Software Quality Journal
https://doi.org/10.1007/s11219-021-09561-2

Exploring human factors of the agile software tester

Viktoria Stray[1,2] · **Raluca Florea**[1] · **Lucas Paruch**[1]

Accepted: 25 May 2021

Abstract
Although extensive research has been conducted on the characteristics of the agile developer, little attention has been given to the features of the software-testing role. This paper explores the human factors of the software testers working in agile projects through a qualitative study focusing on how these factors are perceived. We interviewed 22 agile software practitioners working in three international companies: 14 testers, five developers, and three designers. Additionally, we observed 11 meetings and daily work of 13 participants in one of the companies. Our findings show that the views on the human factors shaping the agile software tester's role were crystallized into seven traits, which the agile team members saw as central for the software-testing role: the ability to see the whole picture, good communication skills, detail-orientation, structuredness, creativeness, curiosity, and adaptability. The testers spent half their day communicating and learned how to mitigate the fact that they had to bring bad news to other project members. They also facilitated communication between the business side and development. Based on our results, we propose the seven traits as dimensions to consider for organizations recruiting agile software testers, as well as a reference for IT and non-IT professionals considering a software-testing career.

Keywords Software testing · Human traits · Soft skills · Agile software development · Agile tester

1 Introduction

Agile software development is now widely adopted among companies seeking to improve their industrial competitiveness (Yang et al., 2016). Agile methods are based on iterative and incremental development, characterized by qualities such as short cycles and rapid customer feedback. As advantageous as it may be, working in an agile environment brings challenges such as parallel information flow, communication barriers, and lack of alignment (Dingsøyr et al., 2019; Ghobadi & Mathiassen, 2016). Software testing is a pivotal activity in agile software projects, to ensure the quality of the

✉ Viktoria Stray
stray@ifi.uio.no

[1] Department of Informatics, University of Oslo, Oslo, Norway

[2] SINTEF Digital, Trondheim, Norway

Published online: 16 June 2021

software product throughout the iterative development process, with frequent releases (Korhonen, 2013).

In agile, a noticeable divergence from traditional plan-driven development methods entails involving testers from the beginning of each development increment. Therefore, dedicated testers can plan and complete various aspects of the test strategy, such as exploratory testing, usability testing, and improving test coverage with the developers (Bai et al., 2017; Santos et al., 2011). In such work environments, knowledge transfer also occurs more frequently and naturally between developers and testers, as both testing and development happen concurrently during each iteration. Even though testers had a difficult time finding their place in teams during the early phases of agile adoption, they quickly became integrated and recognized as an important part of agile software development (Cohn & Ford, 2003).

The skills within development teams are crucial to the success of software projects because they directly affect central aspects of software attributes such as performance, reliability, and simplicity (Byrd & Turner, 2001). There have been numerous studies on how team members' skills affect the qualities of software as well as teamwork performance and the competitive market advantages those skills bring (Capretz & Ahmed, 2010; Ebert & De Neve, 2001; Faraj & Sproull, 2000). Alternatively, as many researchers have shown, a lack of skills within development teams directly affects the costs of software projects, their delivery times, and even their completion (Jiang & Klein, 2000).

In general, the literature presents testers with an emphasis on their hard skills—for instance, as the ones responsible for carrying out testing and building up test cases and test plans (Mathur & Malik, 2010). Davidov et al. (2010) described testers as professionals who identify new defects from failed test cases, analyze defects, and report them in a bug-tracking system. However, the testing process requires not only technical skills but also specific socio-technical abilities (Florea & Stray, 2018; Sánchez-Gordón et al., 2020); therefore, an efficient software tester is needed to cover a broad area of abilities and expertise, frequently extending beyond testing. For instance, in agile, software testers play a key role in connecting developers with development stakeholders and customers by intimately understanding and tracing business requirements throughout development (Saiedian & Dale, 2000). Nevertheless, there is a need for further studies focusing on agile testers' soft skills (Sánchez-Gordón et al., 2020).

In agile, attention has been given to developers' skills from both industry (Capretz & Ahmed, 2010; Ebert & De Neve, 2001; Faraj & Sproull, 2000) and educational perspectives (Lethbridge, 2000; Lindstrom & Jeffries, 2004). However, the same cannot be said about the testers' skills, as the role was largely considered a junior or entry position, and testing seen as a side activity (Deak et al., 2013; Juristo et al., 2006).

To better understand the agile software tester, we conducted an exploratory case study in three companies. The aim of this paper is to fill a research gap within the area of software testing, focusing on the human dimension of the software-testing role. Although there has been extensive research on the technological aspects of software testing—such as tool usage, test automation, and test processes—little research has been conducted on the human factors central to the role of the software tester (Deak et al., 2016; Kanij et al., 2015). The current paper extends our preliminary findings on the human factors of the agile software tester from a single-case study involving 13 participants (Paruch et al., 2020a, 2020b).

We aimed to answer the following research question: *Which human factors are essential for the agile software tester?*

The remainder of this paper is structured as follows. First, Sect. 2 presents the background and related work on the agile software tester. Section 3 reports on our case study methodology. Section 4 presents the results of the study. Section 5 discusses the results and implications for practice as well as the threats to the study's validity. Finally, Sect. 6 concludes the paper.

2 Background and related work

In this section, we first present the background of the agile software tester. Then, we present related work on human aspects of software testers through a literature review.

2.1 Software testing in agile

In agile, a noticeable difference from sequential development methods entails involving testers at the beginning of each development iteration, instead of end-of-phase testing. One of the seven testing principles states that quality-assurance activities should be started as early as possible in the life cycle to avoid additional cost and time (ISTQB, 2019). In agile, early involvement of testers at the start of the iteration enables an emphasis on the user stories and the system architecture, with which primary testing personnel will acquire a better understanding of the testing scope.

Two studies focused on the differences between the test activities conducted in an agile project compared to a plan-driven project (Dhir & Kumar, 2019; Kettunen et al., 2010). Kettunen et al. (2010) performed a study on agile projects and identified the absence of guidance on how testing should be arranged at the same time as development. Dhir and Kumar (2019) compared the testing of a Web application in a plan-driven project versus in an agile project. Their results showed an improvement in the agile project compared to the traditional one: the agile project had increased test coverage, reduced costs, and improved testing productivity.

Deak (2014a) identified and ranked testers' motivation factors. The results showed that the testers working in plan-driven projects experienced more stress but manifested a more positive attitude for tackling challenges than their agile counterparts did. The agile testers were better integrated into their teams but expressed distress in their relationship with the developers, with whom they found it demanding to communicate.

The delegation of testing tasks is also an essential feature of agile, as developers are expected to perform unit tests independently. As such, testers can focus on test techniques such as exploratory testing and usability testing and improving the test coverage jointly with the developers (Bai et al., 2017; Santos et al., 2011).

In agile, knowledge transfer also happens more frequently and naturally among developers and testers, as testing and development occur concurrently in each iteration (Li et al., 2010). While new team members benefit from the ongoing feedback in their onboarding, this competence sharing also contributes to less misunderstandings or confusion during development. The tight coupling among testers, developers, and stakeholders enables fast learning and greater mutual understanding.

2.2 Related work

As we found limited research on the human factors of the *agile* software tester, we aimed to provide an overview of the human factors in software testing, regardless of development methodology. The literature review was performed from September 2019 to December 2019, with the main goal being to map the existing studies on the human factors in software testing and the secondary goal of finding research gaps and needs for further studies. We looked to the literature to answer the research question: *Which human factors are essential for the software tester*?

2.2.1 Review method

The search string presented in Table 1 was modified to fit the syntax of the different databases; however, the semantics were ensured to be consistent. We included the search term "quality assurance" because some of the research papers use quality assurance instead of "software testing," and the two terms generally are interchangeable in the agile terminology. We followed Kitchenham and Charters' (2007) suggestion to select search criteria meant to identify the primary studies that provide direct evidence about our research question. In addition, we decided on the criteria before the study selection to reduce the likelihood of bias.

We devised the following protocols, which included both inclusion and exclusion criteria. In the inclusion criteria, we considered the relevant academic and industrial studies, both qualitative and quantitative, as well as relevant conference papers, journal papers, workshop papers, review articles, research articles, book chapters, and conference papers published in the timeframe 2009–2019. We excluded from the analysis any non-English contributions, studies not related to software engineering, encyclopedias, prefaces, book reviews, case reports, correspondences, tutorials, editorials, and news reports.

We obtained papers through four stages. The reviewing process was initiated by applying the search string to four scientific databases: Scopus, Science Direct, IEEE Xplore, and ACM (Stage 1). The application of the inclusion criteria and exclusion criteria and the removal of duplicate studies resulted in 1160 papers, with most stemming from Scopus (956 papers). All of the articles underwent a scrutinization phase where we inspected the title, abstract, and keywords for relevancy to our goal. We selected the relevant literature by removing studies without references to human factors and software testing. Most of the initial search results were excluded from further analysis, as they did not fit the general objectives of our research (Stage 2) and belonged primarily to the medical and psychology fields. After skimming through the 1160 papers, a total of 22 studies were included and read carefully. Seven of these papers did not focus on the human factors within software testing and were excluded from further analysis (Stage 3). In Stage 4, we used the snowballing technique and found two additional articles on the human factors in software testing. As such, at the end of the selection process, we had obtained 17 studies relevant to the research question.

For the 17 selected papers, we extracted data manually and organized the information in an Excel spreadsheet. For each paper, we extracted the title, names of authors, publication year, source title, number of citations, and whether it was a conference or journal paper.

Table 1 Search string for the literature review

("human factors" OR "soft skills") AND	("software testing" OR "quality assurance" OR "QA" OR "software quality assurance" OR "SQA")

When reading the papers in detail, we highlighted passages and put relevant themes and comments into the Excel sheet. By analyzing the objective of the papers, their results, and the discussion of the findings, we found three themes emerging from the papers, and we grouped the 17 papers by the theme they shared: (1) software testing as a profession, (2) motivational factors for software testers, and (3) the personal characteristics of software testers. The grouping of the papers is shown in Fig. 1, and the findings in the three categories are presented next.

2.2.2 Software testing as a profession

Research has been conducted before on software testing as a profession. Capretz et al. (2019) conducted a quantitative survey among professionals in four geographic regions to determine the profession's degree of attraction. Their results indicate that testing was not a popular career option among software professionals. Among the enumerated reasons were the role's perception as a lower-competence one and the testing job's complexities resulting in stressful and frustrating situations. Shah and Harrold (2010) mentioned that senior testers in a service-based software company located in India voiced similar opinions. Most of the seniors had a negative attitude toward testing and considered it as something that

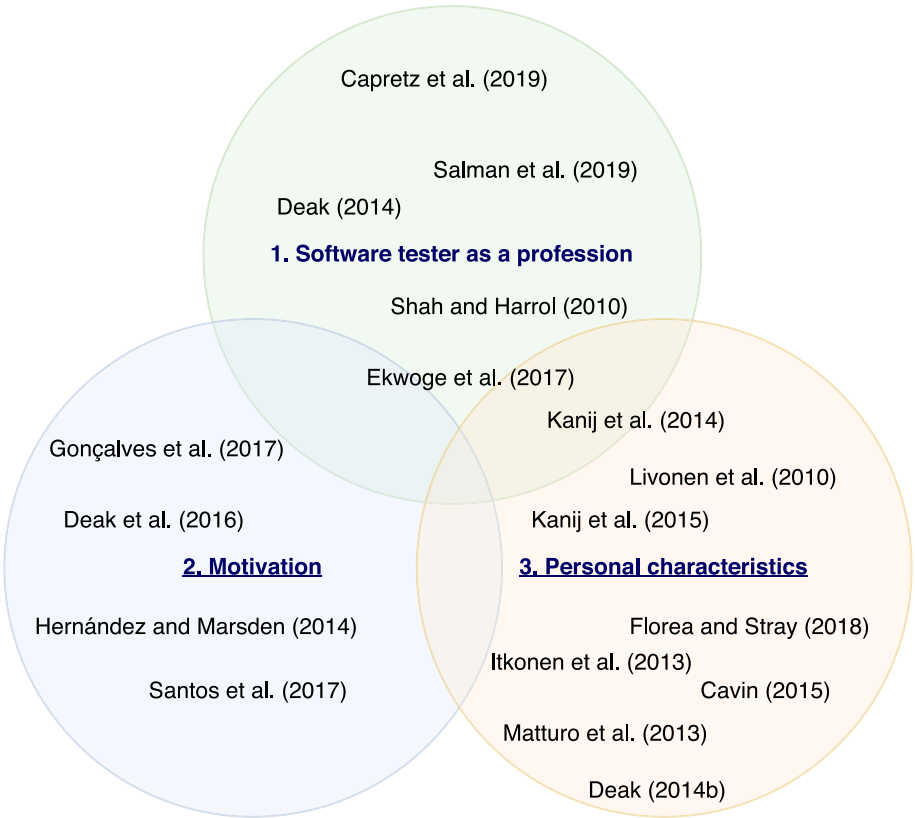

Fig. 1 Three themes in the related literature

just needed to be done. The junior testers had a positive attitude toward the job, stating that testing helped them to learn the system better so they could become skilled programmers in the future. The study found that all but one participant did not want to work permanently as a tester.

Salman et al. (2019) conducted a controlled experiment to find out if testers exhibit confirmatory behavior—also known as positive testing—when designing functional test cases, and whether such behavior increased under time pressure. The findings resulted in the conclusion that confirmatory test cases were present regardless of time pressure; it is therefore necessary to make testers aware of the danger of confirmation bias (the tendency to look for evidence that strengthens his/her prior beliefs) and learn how to design test cases with a disconfirmatory attitude (Salman et al., 2019).

Ekwoge et al. (2017) identified three main categories affecting testers: cognition, conation, and affection. Cognition refers to memory, problem-solving, and decision-making and how testers perceive the testing infrastructure. Conation includes impulse and desire and how testers see the value of their contribution. Affection involves elicited feelings and emotions, and influences testers' respect, team belonging, and social factors. Deak (2014a) found that agile testers were more unhappy about their relationship with developers than testers working in traditional plan-driven development. However, those testers in traditional development reported a higher degree of stress.

2.2.3 Motivational factors for the software testers

Santos et al. (2017) point out the importance of highlighting the testing activities as "a set of human-dependent tasks," therefore emphasizing the need for research within motivation. They argue that five factors influence the motivation of software testers: acquisition of useful knowledge during work, work variety, creativity in solving tasks, well-defined work with the precise sequence of steps, and recognition of work. The authors highlight the last factor, which has a lasting impact on the testers' motivation as well as an increase in individual productivity and teamwork enhancement.

Deak et al. (2016) also report similar motivational factors, such as enjoying challenges, variety of work, and recognition such as positive feedback received from both management and developers. The authors also identified de-motivational factors, exemplified by time pressure, poor relationships with developers, lack of clear processes, redundant meetings, or lack of influence and recognition (Deak et al., 2016). Moreover, some of the participants in this study mentioned the tedious routine of some testing activities and the "feeling of boredom," which would furthermore increase the assumption that the profession is unattractive.

Hernández and Marsden (2014) investigated the challenges software testers face and how they collaborate with other teams. Their findings showed that the testers were highly motivated. They were motivated by the broad variety of topics they could work on and that they were able to have a complete view of the software. They were also intrinsically motivated when they experienced autonomy.

Gonçalves et al. (2017) identified three main types of factors influencing the software profession: cognitive, operational, and organizational. The cognitive aspects included stress, psychological pressure, and retention of information under mental workload. The operational aspects included conflict, receptiveness, and monotony, whereas the organizational aspects included a lack of training, participation, and division of activities. The study showed that professional testers face many demotivational factors such as outdated testing

environments, demobilization, and the devaluation of testing careers—which were often seen as a mere extension of development.

2.2.4 Personal characteristics of the software testers

Kanij et al. (2014) found that human factors are crucial in software testing and that important traits for testers are being open-minded and having curiosity and "attention to details." Kanij et al. (2015) argued that the effectiveness of a tester role was related to their personality and that testers had significantly higher levels of conscientiousness compared to those in other software engineering roles. In the study, conscientiousness was related to being disciplined, hardworking, and dedicated. Although highly conscientious individuals were important in any profession, Kanij et al. (2015) suggested that this quality might be particularly important for testers. Deak (2014b) studied personal characteristics among software testing professionals and found that the most valuable characteristics to possess included communication skills, need for variety, and being detail oriented and curious.

Livonen et al. (2010) explored the characteristics of high-performing testers, identified as such by a high defect-detection rate or by possessing traits that managers and other testers see as important. The authors found four themes: experience, ability to reflect, motivation, and personality. Within those themes, the top characteristics of the high-performing testers were thoroughness, carefulness, patience, and conscientiousness (Livonen et al., 2010). Itkonen et al. (2013) investigated the knowledge types used by testers in exploratory testing and obtained three categories: domain knowledge, system knowledge (the act of knowing the system's mechanisms, logic, and interactions), and generic software engineering knowledge (knowledge of the system's usability and the ability to interpret error messages).

Two studies focused on the soft skills required by software testers. Matturro (2013) analyzed 43 recruitment advertisements to investigate the frequency of demand for soft skills. Among the most common soft skills for software testers were teamwork, proactive, analytical/problem-solving skills, and being methodic. Similarly, Florea and Stray (2018) analyzed 400 recruitment advertisements for software testers across 33 countries, following a pre-existing skill taxonomy. The most frequently solicited traits were the ability to communicate both verbally and in writing, analytical and problem-solving skills, team spirit, and independent-working skills.

Cavin (2015) assessed the viability of military veterans in becoming software testers by initiating a coursework program. Findings show that most veterans possess human factors that were aligned with the characteristics of a tester—such as communication, team player, and flexibility.

3 Case study methodology

To answer our research question, we conducted an exploratory multiple case study (Yin, 2018) following the guidelines proposed by Runeson et al. (2012). The qualitative research process included collecting data from a context-specific environment, analyzing the data, and interpreting the data.

We collected the data through interviews and supplemented them with observations. We interviewed 22 practitioners in three companies. In addition, we observed 11 meetings and

13 of our participants in their daily work. We attended eight daily stand-up meetings, two test status meetings, and one domain-expert workshop.

3.1 Case contexts

Company A is a medium-sized software-service-providing company with over 500 employees in Norway, Denmark, Ukraine, and Slovakia. The company offers expertise in project management, software testing, software development, interaction design, maintenance, and security. The organization consists of over 1000 employees in Scandinavia and has focused, among other things, on automotive financing, sales financing, and loans. The interviewed professionals used agile methods and mainly Kanban development, combined with Scrum ceremonies. They used product backlog items extensively—usually in the form of a user story. Some of the interviewees had no time-boxed sprints, instead working with a continuous stream of tasks, as they appeared in the backlog. They were involved in prioritizing tasks and maintaining the product backlog. The testing professionals also participated in the test-automation processes. The team used mainly Slack, an instant communication tool that has 12 million active users daily (Novet, 2021). The tool allows written, verbal, and video communication and supports coordination and problem-solving in teams (Stray & Moe, 2020).

Company B is a large-size supplier of software and services, mainly in the business, accounting, resource-planning, procurement, and retail areas, with over 9000 employees in Scandinavia, Europe, and South America. All of the interviewed software practitioners were members of agile teams, working distributed in two countries. These practitioners were parts of different teams, yet their agile practices were rather similar. The teams had 6–10 members and a team leader who acted as Scrum master. The product owners were external to the team, which used 2-week sprints, with the product backlog as a central element in prioritizing and assigning tasks.

Company C is a large financial services group provider whose main offices are located in Norway but with branches throughout the world and a medium-size software-development department. The interviewed practitioners were parts of different teams, working in modified agile processes. The sprints spanned from weekly to monthly and could vary in length, depending on the potentially shippable product's readiness.

3.2 Interviews

We conducted semi-structured interviews of 22 software professionals: 14 testers, five developers, and three interaction designers (see Table 2). We used purposeful sampling (Patton, 2014) to provide rich data on the factors perceived as central to the software-tester role. The interviews were conducted between October 2019 and February 2020. A preliminary analysis of 13 of the interviews is reported in Paruch et al. (2020a, 2020b), and seven of the interviews are partly analyzed in Florea and Stray (2020).

All of the respondents (12 women, 10 men) consented to the recording of the interviews and the printing of the results. Where necessary, we furthered our understanding of our participants' responses with follow-up questions and confirmed that we had captured their responses in entirety. We asked targeted open-ended questions, and we gave appropriate feedback by encouraging our respondents to talk and by reflecting on their remarks. Simultaneously, the interviews were, to a great extent, protected from outside interruptions and from competing distractions.

Table 2 Overview of the interviews

	Informant no	Current role	Work experience	Interview method	Interview length
Company A	1	Software tester	7 months	Face-to-face	53 min
	2	Software tester	1 year	Face-to-face	1 h 3 min
	3	Software tester	4 years	Face-to-face	51 min
	4	Software tester	6 years	Face-to-face	1 h 6 min
	5	Software tester	3 years	Face-to-face	41 min
	6	Software tester	11 years	Videocall	30 min
	7	Software developer	1 year	Face-to-face	24 min
	8	Software developer	2.5 years	Face-to-face	20 min
	9	Software developer	2 years	Face-to-face	26 min
	10	Software developer	10 years	Face-to-face	38 min
	11	Interaction designer	1.5 years	Face-to-face	37 min
	12	Interaction designer	4 years	Face-to-face	41 min
Company B	13	Software tester	19 years	Videocall	55 min
	14	Software tester	14 years	Videocall	1 h 5 min
	15	Software tester	21 years	Videocall	1 h
	16	Interaction designer	11 years	Videocall	50 min
	17	Software tester	35 years	Face-to-face	1 h 8 min
	18	Software tester	24 years	Face-to-face	1 h 5 min
	19	Software tester	11 years	Face-to-face	55 min
Company C	20	Software developer	1 year	Face-to-face	23 min
	21	Software tester	17 years	Videocall	1 h 10 min
	22	Software tester	15 years	Videocall	50 min

We employed semi-structured individual interviews because they allowed us to gain insight into each participant's views on the human software-tester factors, based on their work experience. An extract of the questions we asked the interviewees, separated into testing and non-testing positions, is presented in Table 3. The interview guide was slightly modified as our research progressed. Appendix A shows how the full interview guide for the software testers looked in our main data collection period. We held 15 face-to-face interviews and seven Skype video sessions. The shortest interview lasted 23 min, while the longest lasted 1 h and 10 min; the average length was 52 min.

3.3 Observations

To collect additional information for the study, such as the relative position of testers with reference to the other team members, as well as the attitude toward testers, we conducted observations in one project taking place in a customer organization of Company A. We observed the project from November 2019 to January 2020. The meeting observations included daily stand-up meetings, status meetings, and a workshop (see details in Table 4). During these meetings, we gathered information on the participants' identities and their role within the team and company, the duration of the meeting, the topics discussed, and who facilitated the meeting. Additionally, we took notes on the testers' behavior during the meetings and our personal impressions of what was happening.

Table 3 Extract of the interview guide used for the testing and non-testing positions

Testing role	Non-testing role
• Describe your role in the team, with typical tasks and assignments. Give me some examples of your responsibilities. • What kind of testing do you like best? Why? • When practicing your job, describe your knowledge in testing and beyond testing. • What can you tell me with regard to your connection with the team (roles, team members)? • Have there been challenges for your role related to testing? • What traits should a good software tester have? Why? • Are your skills measured and followed up at work? How? • How do you discover a significant bug? • Are some testing skills more complicated to acquire? • Who do you collaborate with the most? For how long?	• Tell me about your role and assignments. Give me some examples of test-related responsibilities you had. • How do you collaborate with software testers? In which contexts? • In what ways do you appreciate your collaboration with the software testers? • How would you describe your software-testing collaborators' abilities? • How would you describe a good tester? What about a sub-optimal tester? • Are any traits or skills more important for a tester to have? • How do you feel that software testing is integrated into the team's practices?

During the workday, we recorded the timestamp and description of the events relevant to our research, such as conversations between the testers and other team members, interactions between the testers, or calls for help involving testers. We also recorded information on each team member's current tasks. When observing casual and formal interactions during the workday, we followed the same procedure as above, noting the timestamp and the participants during the interaction. We aimed to note down as much details as possible.

Additionally, we had access to all the participants in Company A on Slack, and could observe the participants communicating in an open Slack channel for the testers. The observation notes, together with Slack logs, were used to extract additional support for the data collected through observations and interviews. For example, we observed what the participants discussed in the channel that could shed light on our findings and we also used Slack for follow-up questions to the interviewees.

Table 4 Overview of meetings observed

Meeting	Number of participants	Length of meeting
Daily stand-up	7	8 min
Daily stand-up	7	12 min
Daily stand-up	8	12 min
Daily stand-up	6	8 min
Daily stand-up	8	11 min
Daily stand-up	8	14 min
Daily stand-up	8	15 min
Daily stand-up	8	17 min
Test status meeting	9	27 min
Test status meeting	8	38 min
Domain expert workshop	6	2 h 40 min

3.4 Data analysis

We chose thematic analysis (Braun & Clarke, 2006) for the data analysis, which we found to be a flexible approach for identifying, analyzing, and reporting patterns, or themes, within qualitative datasets (Braun et al., 2019). We approached the analysis inductively, using the data as a starting point to generate themes.

Through familiarization and coding, we identified patterns of meaning, with which we generated themes, which we reviewed in a constant, iterative process. We used meeting records and observations to check how well the themes worked, both individually and overall.

We coded the collected data and grouped the codes into themes (see Fig. 2). For this purpose, we used NVivo because it enabled us to organize the codes and explore the relationships between them easily. During our analysis, more than a hundred codes emerged, which we grouped into seven traits characteristic of the agile software tester role, as described by both the testers themselves and the other team members. Figure 2 shows an example of how the raw data were processed into codes and then allocated to a theme.

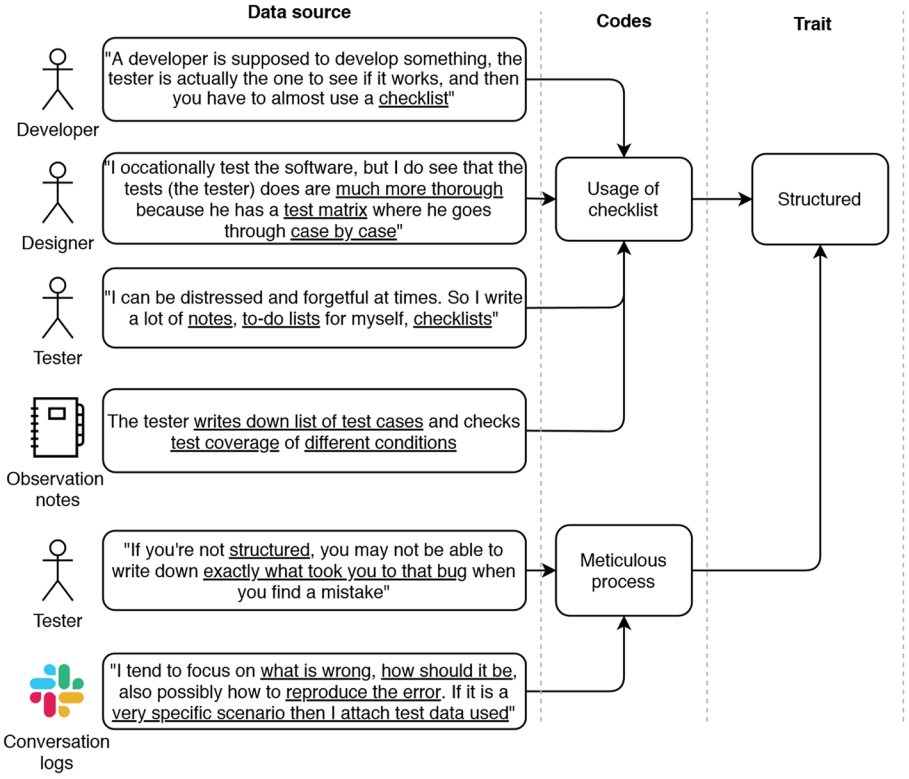

Fig. 2 Example of the thematic coding of the collected data

4 Results

Regarding the discussion on the need for dedicated software testers in agile, we found the agile software-development projects needed testers. As Informant 17 reflected, "I have been on several agile projects where they did not want a tester or a test manager because they had a product owner who tested, they had the developers who tested. But then they pushed that button, and kaboom, the system went down. And they said, 'Oops.' They didn't see that. Now, it's the tester experience that sees that." Informant 17 further reflected on how agile ways of working is changing testing: "So I think it will go a number of years now where you have a shift over to developers doing testing, and then you have a shift back and adjust somewhere in the middle."

Regarding the identified human factors characteristic of the software-testing role, our analysis of the data revealed seven traits of the agile software tester, as presented next and depicted in Fig. 3.

4.1 Ability to see the whole picture

All of the professionals stated that having to understand the project as a whole is essential for an agile tester. The ability to see the high-level aspects of the project objectives, customer requirements, quality demands, and both domain knowledge and technical knowledge was voiced as a central tester trait.

Informant 15 described this factor: "Somehow, they will have to intermediate the customer's wishes with what is technically possible, even though the developers say that everything is possible. It's in fact the architecture, as it is at that point [when it] is too far off to be able to do something. So, the testers need to find that balance. So, somehow, they have to listen to everyone."

Informant 3 indicated that the trait was significant with the following example: "It's not about finding the most bugs or running the most test cases. A tester who completed one test case but has found that the system does not work how it is supposed to is often more

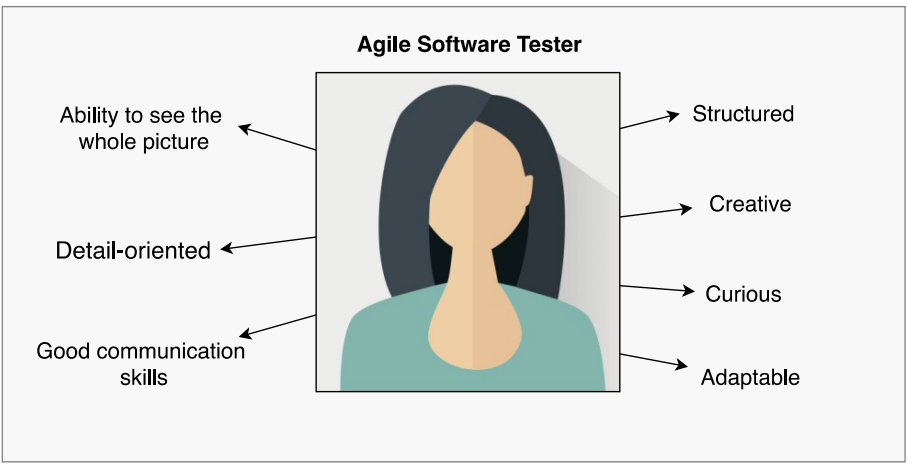

Fig. 3 The seven traits of highly effective testers

effective than a tester who ran a hundred test cases and has not found anything. It's about being able to see the total picture."

An interviewed interaction designer recalled that the testers had impressive domain and technical skills as well as the ability to use both in dealing with high-level project aspects: "He is the only one in the team who has a good overview of how things are connected, a helicopter perspective, how insurance works, the scope of different insurance types, [and] how things are connected [on the] back-end." (Informant 12).

A developer, Informant 1, justified this factor as rather specific to the tester role, as follows: "Often, as a developer, you receive a task and you do it. Developers are not very involved in the entire process. I think it's more exciting to be involved in both the business and technical aspects, not just super-technical." The need for this first trait was summed up by Informant 17: "Some companies say that the developers can do the job. But they forget that the tester is the role that keeps everything, the focus and the overall quality. Sometimes, you need someone to see the bigger picture."

4.2 Detail-oriented

A good tester needed to be a person who was "paying attention to details" (Informant 16). The testers stated that they had to manage a considerable amount of details on things they worked on. One interviewee in an interaction designer role (Informant 12) mentioned that the testers with whom they worked helped to identify user-experience issues that the designers themselves had overlooked: "I remember he went through an old application, he began to carefully read the text that was there. I thought we were pretty attentive on death insurances—the fact that someone needed to report it when a person dies. He tested the whole process and made me aware of the text; it wasn't pleasant in a highly sensitive situation. So yeah, he is very aware of details."

In addition, the interviewed developers mentioned that good software testers needed to have attention to detail: "This is why I love working with them, because of exactly that—they pick up things that we don't see and things we haven't even thought of. They inform us that 'This is wrong' and 'This wouldn't work because of this method and this module'—very logical individuals" (Informant 10). Moreover, one developer (Informant 7) remembered that it gave them a feeling of safety if a tester was available during development: "If it's a complex product, then it is nice to have a tester next to you. It gives me safety in the form of, 'Okay, he confirms that everything I do is correct'."

Informant 15, who was in a testing role, described attention to detail with a technical perspective: "One thing that it is not understood by those who don't do automated testing is that […] they think that everything is out of the box. They don't realize that you have to go into the code and actually have to check which data to take, which data to send, how to execute tests. And the automatic tests have to be maintained all the time."

4.3 Good communication skills

According to our interviewees, and based on the results of our observations, agile software testers spend as much as 50% of their time communicating with others. Agile also increased the frequency of communication, particularly "the communication and the contact with the customers. Because if you release two times a year, you contact people two times a year. But here, you can be in contact with the customers on a daily basis." (Informant 19).

The testers explained that they were often in the "buffer zone" between business and development. Informant 15 said that they often had to facilitate communication between people to make them talk to each other. "It's very important, that part with communication, when you must talk to people and you need to make people talk together—to organize them a bit, because some might not even talk with each other otherwise." Informant 3 emphasized how working with people was one of the motivational factors for her: "What motivates me in my job is demanding tasks that require that I use my head and working with a lot of people that is always fun." Informant 2 stated how he was motivated by working with people in a team.

One developer, Informant 8, described the business-oriented aspect of the testers' communication: "They are very active in meetings with product-owners and ask questions about functionalities and what they need to find out in order to set up test-scenarios." Informant 13, who works in a testing role, confirmed this point: "What I found is that my accountant background is really useful because I know the accountant language, and the developer doesn't know that. So, I sit in the middle and translate between the customer and the developer—and that is helpful. You need to know a bit about how the customers work and what info the developers need in order to do a good job. You basically need communication skills."

Most of the testers mentioned that they communicate mostly—and rigorously—with the developers. Two testers mentioned that they talk mostly with developers because they are the ones requiring additional information while fixing the bugs. The ability to provide more information in an understandable, coherent way was therefore essential. One tester noted that the bugs described by business staff were not always detailed: "…we have many testers from the business aspect, and they vary in how good they are at describing. I often need to add additional information so that the developers don't have to engage in dialogue with them." (Informant 2).

The agile testers with less work experience communicated mostly with the developers, while the experienced ones communicated with other stakeholders besides developers, such as product owners and support, as they considered them to have goals that otherwise might be overlooked. A software tester (Informant 3), described the communications network: "If the specification is too vague and we find that there can be many different ways to interpret it, then often I'm the one who has to go ask the ones who wrote the specification and find out exactly what they mean because I'm the one who specifies concrete requirements to the developers."

When communicating, the testers needed to provide constructive feedback regarding the issues discovered and to mitigate potential negative team dynamics. For example, one software tester (Informant 3) mentioned, "I was on a project a few years back where I sat next to the developers. When I found a bug, I stood up and walked towards them with a friendly smile." Informant 5 mentioned "Whenever I find a bug, I go to the developer and ask him if it's supposed to be like that. I try not to point any fingers because that's never pleasant for anyone, and it's not appreciated." One interviewee stated, "You will always make mistakes, and it's important not to be afraid of mistakes, but it's important to learn from them. So, if I see a bug and don't discuss the bug with people for a while, then it would be a problem if I discuss the bug at a later point. Then, knowledge gets lost" (Informant 17).

In communicating with the developers, the testers were often bearers of bad news. One tester described his task in these cases: "You have to talk to people, and not everyone is super-pleased to hear what you have to say. People are stressed and not really glad to hear about problems" (Informant 15). Another interviewee stated (Informant 22): I have been in an environment with the pressure of delivering. Everyone is yelling at each other and

hating each other. I do not want to be that person who says, 'We are not ready. I found another bug.' But, it is my responsibility, so I let them know."

4.4 Structured

During the interviews, the software testers emphasized that they were organized and structured and that they enjoyed creating and using checklists; they mentioned that they usually noted everything in lists, in notes, or in their calendars. One tester (Informant 2) stated that in order to succeed as a tester, one had to be organized: "Even if one performs exploratory testing—through just playing around—if you're not structured, then you might not be able to describe the steps you performed when you found a bug. If you're just exploring without being systematic, then I don't think you can retrace your steps. So, in my opinion, all testers must be structured."

The testers not only appreciated their own use of structure but also thrived in organized job environments: "I applied for a job and got it. And what do I encounter? No structure. […] I can't work in these conditions, so I applied for another job." (Informant 18).

A developer (Informant 8) explained how the project became significantly more organized when the testers joined the team. He mentioned that the project lacked a concrete work process and had vague requirement specifications, in addition to a general lack of documentation: "The testers worked really hard to systematically map, find, and clarify the requirement specifications for us, and even found things that we overlooked. And, as a result, we also adjusted and improved our work processes."

4.5 Creative

The interviewees described the importance of being creative in testing. Creativity allowed them to find abnormal bugs. One participant (Informant 3) described the role of creativity in this way: "I've managed to find weird bugs by being creative, such as mid-way force shutdowns and performing unusual process-sequences. One has to test like that because the users are always creative."

The interviewees emphasized that creativity was necessary for the profession. Informant 17 said, "Testers need to have imagination, a passion for the end-users and a passion for quality." However, one participant (Informant 2) who worked in a testing role explained that the team could choose not to fix all the limit-scenarios: "I daresay I use a lot of creativity to the point that I was told, 'the bug you've reported is so specific that it only affects one specific customer during a leap year, so we're not going to fix it'."

One of the interviewed designers (Informant 11) mentioned that they found the collaboration with the testers to be valuable, as the testers were creative in solving issues such as suggesting several alternative ways to resolve obstacles: "One finds strange things and loopholes by being creative. This is very much appreciated from a tester because we (non-testers) have "tunneled" ways of testing, […] meaning we would only test the system's behavior when we do things right."

One interviewee working in a developer role (Informant 10) mentioned that creativity was important for a tester to possess, mainly because of the differences in the focus of the role: "As a developer, you receive a business requirement. Your job is only to fulfill those requirements. You can have a creative process where you construct the architecture, choose frameworks, etc. But in the end, you're fulfilling a requirement. Testers are supposed to

test a system that's going to work 100%, and there could be many anomalies. So a creative tester is most likely much more important to have on a team than a creative developer."

One developer (Informant 7) referred to both developers and testers as creative, but in different ways: "On the developer side, I reckon it's more how one constructs things or puts them together, while for the tester, it's more like, 'How can I find ways to destroy the system?'" The developers also depicted the testers as creative when verifying the non-functional attributes of the system in development, such as system flow, memory leaks, and security checks.

Several testers appreciated working with agile because it gave them and their teams considerable freedom in their projects. One tester (Informant 4) remarked that projects with a high level of autonomy allowed creativity to flourish and encouraged new ways of testing. However, she also stated that to become creative, one needs to have enough experience in both the business and technical areas of knowledge, "[…] because if you don't know it, then it'll be hard for you to be creative. You need to understand the domain, know enough about domain knowledge and technical knowledge in order to open the doors to creativity." They also needed enough time, one of the testers mentioned that they did not always have the opportunity to be creative, mainly due to time pressure. The participant added that some of the bugs that were found post-testing could have been found earlier if the testers had the time to play more with the system undergoing tests.

4.6 Curious

Curiosity was perceived as a key aspect of the testing role, as Informant 21 illustrated: "You need to be curious, you need to have a desire to break the system. Although you don't necessarily need to break the system, you need to have such an urge." Most of the testers brought up that they were curious and eager to learn when it came to meeting the unknown. One (Informant 2) stated that continually learning new things was their passion: "I think it is important as a tester to be eager and possess a wish to learn. I think acquiring domain knowledge quickly and using it […]. It goes without saying that the more you know, the better you can conduct testing."

Furthermore, Informant 2 explained that his curiosity helped him to ask basic but useful questions such as "I'm not familiar with that, can you tell me more about it?" They stated that the stakeholders and the customers of the project perceived this interest as commitment, and opened up towards the testers.

One tester (Informant 5) mentioned that their curiosity in learning new things made them a central figure on the team: "I focus a lot on teaching others my domain knowledge and testing, and it gets noticed. It's not that they panic when I'm not at work, but rather everyone works much better when I'm here. You contribute in a way to build everyone else up and not make yourself a bottleneck."

One tester stated that she learned the most through observation and practical experience. She felt that although the literature gave her an idea of how to do things, her knowledge became more concrete as she received hands-on experience during work. Another tester revealed during the interview that he wanted to begin conducting security testing but was unsure about where to start. His curiosity and persuasion led him back to becoming a student and taking a practical course in ethical hacking. The interviewee stated that the course helped greatly with deepening his technological knowledge.

The testers who were coaching juniors on the role emphasized that beginning testers needed to be eager to learn new things through practical experience if possible. They

explained that they would show the juniors how to perform a test and then encourage them to perform their own testing tasks through trial-and-error. For example, one tester (Informant 2) working in a mentor role explained: "I usually sit together with them and say, 'Okay, here are some test cases that I have written. I can walk you through the first one, feel free to ask any questions.' Then I'll ask them to write the next test case. Finally, I'll tell them to write some without my supervision." He also stated that this was a way to challenge themselves: "I'd ask them, 'Is there something you think could go wrong? Are there any other test cases you can think of? Should we talk to a domain expert if you're unsure?' It's a way to challenge them to a certain degree so that they are used to this way of working."

One developer (Informant 7) stated that an excellent tester should always be curious: "For example, if I say, 'We also have to test the APIs,' a good tester would admit that API testing isn't something familiar and would request half an hour to get more insight in it." The developer mentioned that it was always better for a tester to spend some time and come prepared to meetings instead of not knowing how to approach the information received from the developers.

4.7 Adaptable

The interviewees working in the testing role stated that their responsibilities and tasks changed rather frequently, and therefore, they needed to be flexible to respond quickly to change and adapt to changing work conditions. According to informant 1: "I think this is sort of the highlight of someone who is working with agile, the fact that one can quickly switch between things and the fact that one has sufficient control so that it does not take long to adapt to something different." When asked, those who had previously worked in sequential projects stated their preference for agile. One senior tester emphasized that working with agile improved the team's effectiveness, but they had to learn to switch between assignments quickly: "I work here because I like the way we work, and I want to continue doing that, so I need to adapt my skill set to the company and see what is needed."

Three of the interviewees working in a testing role mentioned that they had to adapt to follow a quickly developing work situation. Informant 1 explained: "It can sometimes be as early as after looking over my tasks and feeling ready to start, someone would pat me on the shoulder and say that I have to do something else." Another tester (Informant 5) stated: "We work with prioritization, when an item of high priority is incomplete from the developers' side, I'll start to work on something with medium priority. However, often, the developers finish before I get to complete the testing on that item, so I have to drop it and start testing on the one with higher priority."

One junior tester stated that since she was newly qualified, the constant context switching proved to be challenging to keep up with, along with keeping the same pace as the rest of the team. During our observation, she mentioned working extensively with a more experienced tester, and she stated this work approach had significantly improved her ability to switch contexts.

5 Discussion

In this section, we will first relate our findings to the results of the literature review, and then we will discuss our research question. We will end the discussion by giving some implications for practice and discussing the threats to the study's validity.

5.1 The software tester role and motivational factors

Our literature review discussed three themes: (1) software testing as a profession, (2) motivational factors for the software testers, and (3) the personal characteristics of software testers. The literature review results regarding software testing as a profession pointed to a rather negative perspective of the software tester role. For example, the findings of Capretz et al. (2015) indicated that among the people working in software development, tester and maintainer were the least popular roles. Similarly, Capretz et al. (2015) found that the vast majority of engineers in Canada, China, Cuba, and India perceive testing jobs as unattractive and would not choose the role of a tester. However, our study indicates that this discouraging perception might not be the case in agile projects. The interviewed practitioners working in software testing or in close collaboration with testers expressed full recognition of the merits of the software tester role. Both the testing and non-testing interviewees stated that testers were viewed on an equal level and with the same status as the others. This outlook improved the testers' motivation to perform in their work—an exciting find, considering the previous research, such as that of Shah and Harrold (2010), who reported that seniors considered testing to be boring, a stepping stone toward a career as a developer, and that testers did not get the recognition they deserved. Our results suggest that the agile software testers' colleagues showed deep respect for the role of the software testers; moreover, they identified traits by which the testers uniquely added value to software development.

Furthermore, our literature review showed that there are several factors that motivate software testers, such as having variety in their work, using creativity in solving tasks, and having well-defined work with a precise sequence of steps (Santos et al., 2017). Similarly, the factors that de-motivate software testers are time pressure, lack of clear processes, and working on test activities that are tedious, boring, or monotonous (Deak et al., 2016; Gonçalves et al., 2017). Our respondents were also motivated by being able to use creativity in their work, and they enjoyed structure and using checklists. Furthermore, the participants were also motivated by complex tasks and the fact that their job required communicating and working with many people in different roles.

The third group of papers in our literature review discussed personal characteristics of the software tester, which is closely related to our research question and will be discussed next.

5.2 Human factors of the agile software tester

There is a need to improve understanding of the part that software testers play in agile projects, and especially the human aspects (Sánchez-Gordón et al., 2020). Through the literature review, we found several common characteristics of software testers, such as being open-minded, curios, disciplined, and paying attention to details (Kanij et al., 2015; Livonen et al., 2010; Matturro, 2013; Deak, 2014b). Livonen et al. (2010) found that high-performing testers were thorough and patient. Studies of job advertisements have found that testers are required to be team players, proactive, and good communicators (Matturro, 2013; Florea & Stray, 2018). Our empirical investigation of agile software tester also found many of these characteristics. We will now discuss our empirical results in light of our research question: *Which human factors are essential for the agile software tester?*

Our findings suggest that testers need to see the whole picture. In other words, they need to know what the business side requires, whether the user journey is plausible,

and whether the technical modules work with each other as intended. Our findings support the work of Hernández and Marsden (2014), who pointed out that software testers work on a wide variety of topics because their role "requires a complete view of the software," as well as communication skills for effective collaboration with other departments. Testers who are able to take a wider view can bridge the gap between the domain and the technical part of the product, while also commenting for each respective part on what to keep in mind for the other parts during implementation. This has the potential to increase team transparency, making teamwork more fruitful while minimizing misunderstandings.

Most of our interviewees emphasized attention to detail as imperative for a tester to possess because it is useful in user experience, technical solutions, and enhancing domain knowledge and technical knowledge. This finding is in line with the survey research conducted by Kanij et al. (2014) in which most respondents agreed that this trait was something a good software tester should have. However, Capretz et al. (2019) found a set of demotivating factors in which the requirement of detail-oriented skill demotivated Cuban software testers. Our findings suggested that being attentive allows testers to catch mistakes earlier, with an increase in teamwork, effectiveness, and motivation.

It emerged from the interviews that fellow team members often called on agile software testers to provide support by clarifying requirements or providing additional information. This often leads to an increase in teamwork through transparency, efficiency, and constructiveness, suggesting that testers also must perform much social navigation to mitigate potential negative team dynamics. The agile manifesto makes communication a central part of software development. Therefore, it was not surprising that the interviewees stated that they spent half of their time communicating and that they viewed good communication skills as crucial. We also observed that the testers spent much time communicating with other project members, both face-to-face and on Slack. Earlier studies also highlighted this trait (Deak, 2014b; Florea & Stray, 2018). Ahmed et al. (2012) described software testers as "the software development team's worst enemy." Therefore, they need good relational skills. The testers in our study confirmed that they sometimes felt that way but that they had found ways to behave that would avoid any animosity, and they highlighted the value of being able to communicate in a diplomatic and friendly manner. As the software-testing role implies bringing unwanted news to the team, our findings confirm the need for good communication skills to avoid provoking conflicts. A recent study found that one of the key reasons why software practitioners do not want to take up a testing career is the worry that other project members will be upset when the testers report the results from assessing their work (Lizama et al., 2020).

Most testers stated that writing checklists and creating notes allowed them to free up their minds to focus on other tasks. Some of our interviewees indicated that working in an agile environment could be somewhat distracting. Kanij et al. (2015) utilized the Big Five taxonomy from psychology to highlight testers' personalities. Their findings show that testers generally have a higher level of conscientiousness than other roles—conscientiousness including orderliness, self-discipline, diligence, and dedication. Our results show that testers tend to be more organized, systematic, and structured—all of which can be subsumed under conscientiousness.

Although creativity is generally useful in software development, the degree of importance varies greatly among roles (Li et al., 2020). Regarding software testers, our study points to that creativity was perceived as necessary in two courses of action: to conduct software testing through various user personas in different scenarios and to come up with creative ways of testing the technicalities of the system. Our study confirms the results

obtained by Itkonen et al. (2013), who found that exploratory testing nurtured diverse, creative opportunities for testing.

Curiosity was an important trait of the agile software testers as it gave them a desire to investigate and an ability to gain both domain knowledge and technical knowledge, often at a fast pace. The testers were eager to learn new things actively and investigate on their own. Kanij et al. (2014) found that 70% of the 104 respondents ultimately agreed that intellectual curiosity was an important characteristic, implying that the trait was useful in enhancing the destructive mindset of a tester and help in deliberately trying to break a system. In a similar sense, Deak (2014b) found that participants considered being curious an "incentive for continuously improving the understanding of the product" as well as coming up with unusual testing scenarios. Li et al. (2020) argue that curiosity is an important trait to have in the field of software engineering because the field is constantly changing, and that curiosity is a motivating factor behind learning. Our findings support these findings, as the interviewees perceived curiosity to be an essential trait for software testers.

We found that in an agile environment, good software testers were those who could switch context fast. Having good knowledge of the domain as well as technical competencies greatly enhanced the ability to respond quickly to changes, such as changing, adding, or improving test cases. This ability can also foster effective communication with domain experts, product owners, or technical staff in order to shape the product to its highest quality. Ekwoge et al. (2017) mentioned that adaptability was needed for the use of new tools and techniques of testing. Our results suggest that although adaptability indeed benefits technical proficiency, the trait is more holistic in the sense that it also concerns domain knowledge and the ability to switch quickly between different contexts and different mindsets.

5.3 Implications for practice

Our findings on the important human factors of agile software testers can have several implications. Given the effort that software companies invest in recruiting and retaining the best-suited individuals for the roles, the first implication for the practice of our study concerns the employers in the software development industry. This paper's findings could benefit the industry by providing a set of relevant traits that software testers should possess, useful as a checklist to those in charge of hiring new testing personnel.

Additionally, our findings could benefit all professionals, whether with a technical background or not, who are considering careers as software testers by setting a frame of expectations for the role in soft factors. Garousi et al. (2020) recently reported that one of the top-ranked knowledge gaps for software engineers was software testing and argue that it is vital to teach more about testing. The second implication of our findings relates to the ISTQB syllabus (ISTQB, 2019) for the agile tester. The listed skills of an agile software tester should be updated with, for example, being able to view the whole picture and being adaptable.

The results also show a change in the perception of the role in the last decade, when Capretz et al. (2019) and Shah and Harrold (2010) found that people were choosing roles in software testing when they were technically proficient.

Another notable result is the usage of digital communication tools in agile teams. We noted that the use of Slack was perceived as positive by the testers and increased team transparency—supporting recent studies stating that the use of Slack makes communication more transparent in agile projects (Calefato et al., 2020).

Universities could include more human factors in courses on software testing. For example, instructors might teach techniques for viewing the whole picture while at the same time focusing on details. As the testers in our study also acted as a bridge between business and development, courses might include some curriculum based on agile coaching. It would also be a valuable addition for students to be able to practice through exercises how to communicate poor results. Soft skills are often seen by practitioners as more important than hard skills in new graduates, and there is therefore a need to introduce soft skills in the curriculum (Jiang & Klein, 2000).

Future work could investigate the traits that are important when specializing in test automation. Automation is used to improve the effectiveness of testing (Mahmud et al., 2014), reduce human errors (Rafi et al., 2012), and improve the reliability of testing, but it is not a replacement for human labor. To write, execute, and follow up on automated tests, one must have time and invest significant effort. Therefore, employers need to check whether the testers have the time and skills to develop and maintain a good automation process. This aspect is important, as Karlström et al. (2005) report that the main impediments to adopting software-test automation are of a managerial and not a technical nature.

5.4 Threats to validity

There are multiple threats when conducting qualitative research. For example, construct validity refers to how well a study measures its claimed construct and concerns whether the study has been affected by the researcher's subjective judgment (Yin, 2018). According to Yin, the use of multiple sources of evidence and "letting key informants review draft case study reports" are tactics that can increase the construct validity of a study. Since what people report is not always consistent with reality, triangulation may yield more reliable data than the use of only one data source (Robson & McCartan, 2016). We complemented the interviews with observations and analysis of Slack discussions, and we presented and discussed our results with the interviewees in the study.

Observing the employees working for several months might pose a risk that the interviewees in Company A were influenced by the researchers. However, we aimed to be neutral and believe that because the interviewees became familiar with the interviewer and the aim of the research, it made them share information more openly because they felt safe in the interview situation. Also, as suggested by Becker (2008), to reduce observer bias, we noted down as much information as possible when observing the project members interacting to increase the ability to remember details.

External validity, also known as generalizability, refers to whether the study's findings are generalizable beyond the contextual setting of the case study (Yin, 2018). For case studies, special attention is recommended for analytical generalization—that is, striving for generalizable findings and "going beyond the setting for the specific case or experiment that had been studied" (Yin, 2018). We interviewed people employed in three different companies. We also aimed to increase the external validity by interviewing three roles within the software development team and asking them to describe the relevant human factors affecting the role of the software tester. To avoid ambiguity, we allowed time for reflection on the questions. We were open to all input and insight, and we maintained contact with our respondents until the finalization of the study. We routinely checked the consistency of the data in the transcripts, the codes, and the themes we used.

6 Conclusion

Through a qualitative study of 22 software practitioners performed to capture their view of the relevant human factors for the testing role in agile projects, we brought to light that the software testers in agile settings were perceived as professionals with seven distinct traits: the ability to see the whole picture, an orientation toward detail, the possession of good communication skills, a strong sense of structure, creativity, curiosity, and adaptability.

Being able to see the whole picture is a valuable trait for agile testers; they must think of the quality of all aspects of the product encompassing technical and domain-specific abilities in order to conduct testing effectively. The agile testers spend much time communicating and need to do this in a timely and friendly manner. They had to give feedback in a constructive way to mitigate potential negative team dynamics. The use of agile methods increased the frequency of communication, particularly with customer representatives. An essential trait to possess is being detail-oriented. The testers enjoyed being attentive to details, catching things others did not see. The developers expressed a feeling of safety to have the detail-oriented testers on the team. Agile software testers have an advantage if they are creative. The testers in our study were creative in testing user perspectives such as personas, domain coverage, and non-standard scenarios, which made them discover abnormal bugs. Furthermore, they were seen as structured in testing and enjoyed creating and using checklists, and they were viewed as curious in ensuring the best agile product. In the final traits that we found, agile testers should be adaptable to changes and adept in switching contexts.

The findings from our study can be used in the industry, in particular by those with responsibility for recruiting software testers, helping to ensure that the candidates for the role exhibit these traits. Moreover, those considering a career in software testing should use our results to scrutinize and assess their fitness for a role as a tester in an agile environment.

Appendix A: Interview guide for the software tester

Part	Question
1. Introduction	Present ourselves
	Thank the participant for taking his/her time to be interviewed
	Inform the participant about privacy policy and voluntary participation • Data gathered from the interviews will be used to answer the research question • Any personal data will be treated confidentially • All participants will be anonymized • It is voluntary to participate in the interview, and the participant can at any time withdraw his/her consent
	Ask permission to record
	Estimate the length of the interview (45–60 min)
	What is your current role within the company?
2. Background and warm-up	How long have you been working as a software tester?
	How long have you worked at this company?
	Can you shortly explain the project you work in? How long have you worked on it?

Part	Question
	What kind of software development process do you use for the project?
	Describe your role in the team, with typical tasks and assignments. Give me some examples of your responsibilities
3. Role and tasks	What kind of testing do you like best? Why?
	When practicing your job, describe your knowledge in testing and beyond testing
	How do you discover a significant bug?
	Are your skills measured and followed up at work? How?
	Are some testing skills more complicated to acquire?
4. Team context	What can you tell me with regard to your connection with the team (roles, team members)?
	Who do you collaborate with the most? For how long?
	Do team members show interest in other individuals' tasks? How?
	How do you feel that software testing is integrated into the team's practices?
	What traits should a good software tester have? Why?
5. View on testers	What about a sub-optimal tester?
	Are any traits or skills more important for a tester to have?
6. Barriers/negative aspects	Have there been challenges for your role related to testing?
	Have there been challenges working with others in the team?
	What stresses you at work?
7. Closing	Do you have any questions for me?
	Is there anything else you would like to discuss that was not covered by the questions asked?

Acknowledgements We would like to thank all the participants for their generous and thoughtful collaboration on this study and for allowing us to observe and conduct interviews. We extend a special thanks to the companies for making the collaboration setup possible and to the Research Council of Norway for their support through grant 309344.

Funding Open access funding provided by University of Oslo (incl Oslo University Hospital).

References

Ahmed, F., Capretz, L. F., & Campbell, P. (2012). Evaluating the demand for soft skills in software development. *IT Professional, 14*(1), 44–49.

Becker, H. S. (2008). *Tricks of the trade: how to think about your research while you're doing it*. University of Chicago Press.

Bai, A., Mork, H. C., & Stray, V. (2017). A cost-benefit analysis of accessibility testing in agile software development: results from a multiple case study. *International Journal on Advances in Software*, 10(1).

Braun, V., & Clarke, V. (2006). Using thematic analysis in psychology. *Qualitative Research in Psychology, 3*(2), 77–101. https://doi.org/10.1191/1478088706qp063oa

Braun, V., Clarke, V., Hayfield, N., & Terry, G. (2019). Thematic analysis. In P. Liamputtong (Ed.), Handbook of Research Methods in Health Social Sciences (pp. 843–860). Singapore: Springer. https://doi.org/10.1007/978-981-10-5251-4_103

Byrd, T. A., & Turner, D. E. (2001). An exploratory analysis of the value of the skills of IT personnel: their relationship to IS infrastructure and competitive advantage. *Decision Sciences, 32*(1), 21–54.

Calefato, F., Giove, A., Losavio, M., & Lanubile, F. (2020). A case study on tool support for collaboration in agile development. IEEE/ACM International Conference on Global Software Engineering.

Capretz, L. F., Waychal, P., Jia, J., Varona, D., & Lizama, Y. (2019). Studies on the software testing profession. In 2019 IEEE/ACM 41st International Conference on Software Engineering: Companion Proceedings (ICSE-Companion) (pp. 262–263). https://doi.org/10.1109/ICSE-Companion.2019.00105

Capretz, L. F., Waychal, P., Jia, J., & Lizama, D. V. Y. (2015). Influence of personality types in software tasks choices. *Computers in Human Behavior, 52*, 373–378.

Capretz, Luiz Fernando, & Ahmed, F. (2010). Making sense of software development and personality types. IT Professional, 12(1).

Cavin, J. D. (2015). The role of human factors in veteran SQA training. *Procedia Manufacturing, 3*, 1535–1542. https://doi.org/10.1016/j.promfg.2015.07.416

Cohn, M., & Ford, D. (2003). Introducing an agile process to an organization [software development]. *Computer, 36*(6), 74–78.

Davidov, M. B., Kuchkova, I. N., Barilov, A. A., & Pastsyak, A. R. (2010). Method and system for analyzing software test results. Google Patents.

Deak, A. (2014a). A comparative study of testers' motivation in traditional and agile software development. In A. Jedlitschka, P. Kuvaja, M. Kuhrmann, T. Männistö, J. Münch, & M. Raatikainen (Eds.), Product-focused software process improvement (pp. 1–16). Springer International Publishing.

Deak, A. (2014). What characterizes a good software tester?—a survey in four Norwegian companies. In M. G. Merayo & E. M. de Oca (Eds.), *Testing Software and Systems* (pp. 161–172). Springer.

Deak, A., Stålhane, T., & Cruzes, D. (2013). Factors influencing the choice of a career in software testing among Norwegian students. Software Engineering, 796.

Deak, A., Stålhane, T., & Sindre, G. (2016). Challenges and strategies for motivating software testing personnel. *Information and Software Technology, 73*, 1–15. https://doi.org/10.1016/j.infsof.2016.01.002

Dhir, S., & Kumar, D. (2019). Automation software testing on web-based application. In M. N. Hoda, N. Chauhan, S. M. K. Quadri, & P. R. Srivastava (Eds.), Software Engineering (pp. 691–698). Springer Singapore.

Dingsøyr, T., Falessi, D., & Power, K. (2019). Agile development at scale: the next frontier. IEEE Software, 36(2). https://doi.org/10.1109/MS.2018.2884884

Ebert, C., & De Neve, P. (2001). *Surviving Global Software Development. IEEE Software, 18*(2), 62–69.

Ekwoge, O. M., Fontão, A., & Dias-Neto, A. C. (2017). Tester experience: concept, issues and definition. In 2017 IEEE 41st Annual Computer Software and Applications Conference (Vol. 1, pp. 208–213). https://doi.org/10.1109/COMPSAC.2017.232

Faraj, S., & Sproull, L. (2000). Coordinating expertise in software development teams. *Management Science, 46*(12), 1554–1568.

Florea, R., & Stray, V. (2018). Software tester, we want to hire you! An analysis of the demand for soft skills. In Garbajosa, J., Wang, X., & Aguiar, A.(Eds.), Agile Processes in Software Engineering and Extreme Programming, 54–67. Springer International Publishing.

Florea, R., & Stray, V. (2020). A qualitative study of the background, skill acquisition, and learning preferences of software testers. In Proceedings of the Evaluation and Assessment in Software Engineering, 299–305.

Garousi, V., Giray, G., Tuzun, E., Catal, C., & Felderer, M. (2020). Closing the gap between software engineering education and industrial needs. *IEEE Software, 37*(2), 68–77. https://doi.org/10.1109/MS.2018.2880823

Ghobadi, S., & Mathiassen, L. (2016). Perceived barriers to effective knowledge sharing in agile software teams. *Information Systems Journal, 26*(2), 95–125.

Gonçalves, W. F., de Almeida, C. B., de Araújo, L. L., Ferraz, M. S., Xandú, R. B., & de Farias, I. (2017). The influence of human factors on the software testing process: The impact of these factors on the software testing process. In *2017 12th Iberian conference on information systems and technologies (CISTI)* (pp. 1-6). IEEE. https://doi.org/10.23919/CISTI.2017.7975873

Hernández y, T. P. R., & Marsden, N. (2014). Understanding software testers in the automotive industry a mixed-method case study. In *2014 9th International Conference on Software Engineering and Applications (ICSOFT-EA)* (pp. 305-314). IEEE.

ISTQB, F. (2019). Foundation Level Syllabus version 2018. International Software Testing Qualifications Board.

Itkonen, J., Mäntylä, M. V., & Lassenius, C. (2013). The role of the tester's knowledge in exploratory software testing. *IEEE Transactions on Software Engineering, 39*(5), 707–724. https://doi.org/10.1109/TSE.2012.55

Jiang, J., & Klein, G. (2000). Software development risks to project effectiveness. *Journal of Systems and Software, 52*(1), 3–10.

Juristo, N., Moreno, A. M., & Strigel, W. (2006). Guest editors' introduction: software testing practices in industry. *IEEE Software, 23*(4), 19–21.

Kanij, T., Merkel, R., & Grundy, J. (2014). A preliminary survey of factors affecting software testers. In *2014 23rd Australian Software Engineering Conference* (pp. 180-189). IEEE. https://doi.org/10.1109/ASWEC.2014.32

Kanij, T., Merkel, R., & Grundy, J. (2015). An empirical investigation of personality traits of software testers. In *2015 IEEE/ACM 8th International Workshop on Cooperative and Human Aspects of Software Engineering* (pp. 1-7). IEEE. https://doi.org/10.1109/CHASE.2015.7

Karlström, D., Runeson, P., & Nordén, S. (2005). A minimal test practice framework for emerging software organizations. *Software Testing, Verification and Reliability, 15*(3), 145–166.

Kettunen, V., Kasurinen, J., Taipale, O., & Smolander, K. (2010). A study on agility and testing processes in software organizations. In *Proceedings of the 19th international symposium on Software testing and analysis* (pp. 231-240). https://doi.org/10.1145/1831708.1831737

Kitchenham, B. A. & Charters, S. (2007). *Guidelines for performing Systematic Literature Reviews in Software Engineering* (EBSE 2007-001). Keele University and Durham University Joint Report .

Korhonen, K. (2013). Evaluating the impact of an agile transformation: a longitudinal case study in a distributed context. *Software Quality Journal, 21*(4), 599–624. https://doi.org/10.1007/s11219-012-9189-4

Lethbridge, T. C. (2000). What knowledge is important to a software professional? *Computer, 33*(5), 44–50.

Li, J., Moe, N. B., & Dybå, T. (2010). Transition from a plan-driven process to scrum: a longitudinal case study on software quality. In *Proceedings of the 2010 ACM-IEEE international symposium on empirical software engineering and measurement* (pp. 1-10). https://doi.org/10.1145/1852786.1852804

Li, P. L., Ko, A. J., & Begel, A. (2020). What distinguishes great software engineers? *Empirical Software Engineering, 25*(1), 322–352. https://doi.org/10.1007/s10664-019-09773-y

Lindstrom, L., & Jeffries, R. (2004). Extreme programming and agile software development methodologies. *Information Systems Management, 21*(3), 41–52.

Livonen, J., Mäntylä, M. V., & Itkonen, J. (2010). Characteristics of high performing testers: a case study. In *Proceedings of the 2010 ACM-IEEE international symposium on empirical software engineering and measurement* (pp. 1-1), ESEM 2010. Bolzano, Italy. https://www.researchgate.net/publication/221494743_Characteristics_of_high_performing_testers_a_case_study. Accessed 10 September 2019

Lizama, Y., Varona, D., Waychal, P., & Capretz, L. F. (2020). The Unpopularity of the Software Tester Role Among Software Practitioners: A Case Study. In *Advances in RAMS Engineering* (pp. 185-197). Springer, Cham.

Mahmud, J., Cypher, A., Haber, E., & Lau, T. (2014). Design and industrial evaluation of a tool supporting semi-automated website testing. *Software Testing, Verification and Reliability, 24*(1), 61–82.

Mathur, S., & Malik, S. (2010). Advancements in the V-Model. *International Journal of Computer Applications*, *1*(12), 29-34.

Matturro, G. (2013). Soft skills in software engineering: A study of its demand by software companies in Uruguay. In *2013 6th international workshop on cooperative and human aspects of software engineering (CHASE)* (pp. 133-136). IEEE. https://doi.org/10.1109/CHASE.2013.6614749

Novet, J. (2021). Slack touts user growth as it faces growing competition from Microsoft. *CNBC*. https://www.cnbc.com/2019/10/10/slack-says-it-crossed-12-million-daily-active-users.html

Patton, M. Q. (2014).*Qualitative evaluation and research methods: Integrating theory and practice*. Sage Publications.

Paruch, L., Stray, V., & Blindheim, C. B. (2020a). Characteristic Traits of Software Testers. In *Proceedings of the Evaluation and Assessment in Software Engineering* (pp. 371-372).

Paruch, L., Stray, V., & Florea, R. (2020b). The Human Factors of the Agile Software Tester. In *International Conference on the Quality of Information and Communications Technology* (pp. 474-487). Springer, Cham.

Rafi, D. M., Moses, K. R. K., Petersen, K., & Mäntylä, M. V. (2012). Benefits and limitations of automated software testing: Systematic literature review and practitioner survey. In *2012 7th International Workshop on Automation of Software Test (AST)* (pp. 36-42). IEEE.

Robson, C., & McCartan K. (2016). Real world research. John Wiley & Sons. ISBN: 978-1-118-74523-6

Runeson, P., Host, M., Rainer, A., & Regnell, B. (2012). *Case study research in software engineering: guidelines and examples*. John Wiley & Sons.

Saiedian, H., & Dale, R. (2000). Requirements engineering: making the connection between the software developer and customer. *Information and Software Technology, 42*(6), 419–428. https://doi.org/10.1016/S0950-5849(99)00101-9

Salman, I., Turhan, B., & Vegas, S. (2019). A controlled experiment on time pressure and confirmation bias in functional software testing. *Empirical Software Engineering, 24*(4), 1727–1761. https://doi.org/10.1007/s10664-018-9668-8

Sánchez-Gordón, M., Rijal, L., & Colomo-Palacios, R. (2020). Beyond Technical Skills in Software Testing: Automated versus Manual Testing. In *Proceedings of the IEEE/ACM 42nd International Conference on Software Engineering Workshops* (pp. 161-164).

Santos dos, A. M., Karlsson, B. F., Cavalcante, A. M., Correia, I. B., & Silva, E. (2011). Testing in an agile product development environment: An industry experience report. In *2011 12th Latin American Test Workshop (LATW)* (pp. 1-6). IEEE. https://doi.org/10.1109/LATW.2011.5985897

Souza Santos de, R. E., de Magalhães, C. V. C., da Silva Correia-Neto, J., da Silva, F. Q. B., Capretz, L. F., & Souza, R. (2017). Would you like to motivate software testers? Ask them how. In *2017 ACM/IEEE International Symposium on Empirical Software Engineering and Measurement (ESEM)* (pp. 95-104). IEEE. https://doi.org/10.1109/ESEM.2017.16

Shah, H., & Harrold, M. J. (2010). Studying human and social aspects of testing in a service-based software company: case study. In *Proceedings of the 2010 ICSE Workshop on Cooperative and Human Aspects of software Engineering*(pp. 102-108). New York, NY, USA: ACM. https://doi.org/10.1145/1833310.1833327

Stray, V., & Moe, N. B. (2020). Understanding coordination in global software engineering: a mixed-methods study on the use of meetings and Slack. *Journal of Systems and Software, 170,* 110717. https://doi.org/10.1016/j.jss.2020.110717.

Yang, C., Liang, P., & Avgeriou, P. (2016). A systematic mapping study on the combination of software architecture and agile development. *Journal of Systems and Software, 111*, 157–184. https://doi.org/10.1016/j.jss.2015.09.028

Yin, R. K. (2018). *Case study research and applications: design and methods* (6th ed.). SAGE publications.

Viktoria Stray is an associate professor at the University of Oslo's Department of Informatics. Her research interests include agile methods, global software engineering, software testing, coordination, and large-scale development. Her main focus is to improve the productivity of testers and developers and increase the success of software projects. Stray has a Ph.D. in software engineering and has worked several years in the industry participating in some of the largest software development projects in Norway. She also holds a research position at SINTEF.

Raluca Florea is a Ph.D. candidate at the University of Oslo. She has 15 years' experience in the field of software development, in the industries of telecommunication, finance, logistics, and learning platforms. She worked as a software tester, test manager, automation engineer, project manager,and team leader. Raluca is a member of the Norwegian Testing Board and the International Software Testing Qualifications Board, co-authoring syllabi on the fundamentals of software testing, test management, and test automation.

Lucas Paruch Lucas Paruch received his M.Sc degree in Computer Science from University of Oslo in May 2020, along with two published papers. He has several years of experience working as a teaching assistant and examiner for the software testing course held at the university. Lucas is currently working as a software tester in Itera, a Norwegian consultancy company.