is based on data collected by means of an online questionnaire, which was made available for two-and-a-half weeks. As the link was sent only to a specific group of people, it is unlikely that anyone outside this group answered the questionnaire, and it is reasonable to assume that the data came from valid sources. Additionally, most of the participants left their contact details (email), indicating that only those who were invited actually participated. Although the participants were from several companies, all were related to one supplier, which is a potential source of bias. Against that, the supplier has an extensive clientele from diverse business areas.

The overall study accumulated a much larger volume of data covering the wider area of software engineering. However, this paper utilized only those data of immediate relevance to future trends in SQA. As we were especially interested in differences among SQA practitioners based on the quality categories, analysis based on emphasis points was considered appropriate here. We would argue that this approach also enables practitioners to apply these results to future SQA strategy setups.

6.2 Limitations

The results of this study are valid only for the specific research context; a survey of other suppliers and their clients would probably yield different results.

7 References

[1] Fuggetta, A., Di Nitto, E.: 'Software Process', in 'Proceedings of the on Future of Software Engineering—FOSE 2014' (2014), pp. 1–12. https://doi.org/10.1145/2593882.2593883

[2] Gopalakrishnan, R., Rajasekaran, N., Eswaramoorthi, G.: 'A Study of Emerging Trends in Software Testing and Quality Assurance' *Int. J. Eng. Manag. Res.*, 2016, **6**, (1), pp. 376–380.

[3] Borg, M., Olsson, T., Franke, U., Assar, S.: 'Digitalization of Swedish Government Agencies—A Perspective Through the Lens of a Software Development Census', in 'International Conference on Software Engineering' (2018). https://doi.org/10.1145/3183428.3183434

[4] Theocharis, G., Kuhrmann, M., Münch, J., Diebold, P.: 'Is water-scrum-fall reality? On the use of agile and traditional development practices', in 'International Conference on Product-Focused Software Process Improvement' (Springer, 2015), pp. 149–166. https://doi.org/10.1007/978-3-319-26844-6_11

[5] Capgemini, Micro Focus, Sogeti: 'World Quality Report 2017–18' (2018).

[6] Rodríguez, P., Haghighatkhah, A., Lwakatare, L.E., *et al.*: 'Continuous deployment of software intensive products and services: A systematic mapping study' *J. Syst. Softw.*, 2017, **123**, pp. 263–291. https://doi.org/10.1016/j.jss.2015.12.015

[7] Sommerville, I.: 'Software Process Models' *ACM Comput. Surv.*, 1996, **28**, (1), pp. 269–271. https://doi.org/10.1145/234313.234420

[8] Dahlberg, T., Lagstedt, A.: 'There Is Still No " Fit for All " IS Development Method : Business Development Context and IS Development Characteristics Need to Match', in 'Proceedings of the 51st Hawaii International Conference on System Sciences' (2018). https://doi.org/10.24251/HICSS.2018.604

[9] Cunningham, W.: 'Experience Report- The WyCash Portfolio Management System'*ACM SIGPLAN OOPS Messenger*, 1992, **4**, (2), pp. 29–30. https://doi.org/10.1145/157710.157715

[10] Behutiye, W.N., Rodríguez, P., Oivo, M., Tosun, A.: 'Analyzing the concept of technical debt in the context of agile software development: A systematic literature review'*Inf. Softw. Technol.*, 2017, **82**, pp. 139–158. https://doi.org/10.1016/j.infsof.2016.10.004

[11] Codabux, Z., Williams, B.: 'Managing Technical Debt: An Industrial Case Study'*Proc. 4th Int. Work. Manag. Tech. Debt*, 2013, pp. 8–15. https://doi.org/10.1109/MTD.2013.6608672

[12] Wiklund, K., Eldh, S., Sundmark, D., Lundqvist, K.: 'Technical debt in test automation'*Proc. - IEEE 5th Int. Conf. Softw. Testing, Verif. Validation, ICST 2012*, 2012, pp. 887–892. https://doi.org/10.1109/ICST.2012.192

[13] Heikkilä, V.T., Lassenius, C., Damian, D., Paasivaara, M.: 'A Mapping Study on Requirements Engineering in Agile Software Development', in 'Software Engineering and Advanced Applications (SEAA), 2015 41st Euromicro Conference on' (IEEE, 2015), pp. 199–207. https://doi.org/10.1109/SEAA.2015.70

[14] Moe, N.B., Aurum, A., Dybå, T.: 'Challenges of shared decision-making: A multiple case study of agile software development'*Inf. Softw. Technol.*, 2012, **54**, (8), pp. 853–865. https://doi.org/10.1016/j.infsof.2011.11.006

[15] Bocij, P., Greasley, A., Hickie, S.: 'Business information systems: technology, development and management' (Pearson Education, 2015, 5th edn).

[16] International Organization for Standardization: 'ISO/IEC FDIS 25010:2010 Systems and software engineering—Systems and software Quality Requirements and Evaluation (SQuaRE)—System and software quality models' (2010).

[17] Galin, D.: 'Software quality assurance: from theory to implementation' (Pearson Education India, 2004).

[18] Itkonen, J., Rautiainen, K., Lassenius, C.: 'Towards Understanding Quality Assurance in Agile Software Development', in 'International Conference on Agility ICAM, 2005' (2005).

[19] Kitchenham, B.A.: 'Software quality assurance'*Microprocess. Microsyst.*, 1989, **13**, (6), pp. 373–381. https://doi.org/10.1016/0141-9331(89)90045-8

[20] Kupiainen, E., Mäntylä, M. V., Itkonen, J.: 'Using metrics in Agile and Lean Software Development – A systematic literature review of industrial studies'*Inf. Softw. Technol.*, 2015, **62**, pp. 143–163. https://doi.org/10.1016/j.infsof.2015.02.005

[21] Pinto, J.K., Slevin, D.P.: 'Project success: Definition and Measurement Techniques'*Proj. Manag. J.*, 1988, **19**, (3), pp. 67–73.

[22] Taylor, H.: 'Outsourced IT Projects from the Vendor Perspective: Different Goals, Different Risks'*J. Glob. Inf. Manag.*, 2007, **15**, (2), pp. 1–27. https://doi.org/10.4018/jgim.2007040101

[23] Savolainen, P., Ahonen, J.J., Richardson, I.: 'When did your project start? – The software supplier's perspective'*J. Syst. Softw.*, 2015, **104**, pp. 32–40. https://doi.org/10.1016/j.jss.2015.02.041

[24] McLeod, L., Doolin, B., MacDonell, S.G.: 'A perspective-based understanding of project success'*Proj. Manag. J.*, 2012, **43**, (5), pp. 68–86. https://doi.org/10.1002/pmj.21290

[25] International Organization for Standardization: 'ISO/IEC FDIS 25020:2007 Software Engineering—Software quality requirements and evaluation (SQuaRE)—Quality measurement—Measurement reference model and guide' (2007).

[26] de Wit, A.: 'Measurement of project success'*Int. J. Proj. Manag.*, 1988, **6**, (3), pp. 164–170. https://doi.org/10.1016/0263-7863(88)90043-9

[27] Ika, L.A.: 'Project success as a topic in project management journals'*Proj. Manag. J.*, 2009, **40**, (4), pp. 6–19. https://doi.org/10.1002/pmj.20137
[28] Shenhar, A.J., Dvir, D., Levy, O., Maltz, A.C.: 'Project Success: A Multidimensional Strategic Concept'*Int. J. Proj. Manag.*, 2001, **34**, (6), pp. 699–725. https://doi.org/10.1016/S0024-6301(01)00097-8
[29] Petter, S., DeLone, W., McLean, E.R.: 'The Past, Present, and Future of "IS Success"'*J. Assoc. Informait. Syst.*, 2012, **13**, pp. 341–362. https://doi.org/10.17705/1jais.00296
[30] Owens, D.M., Khazanchi, D.: 'Software Quality Assurance', in 'Handbook of Research on Technology Project Management, Planning, and Operations' (IGI Global, 2009), pp. 245–263. https://doi.org/10.4018/978-1-60566-400-2.ch016
[31] Larman, C., Basili, V.R.: 'Iterative and incremental development: A brief history'*Computer (Long. Beach. Calif).*, 2003, **36**, (6), pp. 47–56. https://doi.org/10.1109/MC.2003.1204375
[32] Brooks, F.P.J.: 'No silver bullet—essence and accidents of software engineering', in 'Proceedings of the IFIP Tenth World Computing Conference' (1986), pp. 1069–1076.
[33] Hall, J.G., Rapanotti, L.: 'Towards a Design-Theoretic Characterisation of Software Development Process Models', in 'GTSE '15: Proceedings of the Fourth SEMAT Workshop on General Theory of Software Engineering' (IEEE Press 2015, 2015), pp. 3–14. https://doi.org/10.1109/GTSE.2015.8
[34] Hastie, S., Wojewoda, S.: 'Standish Group 2015 Chaos Report - Q&A with Jennifer Lynch', https://www.infoq.com/articles/standish-chaos-2015, accessed February 2018.
[35] Mahmood, M. A.: 'System development methods - A comparative investigation'*MIS Q. Manag. Inf. Syst.*, 1987, **11**, (3), pp. 293–307. https://doi.org/10.2307/248674
[36] Khan, M.A., Parveen, A., Sadiq, M.: 'A method for the selection of software development life cycle models using analytic hierarchy process'*2014 Int. Conf. Issues Challenges Intell. Comput. Tech.*, 2014, pp. 534–540. https://doi.org/10.1109/ICICICT.2014.6781338
[37] Moløkken-østvold, K., Jørgensen, M.: 'A comparison of software project overruns-flexible versus sequential development models'*IEEE Trans. Softw. Eng.*, 2005, **31**, (9), pp. 754–767. https://doi.org/10.1109/TSE.2005.96
[38] Hug, C., Front, A., Rieu, D., Henderson-Sellers, B.: 'A method to build information systems engineering process metamodels'*J. Syst. Softw.*, 2009, **82**, (10), pp. 1730–1742. https://doi.org/10.1016/j.jss.2009.05.020
[39] Boehm, B.W.: 'Value-based software engineering'*SIGSOFT Softw. Eng. Notes*, 2003, **28**, (2), pp. 1–12. https://doi.org/10.1145/638750.638776
[40] Henderson-Sellers, B., Serour, M.K.: 'Creating a Dual-Agility Method: The Value of Method Engineering'*J. Database Manag.*, 2005, **16**, (4), pp. 1–23. https://doi.org/10.4018/jdm.2005100101
[41] Yassien, E.: 'Software Projects Success by Objectives'*J. Manag. Res.*, 2017, **10**, (1), pp. 46–57. https://doi.org/10.5296/jmr.v10i1.10149
[42] Ghanbari, H.: 'Seeking technical debt in critical software development projects: An exploratory field study'*Proc. Annu. Hawaii Int. Conf. Syst. Sci.*, 2016, pp. 5407–5416. https://doi.org/10.1109/HICSS.2016.668
[43] Sommerville, I.: 'Software Engineering' (Addison-Wesley, 2011, 9th edn.).
[44] Salo, O.: 'Enabling Software Process Improvement in Agile Software Development Teams and Organisations' (2006).
[45] Besker, T., Ghanbari, H., Martini, A., Bosch, J.: 'The influence of Technical Debt on software developer morale'*J. Syst. Softw.*, 2020, **167**. https://doi.org/10.1016/j.jss.2020.110586
[46] Holvitie, J., Licorish, S.A., Spínola, R.O., *et al.*: 'Technical debt and agile software development practices and processes: An industry practitioner survey'*Inf. Softw. Technol.*, 2018, **96**, (November 2017), pp. 141–160. https://doi.org/10.1016/j.infsof.2017.11.015

[47] Yli-Huumo, J., Maglyas, A., Smolander, K.: 'The sources and approaches to management of technical debt: A case study of two product lines in a middle-size Finnish software company', in 'International Conference on Product-Focused Software Process Improvement.' (Springer, 2014), pp. 93–107. https://doi.org/10.1007/978-3-319-13835-0_7
[48] Roche, J.: 'Adopting DevOps Practices in Quality Assurance - Merging the art and science of software development'*Commun. ACM*, 2013, pp. 1–8. https://doi.org/10.1145/2538031.2540984
[49] Zohrabi, M.: 'Mixed Method Research: Instruments, Validity, Reliability and Reporting Findings'*Theory Pract. Lang. Stud.*, 2013. https://doi.org/10.4304/tpls.3.2.254-262
[50] Bryman, A., Bell, E.: 'Business research methods' (Oxford University Press, 2015, 4th edn).
[51] Flick, U.: 'An Introduction To Qualitative Research' (SAGE Publications Ltd, 2010, Fourth).
[52] Capgemini, Hewlet Packard, Sogeti: 'World Quality Report 2016–17' (2017).
[53] Solinea: 'DevOps 2017', https://www.solinea.com/open-infrastructureservices/devops, accessed June 2017.
[54] Kaplan, B., Maxwell, J.A.: 'Qualitative research methods for evaluating computer information systems', in 'Evaluating the organizational impact of healthcare information systems' (Springer, 2005), pp. 30–56. https://doi.org/10.1007/0-387-30329-4_2
[55] Williams, P.: 'Future Trends and Development Methods in Software Quality Assurance'. Haaga-Helia University of Applied Sciences, 2017.
[56] Fairley, R.E., Willshire, M.J.: 'Better Now Than Later : Managing Technical Debt in Systems Development'*Computer (Long. Beach. Calif).*, 2017, (May), pp. 80–87. https://doi.org/10.1109/MC.2017.124

8 Authors

Altti Lagstedt, holds a PhD in information science and an MSc in technology. His expertise relates to information systems development methods, software development and business process digitalization, for which he has written several academic and practical publications. Altti is working in Haaga-Helia University of applied sciences as a Principal lecturer. Before the current position, he has been a software developer and a project manager at a software development company and a as researcher in LUT university. E-mail: altti.lagstedt@haag-helia.fi

Dr. Amir Dirin, is an adjunct professor in educational technology at the University of Helsinki and lecturer at Haaga-Helia university of applied science. His main research area is in immersive experience, machine learning, and user experience design and development. E-mail: amir.dirin@haga-helia.fi

Päivi Williams, Business Information Technology, Haaga-Helia University of Applied Sciences, Helsinki, Finland.

Article submitted 2020-11-29. Resubmitted 2021-05-21. Final acceptance 2021-05-21. Final version published as submitted by the authors.