An initial investigation of the effect of quality factors on Agile test case quality through experts' review

Samera Obaid Barraood1,²*, Haslina Mohd² and Fauziah Baharom²

Abstract: A test case is a cornerstone of the testing process; hence, it is quintessential to ensure the quality of the test cases. However, test case design in the Agile testing process has limitations that affect its quality standards, leading to the failure of many software projects. Previous studies are limited in providing a clear guideline for assessing the test case quality. However, evaluating the quality of test cases will help software testing practitioners to understand some critical issues in designing test cases from various perspectives. Therefore, this paper aims to present the verified factors and criteria of test case quality by software testing experts in an Agile environment. The proposed factors and criteria were verified through expert review techniques, which included an online survey of professionals and those in domain and knowledge experts. Twenty-three industry practitioners, including developers, testers, and systems analysts, were used to identify the domain experts. In contrast, the knowledge experts were drawn from 13 academic areas of software engineering and testing. The results showed that the quality of test cases is paramount in Agile projects; therefore, the experts accept seven factors and 32 criteria with a few modifications and suggestions. The finding may assist the researchers in improving the proposed factors and criteria for validation purposes. Hence, may contribute to software testing practitioners as a guideline in constructing, designing, and assessing compelling test cases. Most significantly, the fixed factors and criteria may contribute to the body of knowledge in the software testing domain and industries.

Subjects: Software Engineering & Systems Development; Systems Integration; Computer Science (General); Technology

Keywords: Software testing; test case quality; quality factors; quality criteria; Agile software development

1. Introduction
Software testing is a quality assurance activity and an important component of any project (Nawaz & Malik, 2008). In an Agile context, a testing process has more value for demonstrating quality products (Harichandan et al., 2014). Brian Marick provides a philosophy of Agile testing as “a style of testing, one with lessened reliance on documentation, increased acceptance of change, and the notion that a project is an ongoing conversation about quality” (Leffingwell, 2010). However, continuous changes adopted in Agile methods require much effort on testing activities (Beck, 2003; Gay et al., 2016; Humble & Farley, 2010). The efficiency of testing activities depends mainly on the test case quality (TCO), which directly defines the quality of testing (Causevic, Punnekat et al., 2012; Kumar & Ranjan, 2017; Lai, 2017). A good-quality test case is a test case that has high chances to expose defects in
a minimum effort, produce more accurate results, increase the performance of the system at a lower cost, and has a high probability of discovering unknown defects (i.e. the higher the quality of a test case, the more the capacity to reveal failures; Gómez et al., 2016; Kamde et al., 2006; Mohi-aldeen et al., 2014; Pressman, 2010; Yadav & Yadav, 2015). However, some issues, such as those detailed in Barraood et al. (2021c), are capable of causing reductions in the quality of test cases. Some of these issues are unspecified test target or expected results (Jovanovikj et al., 2018), incomplete, incorrect, and inconsistent software requirements (Kärkila & Pirta, 2018), and traceability between test cases and related requirements are weak (Fischbach et al., 2020). Therefore, it is essential to evaluate and measure the quality of test cases, to understand how much testing is needed and where future test efforts should be applied (Ahmed et al., 2016).

Researchers in Agile methods mainly focused on the quality of the code (Causevic, Punnekkat et al., 2012), with only a few studies that give attention to the quality of test cases. Hence, this paper aims to provide the quality factors and criteria that may assist the Agile team in designing quality test cases, which lead to producing a high-quality testing process in building a good-quality product. The factors and criteria were extracted from the existing empirical studies in TCQ. This study conducted an exploratory survey to investigate the identified test cases’ factors and criteria in an Agile environment. An online survey of academic and industry experts was conducted to verify the factors and criteria. Thirty-six experts, 13 academias, and 23 practitioners in Agile organizations participated in the online survey to give their opinions. The study produces 32 criteria of TCQ grouped into seven factors: documentation quality, management quality, maintainability, reusability, requirement quality, performance efficiency, and test case effectiveness.

The rest of the paper is organized as follows. Section 2 introduces the related work. Section 3 presents the methodology, Section 4 summarizes the results and discussions, and Section 5 concludes the paper.

2. Related work

Significant efforts have been deployed by researchers towards building test cases with quality, as can be seen in (Adlemo et al., 2018; Athanasiou et al., 2014; Bowes et al., 2017; Causevic, Sundmark et al., 2012; Daka & Fraser, 2014; Jovanovikj et al., 2018; Juhnke et al., 2021, 2018; Kamde et al., 2006; Kaner, 2003; Kochhar et al., 2019; Lai, 2017; Tran et al., 2019). Kamde et al. (2006) designed a standard checklist to determine the risk areas and improve the test cases. The study provides a set of quality factors of test cases, including correctness, accuracy, economical, reliability and repeatable, traceable, and measurable. Causevic, Punnekkat et al. (2012) used three attributes of TCQ, code coverage, mutation score, and the total number of failing assertions, as metrics for measuring quality. Athanasiou et al. (2014) built a model that combines metrics to define a test code quality measure. This model combines eight metrics for three quality factors, completeness, effectiveness, and maintainability, for assessing the test code quality. However, the metrics of the aforementioned studies are most applicable to automated test cases, and the model can only be applied to tests implemented in some programming language (Horváth et al., 2015).

Bowes et al. (2017) tried to answer the question of “How good are my tests?” by presenting a list of 15 testing principles that capture the essence of testing goals and can be quantified as indicators of test quality. The authors highlight the importance of tests being correct, maintainable, and simple in their findings. They argue that test cases should be simple and meaningful to improve readability, especially in Agile. However, this study focused on smell tests in unit testing and some of their criteria were not implemented and others were without metrics. Lai (2017) proposed a test case quality measurement (TCQM) model for improving TCQ, which ensures the efficiency of continuous testing activities. Four quality factors were identified for TCQ which are documentation, maintainability, manageability, and reusability. These factors include 13 criteria.
Although these studies (Kamde et al., 2006; Lai, 2017) proposed factors of TCQ, the factors definitions are, however, not dependent on the practitioners’ goals and opinions.

The recent studies that tried to identify the factors of TCQ based on the practitioner perspective are conducted by (Adilemo et al., 2018; Jovanovikj et al., 2018; Kochhar et al., 2019; Tran et al., 2019). Juhnke et al. (2018) conducted a study related to the quality model of test case specifications. Juhnke identified seven quality indicators of informal test case specifications in the automotive domain from 816 test case specifications as a case study specified by an OEM and suppliers. A test case specification contains a set of test cases necessary to adequately test a particular test object according to defined test objectives. They have conducted another study in Juhnke et al. (2021), which identified nine groups of test case specification challenges. The study lists comprehensibility, unambiguity, completeness, uniformity, atomicity, and suitability for the respective test platform as the factors of test case specification quality. Their model focuses on automotive-specific test case specifications; thus, the model may not be suitable for any other domains. Jovanovikj et al. (2018) proposed an approach called TCQP, a quality plan for evaluating the usability of test cases with appropriate tool support. They proposed four quality factors (i.e., usability, maintainability, reusability, and test effectiveness) with four quality criteria (understandability, analysability, comprehensibility, and fault-revealing capability). They defined metrics by using the GQM approach. However, other quality factors and criteria for the test cases are conspicuously missing in their study. Kochhar et al. (2019) provided 29 attributes from practitioners’ reviews related to the sound test cases which they grouped into six factors: content, size, complexity, coverage, maintainability, and bug detection.

Similarly, Tran et al. (2019) conducted a study on TCQ from professional developers and testers perspectives. The study identified understandability, simplicity, step cohesion, completeness, homogeneity, issue-identifying capability, repeatability, traceability, effectiveness, efficiency, and flexibility as quality factors for manual written test cases. Tran et al. improved their work in (Tran et al., 2021) by developing a quality model for test artefacts composed of 30 attributes categorized into nine with 20 attributes features from the ISO/IEC 25010 standard. This model also includes measurements for the attributes. The identifiable drawbacks of this model are as follows: 12 out of 18 measures are without descriptions, 19 attributes have no information, the 20 attributes from ISO need further investigation, and this model needs to be validated. Adilemo et al. (2018) used some criteria from (Kaner, 2003) and (Atmadja & Shuai, 2011) for identifying and ranking the TCQ factors. Adilemo et al. (2018) identified and ranked about 15 criteria for quality test cases based on testing experts in Sweden. These criteria include repeatable, accurate, correct, powerful, maintainable, complete, traceable, consistent, reusable, simple, efficient, clear, independent, covering, and compact. However, these studies are limited to providing guidelines for assessing the quality of test cases.

Additionally, they are mostly conducted based on a traditional, non-Agile approach. Among the very few studies related to test cases in Agile software development is Ahmad et al. (2019), which reported that requirements should be unambiguous to ensure the good quality of test cases. In another study, Shrivastava and Jain (2010) presented an automated test case for unit testing (ATCUT) design metrics explicitly focusing on the testability of test cases. The most related study to our approach was conducted by Lai (2017); however, the study did not depend on practitioners’ goals to define the factors and criteria. In addition, the factors and criteria are minimal, and some critical criteria are missing in his model.

A comprehensive literature review was conducted to identify the TCQ factors and related criteria. About 14 journal articles based on SLR (Barraood et al., 2021b, 2018) and other journal articles were reviewed, and 22 established software testing websites were investigated (for more details, refer to (Barraood et al., 2021a)). The finding from the literature reviewed and software testing website led to identifying seven TCQ factors, documentation quality, management quality, maintainability,
reusability, requirement quality, performance efficiency, and effectiveness of test case, and 32 related criteria. Table 1 illustrates the identified criteria grouped into seven factors and their resources.

Seven factors and 32 criteria are identified from the literature review for the quality of the test cases model in Agile. However, the domain and knowledge experts need to verify these factors and criteria. The validation of the proposed model will be performed after getting feedback from the experts, and the proposed factors will be amended based on the feedback from experts. Hence, all the proposed factors and related criteria shall be accepted based on the experts’ feedback. The verification procedures are discussed in the next section.

3. Research methodology
The exploratory study was part of the research work as a pre-evaluation stage. In this study, an online survey was used to get opinions and pre-verification from the experts on the identified quality factors and criteria in the Agile environment. The study adopted judgment sampling, which is a kind of purposive sampling method to select the experts who responded to the instruments. Figure 1 illustrates the steps of the research methodology in this paper. As mentioned earlier, these quality factors and criteria are identified from literature reviews [Barraood et al., 2021b, 2018] and established software testing websites [Barraood et al., 2021a]. The respondents were asked to provide their rating on the proposed criteria based on the 5-Likert scale provided in the instrument.

**Subjects:** The two categories of subjects for this exploratory study are experts from industry and academics. The industry experts were obtained through contacts on social media (Facebook, LinkedIn, YouTube, and WhatsApp) and by contacting friends working in the software industry from different countries such as Malaysia, Yemen, India, and Jordan to get many of respondents in Agile area. These experts were selected based on the criteria that they are i) Agile Testing practitioners and ii) have experience in Agile Testing and quality assurance for more than 3 years. This is as adapted from Mohamed (2015) and Tran et al. (2019). Academic respondents were contacted through emails based on the following criteria, as suggested by Hallowell and Gambatese (2009), Rogers and Lopez (2002), Mohamed (2015), and Rajaram et al. (2021): i) currently lecturing in the field of the study, ii) holds an advanced degree such as PhD in Software Engineering or software testing, iii) faculty members at an accredited university, iv) have authored book/academics materials related to the software testing, and v) have at least 5 years of experience in software testing. The selected responded from academics category are experts in software engineering from six Malaysian universities (UUM, UKM, USM, UM, UTM, and UPM) and three Nigerian universities (ABU, BUK, and SSU). They were introduced by senior faculty advisors and friends based on their excellent reputation in the software testing domain.

The experts were asked to indicate their willingness to participate in the survey after they were informed about the purpose of the study. The verification forms were then sent to the experts who accepted the invitation emails, and they were asked to spread them to their colleagues and friends in the related field. The thankful notification was sent to the experts who apologized for their inability to participate in the exercise.

**A survey:** An online survey was created using the Google form, and a link was sent to the corresponding respondents who agreed to participate in the survey. The instrument consists of three parts, but in this paper, we described only the part related to the importance of test case quality and the part for verifying the quality factors and criteria of test cases. The respondents were asked to provide their rating on the proposed criteria. The scale ranged from Strongly Disagree to Strongly Agree on a Likert scale from 1 to 5. The instrument consists of 32 criteria grouped into seven factors: documentation quality, management quality, maintainability, reusability, requirement quality, performance efficiency, and effectiveness of test case, which are to be rated by the respondents. Also, the respondents were asked to provide their comments or
| Quality Factors            | Criteria                      | Resources                                                                 |
|----------------------------|-------------------------------|---------------------------------------------------------------------------|
| Documentation quality      | Uniform format                | (Adlemo et al., 2018; Lai, 2017; Tran et al., 2019)                       |
|                            | Correctness                   | (Adlemo et al., 2018; Bowes et al., 2017; Kamde et al., 2006; Lai, 2017; Meszaros et al., 2003; Tran et al., 2019) |
|                            | Completeness                  | (Adlemo et al., 2018; Bowes et al., 2017; Kochhar et al., 2019; Lai, 2017; Meszaros et al., 2003; Tran et al., 2019) |
|                            | Consistency                   | (Adlemo et al., 2018; Bowes et al., 2017; Lai, 2017; Tran et al., 2019)  |
|                            | Readability                   | (Adlemo et al., 2018; Bowes et al., 2017; Kaner, 2003; Kochhar et al., 2019; Meszaros et al., 2003) |
|                            | Understandability             | (Adlemo et al., 2018; Bowes et al., 2017; Jovanovikj et al., 2018; Kamde et al., 2006; Kaner, 2003; Kochhar et al., 2019; Shrivastava & Jain, 2010; Tran et al., 2019) |
|                            | Independent                   | (Adlemo et al., 2018; Bowes et al., 2017; Kamde et al., 2006; Kochhar et al., 2019; Lai, 2017; Meszaros et al., 2003; Quadri & Farooq, 2010; Shrivastava & Jain, 2010) |
|                            | Specific                      | (Bowes et al., 2017; Chauhan, 2010; Kochhar et al., 2019; Meszaros et al., 2003) |
| Management quality         | Retrieval quality             | (Lai, 2017)                                                              |
|                            | Recoverability                | (ISO/IEC25010, ISO/IEC25010, 2011; Lai, 2017; Tran et al., 2021)          |
|                            | Version control mechanism     | (Lai, 2017)                                                              |
|                            | Prioritization and Organizability | (Berberyen & Ali, 2019)                                            |
| Maintainability            |                               |                                                                          |
|                            |                               |                                                                          |
| Quality Factors | Criteria | Resources |
|----------------|----------|-----------|
| Simplicity     | (Adlemo et al., 2018; Bowes et al., 2017; Kamde et al., 2006; Kochhar et al., 2019; Lai, 2017; Meszaros et al., 2003; Tran et al., 2019) |
| Changeability | (Bowes et al., 2017; ISO/IEC 2000; Juan et al., 2010; Kochhar et al., 2019; Lai, 2017; Meszaros et al., 2003; Tran et al., 2019) |
| Traceability   | (Adlemo et al., 2018; Ali, 2006; Bowes et al., 2017; Kamde et al., 2006; Kochhar et al., 2019; Lai, 2017; Meszaros et al., 2003; Tran et al., 2021; Tran et al., 2019) |
| Peer Review    | (Quadri & Farooq, 2010) |
| Update Regularly | (Adlemo et al., 2018; Collins et al., 2012; Grano et al., 2018; Li et al., 2016; Quadri & Farooq, 2010; Tran et al., 2019) |
| Reusability    | (ISO/IEC25023, ISO/IEC25023, 2016; Lai, 2017; Tran et al., 2021) |
| Automaticity   | (Lai, 2017; Meszaros et al., 2003) |
| Extensibility  | (Bowes et al., 2017; Kochhar et al., 2019; Lai, 2017; Meszaros et al., 2003; Tran et al., 2019) |
| Combinability  | (Lai, 2017) |
| Simplicity     | (Adlemo et al., 2018; Athanasiou et al., 2014; Bowes et al., 2017; Kamde et al., 2006; Kochhar et al., 2019; Lai, 2017; Shrivastava & Jain, 2010; Tran et al., 2019) |
| Repeatable     | (Adlemo et al., 2018; Bowes et al., 2017; Kaner, 2003; Meszaros et al., 2003; Tran et al., 2021; Zeiss et al., 2007) |
| Requirement quality | (Craig & Jaskiel, 2002; Garousi et al., 2017; Kochhar et al., 2019; Pföller et al., 2008; Quadri & Farooq, 2010) |
| Specific       | (Mannion & Keepence, 1995; Quadri & Farooq, 2010; Schedlbauer, 2012) |
| Measurable     | (Mannion & Keepence, 1995) |
| Attainable     | (Mannion & Keepence, 1995) |
| Quality Factors                  | Criteria   | Resources                                                                 |
|---------------------------------|------------|---------------------------------------------------------------------------|
|                                  | Realisable | (Mannion & Keepence, 1995)                                                |
|                                  | Traceable  | (Bucaioni et al., 2018; Mannion & Keepence, 1995)                         |
| **Performance Efficiency**      |            | (Adlemo et al., 2018; Jovanovikj et al., 2018; Kaner, 2003; Kochhar et al., 2019; Meszaros et al., 2003; Tran et al., 2021, Tran et al., 2019) |
| Bug detection                   |            | (Adlemo et al., 2018; Kaner, 2003; Kayes et al., 2016; Kochhar et al., 2019) |
| Resource Utilization            |            | (ISO/IEC25023, 2016; Tran et al., 2021)                                   |
| Time behaviour                  |            | (ISO/IEC25023, 2016; Tran et al., 2021)                                   |
| **Effectiveness**               |            | (Adlemo et al., 2018; Athanasiau et al., 2014; Jovanovikj et al., 2018; Kaner, 2003; Tran et al., 2019) |
| Coverage                        |            | (Adlemo et al., 2018; Athanasiau et al., 2014; Causevic, Punnekkat et al., 2012; Kochhar et al., 2019; Tran et al., 2019) |
| Accuracy                        |            | (Adlemo et al., 2018; Atmadja & Shuai, 2011; Kamde et al., 2006; Kaner, 2003; Quadri & Farooq, 2010) |
suggestions on the proposed criteria and factors in case they wanted to add more criteria or edit the identified criteria.

The instrument was validated through face validity by three experts in software engineering and testing to ensure that the items were comprehensive and appropriate to the targeted construct and assessment objectives. Additionally, the instrument’s structure needs to be friendly, with clear instructions, not too lengthy, and written in a well-understandable language. This number of experts is appropriate based on the assertion by (Lynn, 1986; Waltz et al., 2010) that three to five experts can adequately face validate an instrument. The instruments were refined based on the experts’ suggestions before the actual administration to the target experts.

**Data Collection:** The instruments were sent to many experts, but only 41 were returned. Data collection took 2 months more in contrast to a planned 1-month schedule for the respondents to answer the instruments. Reminders were sent to the respondents who failed to return the instruments at the expiration of the given time. Five instruments with incomplete answers were rejected because the respondents’ companies do not follow Agile methodology, so they could not complete the instrument. Therefore, the valid instruments were 36 (13 academicians and 23 practitioners). These 36 instruments were complete without blank answers (no missing data). The professional experience of the respondents varies from 1 year to more than 10 years. The majority (52.7%) have between 1 year and 5 years of experience, and 33.3% have more than 5 years and less than 11 years of experience. This data was collected in 2 months, between 3 May and 2 July 2020. Several reminders were sent to those who failed to respond until the last day. Table 2 shows the overview of respondents. For the model validation stage,
the data set will be based on the testing reports that use IEEE Standard for Software Test Documentation (IEEE 829, 2008), followed by the interview session with the software testers regarding the process flows of designing constructing the test cases. The validation process will be discussed in the next journal article.

Data analysis: The collected data were analyzed using descriptive statistics to describe the opinion of respondents about the factors and criteria of the test case in Agile software development. Their ratings are converted to a 5-Likert score, which is mapped as strongly agree, agree, neutral, disagree, and strongly disagree to 5 to 1, respectively. Frequency, mean, and cross-tabulation were used for analysis by the SPSS tool version 26. Descriptive analysis was used to compute the mean of each criterion and factor. By the mean value, it can be known which criterion and factor are more suitable to measure good TCQ among the respondents. To support or reject the proposed criteria and factors, the respondents’ comments and suggestions were also sorted, which the authors summarized.

4. Results and discussion
In this section, the feedback of the experts on the importance of test case quality in Agile Software Development (ASD) and the proposed TCQ factors and criteria are presented.

(A) Importance of test case quality:
The respondents were asked to provide their opinion on the importance of good-quality test cases in ASD. Six questions were given to gain their views on that. The results showed that all respondents agreed that the test case quality in ASD is critical to 1) improve the software quality, 2) meet customers’ needs, 3) improve the adequacy of testing, 4) reduce defects and bugs, 5) decrease time of delivery, and 6) reduce the cost of development, as shown in Table 3. However, the benefit of TCQ to improve software quality has the highest value (4.69) among others. This is followed by (4.50) to show that TCQ is very important to meet customers’ needs.

(1) Software Quality Improvement: Software product quality depends on the testing quality (Gupta et al., 2018; Kayes et al., 2016). The testing process has more value for demonstrating the quality product in an Agile environment (Harichandan et al., 2014). Testing can detect that the software has faults and estimate its likely overall quality (Ahmed et al., 2016). Software quality improves when these faults are corrected (Ahmed et al., 2016). The test case is a significant asset in software testing activities (Lai, 2017). A good-quality test case has a high chance to reveal defects (Gómez et al., 2016; Kamde et al., 2006; Mohi-aldeen et al., 2014; Pressman,
Barrood et al., Cogent Engineering (2022), 9: 2082121
https://doi.org/10.1080/23311916.2022.2082121

| Table 3. Test case quality importance of ASD |
|---------------------------------------------|
|                                           | N  | Minimum | Maximum | Mean |
| Good TCQ improve software quality          | 36 | 2       | 5       | 4.69 |
| Good TCQ decrease the time of delivery     | 36 | 2       | 5       | 3.97 |
| Good TCQ reduce the development costs      | 36 | 1       | 5       | 3.81 |
| Good TCQ help to meet customers’ needs     | 36 | 2       | 5       | 4.50 |
| Good TCQ improve the adequacy of testing   | 36 | 2       | 5       | 4.33 |
| Good TCQ reduce defects/bugs               | 36 | 1       | 5       | 4.28 |

2010; Yadav & Yadav, 2015). Therefore, this study intended to investigate the influence of test case quality on increasing software quality improvement. The result showed that most respondents emphasized that good quality of test cases can improve the quality of software products, as in Table 2.

(2) Decreasing Delivery Time: Agile Software Development has many advantages such as frequent delivery (Matharu et al., 2015; Penmetsa, 2017). ASD divides the entire project into smaller pieces or iterations, each of which is handled separately to control time and item change risks (Lai, 2019; Rajasekhar & Shafi, 2014). Testing activities are achieved in each iteration (Crispin & Gregory, 2009; Javed et al., 2019). Every iteration has to be fully integrated and carefully tested as a final production release (Penmetsa, 2017). Delivery time is a critical factor for agile product success (Sanan et al., 2016). Since the iterations make the software development effective and efficient to meet the customer’s requirements and contribute to the project’s success, it also makes the development process a little more complicated and time-consuming (Javed et al., 2019). This complication is because each iteration in ASD contains many activities such as requirement analysis, design, implementation, testing, and deployment (Javed et al., 2019). Short iterations in ASD imply that there must be an efficient testing process to avoid too much time being spent in the iteration (Olausson et al., 2013). Thus, the efficient testing process needs effective test cases. This study confirmed that good-quality test cases could reduce delivery time, as rated by respondents at 3.97, which is close to 4 (agree scale).

(3) Decreasing Development Cost: The continuous changes of requirements in ASD may affect a large number of test cases (Beer & Felderer, 2018) and increase the number of test cases to be executed, which can increase execution cost and duration of software projects (Do, 2016; Hyninen et al., 2018). The good-enough testing of software should sufficiently assess quality at a reasonable cost (Goesch et al., 2010). The good quality of test cases is essential for assuring the quality of software (Tran et al., 2019; Tudjarova et al., 2017) because the higher-quality test cases have a high chance to reveal defects, increase the performance of the system, and reduce the cost of testing and maintenance (Gómez et al., 2016; Kamde et al., 2006; Mohi-aldeen et al., 2014; Pressman, 2010; Yadav & Yadav, 2015). Thus, it can reduce the cost of development as indicated by the experts through the majority of them agreed (3.81) about it. Consequently, this condition explains why test case quality issue has attracted
researchers as demand for global quality software with rapid delivery has become more competitive for organizations due to the volatile nature of software testing.

(4) **Meet Customer Needs:** Companies worldwide adopt Agile methodologies to meet increased software complexity and evolving user demands (Matharu et al., 2015; Penmetsa, 2017). Customer satisfaction is one of the crucial ASD advantages (Matharu et al., 2015; Penmetsa, 2017). When the iteration is developed and tested, the customer sends the system for feedback which they later provide as feedback in the form of stories. When the required functionalities are delivered, customers stop writing stories and development (Olausson et al., 2013). The continuous changes in customer requirements increase the importance of testing practices in ASD methods (Penmetsa, 2017). The testing tasks in ASD should be appropriately prepared to cater for continuous changes in the requirements (Yu, 2018). In order to nimbly test a software system during ASD, it is crucial to identify what to test (e.g., requirements) and how to test it (i.e., test cases; Bucaioni et al., 2018). The test cases in ASD must be developed as the requirements evolve (Lewis, 2009). Therefore, most experts strongly agree (4.5) that good-quality test cases can meet the customer requirements.

(5) **Improving Adequacy of Testing:** In designing test cases, it is notable to ensure that testing achieves a certain level of thoroughness (Romli et al., 2020). This thoroughness will determine how adequate the testing is (Hayhurst, 2001). Adequacy of testing can be measured by code coverage, where there is a relationship between coverage criteria and fault detection (Ahmed et al., 2016). Hence, it can distinguish between good and bad test cases and determine whether the testing is enough (Zhu, 1995). Since good test cases have high coverage and high fault detection (Adlemon et al., 2018; Kochhar et al., 2019), they can provide good adequacy of testing, which the results confirmed through a high score from experts (4.33).

(6) **Reducing Defects:** The software testing goal is to design tests that have the highest likelihood of finding most errors and aids future bug fixes with a minimum amount of time, resources and efforts (Ahmed et al., 2016; Pressman, 2005; Qamar & Malik, 2019). Hence, good software testing allows the reduction of undetected defects in the developed software (Aziz, 2017; Lin et al., 2014; Petunova & Bērziša, 2017). Our results confirmed it via the experts’ opinion with 4.28 scale. Good testing depends on good-quality test cases (Causevic, Punnekat et al., 2012; Kumar & Ranjan, 2017; Lai, 2017).

### 4.1. Factors and criteria of TCQ

The respondents were asked to investigate and provide their rating on the proposed factors and criteria of TCQ in ASD. The experts provide feedback on the proposed 32 TCQ criteria, grouped into seven factors: documentation quality, management quality, maintainability, reusability, requirement quality, performance efficiency, and effectiveness. Table 4 displays the mean values of the 32 criteria of TCQ and the average of each group to provide the mean of the related factors. This mean value is calculated using SPSS Ver 26.0 using the descriptive analysis technique (MEAN function). The mean value or the average is calculated by adding up all the respondents’ feedback about the specific criterion for each item in the instrument as represented in the Likert Scale, then divided by the total number of respondents (36). The following are two examples of calculating the mean value; the first for calculating the mean of the criteria and the second one for calculating the mean value of the factors.

1. Criteria\_mean = \( \frac{\sum_{k=1}^{36} p_k}{n} \), \( n = 1, 2, \ldots, 35, 36 \)

Where \( n \) is the number of experts, \( p \) is the perspective value in the Likert scale (1 to 5) given by the experts.

Correctness = \( \frac{\sum_{k=1}^{36} p_k}{36} = \frac{154}{36} = 4.28 \) (154 is the total rating of 36 experts)
| Quality Factors               | Mean Values |
|------------------------------|-------------|
| **Documentation quality**    | 4.10        |
| Uniform format               | 4.00        |
| Correctness                  | 4.28        |
| Completeness                 | 4.08        |
| Consistency                  | 4.19        |
| Readability                  | 4.25        |
| Understandability            | 4.22        |
| Independent                  | 3.94        |
| Specific                     | 3.89        |
| **Management quality**       | 4.06        |
| Retrieval quality            | 3.97        |
| Recoverability               | 3.92        |
| Version control mechanism    | 4.31        |
| Prioritization and Organizability | 4.06    |
| Maintainability              | 4.04        |

(Continued)
| Quality Factors | Mean Values |
|-----------------|-------------|
| Simplicity      | 3.42        |
| Changeability   | 4.28        |
| Traceability    | 4.19        |
| Peer Review     | 4.08        |
| Update Regularly| 4.25        |
| Reusability     | 4.06        |
| Automaticity    | 4.08        |
| Extensibility   | 4.14        |
| Combinability   | 3.92        |
| Specific        | 4.33        |
| Measurable      | 3.94        |

Simplicity: Test cases should be small in size (in terms of lines of code), and unnecessary words or steps are to be avoided.

Changeability: A good test case should be well-modularized. The steps of the test case should be easily modified based on the change requirements.

Traceability: Traceability links should be maintained between test codes, requirements, and source codes.

Peer Review: A test case needs to be regularly reviewed whenever requirements are altered.

Update Regularly: A test case needs to be updated when the related requirements change.

Reusability: The ability of a test case to be run frequently by recording test data and test results as the format of testing tools.

Automaticity: The ability of a test case to be run frequently by recording test data and test results as the format of testing tools.

Extensibility: A test case should have the ability of easy adjustment and modification to meet the extension of user requirements.

Combinability: Some test cases in higher levels need to combine test cases in base levels of testing.

Simplicity: The needed extensions and their application rules should be easily adapted to the current testing procedures.

Repeatable: A test case should provide the same result if it is run many times.

Specific: The requirement should state what is expected precisely, and it should be clear, comprehensive, and precise.

Measurable: It is possible to verify that the requirement has been met. The requirement should specify a fixed performance against a predefined set of test cases.
| Quality Factors          | Mean Values |
|-------------------------|-------------|
| Attainable              | 3.75        |
| Realisable              | 3.94        |
| Traceable               | 4.19        |
| **Performance Efficiency** | 4.09        |
| Bug detection           | 4.42        |
| Resource Utilization    | 3.92        |
| Time behaviour          | 3.94        |
| **Effectiveness**       | 4.47        |
| Coverage                | 4.28        |
| Accuracy                | 4.67        |
(2) Documentation quality = MEAN (Uniform format, Correctness, Completeness, Consistency, Readability, Understandability, Independent, Specific)

As shown in Table 4, the mean of all TCQ criteria is more than 3.0, which ranged from 3.42 (simplicity–maintainability) to 4.67 (accuracy–effectiveness). That means the respondents accepted all the proposed criteria. Twenty-one out of the 32 criteria receive a Likert score of 4.0 (i.e., agree) and above (strongly agree). By the mean value, it can be known which criteria are more suitable to measure good TCQ among the respondents. The mean of the factors of TCQ in ASD is calculated by getting the average of the related criteria means. The highest score is 4.47 for effectiveness, while the lowest is for requirement quality (4.03). However, the respondents agreed with all the factors with scale value of more than 4 (i.e., agree). In addition, most respondents comment that these factors and criteria are reasonable, satisfied, and sufficient. Following are the descriptions of the experts’ feedback about the factors and criteria of TCQ.

(1) Documentation Quality: A documentation of a test case is considering the essential item for generation, change, revision, integration, and reuse of test case (Lai, 2017). According to Kochhar et al. (2019), well commented, named, and designed test cases may serve as good reference documentation. Aamir and Khan (2017) assured that test case documentation should be performed to develop a quality product. In addition, most developers in Li et al. (2016) survey emphasized the importance of test case documentation to improve product quality. The mean of documentation quality is 4.1, which is calculated based on its criteria score. This value refers to the respondents agreeing that test cases in ASD should be well documented. The eight proposed criteria for this factor are uniform format, correctness, completeness, consistency, readability, understandability, independent, and specific. The findings showed that the whole criteria are essential for measuring test cases documentation quality, as shown in Table 4.

(a) Uniform Format: Uniform test case items can assist in generating practical test cases which reach the test objective (Lai, 2017). The test case documentation should quickly inspect uniform items (Lai, 2017). Therefore, most respondents agreed (the mean is 4.00) that this feature is one of the criteria for getting good-quality test cases.

(b) Correctness: Correctness of a test case is the degree to which a test is free from faults in its specification, design, and applied algorithms, as well as in returning the test results (Zander-Nowicka et al., 2008; Zeiss et al., 2007). The test is correct when it always returns correct test verdicts and when it has reachable end states (Zander-Nowicka et al., 2008; Zeiss et al., 2007). The test case should provide the correct results with the needed degree of precision. Any ambiguity should be removed in test case execution to gain high quality (Adlemo et al., 2018; Bowes et al., 2017; Kamde et al., 2006). The correctness is the most exciting criteria of test case documentation quality based on experts’ opinion, with a mean score of 4.28.

(c) Completeness: A test case should contain all relevant information for its execution (Tran et al., 2019). Sometimes test cases might be incomplete and not cover some functional specifications (Chernak, 2001). In addition, test-design logic could be incomplete, and some of the necessary test conditions could be missing in test case specifications (Chernak, 2001). Therefore, when test cases cover all system functionalities under test, they will provide good-quality test cases (Boghady et al., 2011). The respondents (experts) provide this high criterion score (4.08) to insist on the completeness of test case documentation.

(d) Consistency: Consistent test case means to always adhere to the same rules. Using the same pattern in organizing the test cases makes the testing easier (Adlemo et al., 2018; Tran et al., 2019). If the test cases are consistent, they are much easier to combine, and the cost of switching between test cases (context switch) becomes much lower (Adlemo et al., 2018). The test case should be consistent to ensure high quality (Lai, 2017).
Therefore, the high score of 4.19 from respondents on this criterion indicates that most experts agree to its importance to test case documentation quality.

(e) **Readability**: A test case should be free of ambiguity. The reader of a test case should not need to make assumptions about implementing any requirement, and a new tester would not require any help to execute (Adlemo et al., 2018). A clear or readable test case is valuable to keep the cost of the executioner low to the test case (Adlemo et al., 2018). According to Beniwal (2015), most defects are found through reading, so the test case readability should be more considered. Test case readability is optimizing the test case understandability (Daka & Fraser, 2014; Setiani et al., 2020). So, any tester can understand it by reading it once (Adlemo et al., 2018; Bowes et al., 2017; Kaner, 2003). The readable test cases benefit developers when performing maintenance tasks on the source code (Grano et al., 2018). Readability is very important and highly rated by the respondents, hence scored the second-highest (4.25) among other test case documentation.

(f) **Understandability**: According to Tran et al. (2019), the information of a test case (e.g., name, objective, precondition, steps, terms) should be easily understood by both testers and developers. Many researchers claim that the test case should be understandable for its quality (Bowes et al., 2017; Daka & Fraser, 2014; Jovanovikj et al., 2018; Kochhar et al., 2019; Li et al., 2016; Setiani et al., 2020; Shrivastava & Jain, 2010; Tran et al., 2019). Understanding the test cases by developers takes a long time, especially automatically generated tests compared to manually written ones (Shamshiri et al., 2018). As shown in Table 4, the score of “understandability” is high (4.22), referring to the experts’ insistence on the necessity of test case to be understandable.

(g) **Independent**: A test case must be self-standing and executed without any tester performing it. One must run a test case individually, and the success or failure of an earlier test case must not affect the outcome of the test case (Kochhar et al., 2019; Limaye, 2009). This criterion allows adding new test cases without considering any effects they might have on existing test cases (Adlemo et al., 2018; Bowes et al., 2017). According to Adlemo et al. (2018), the independent test case allows a tester to execute them feasibly. Also, it can be easier to use them as building blocks to build flows. In addition, the stronger the independence of a test case, the better its reusability (Juan et al., 2010). Despite the benefits of test case independence, it did not receive a favourably high score from the respondents (3.94). However, it is still close to agreeing that this feature is one of the criteria of test case documentation quality.

(h) **Specific**: A good test case should test one aspect of a requirement and address a unique functionality, thereby not wasting time and resources (Chauhan, 2010; Kochhar et al., 2019). Each test case should test one requirement in a specific way, and if the requirements have many facets, many test cases should be written (Adlemo et al., 2018; Bowes et al., 2017; Kamde et al., 2006; Kaner, 2003; Lai, 2017). The Single-Condition Tests principle (Meszaros, 2007) suggests that test cases should not verify many functionalities at once to avoid test obfuscation. These criteria of a test case make it as test focus, so it is easier to understand and, therefore, it eases fault localization and debugging tasks (Grano, 2019). However, based on our findings, this criterion got the lowest rate among other criteria (3.89). That the figure is close to agree refers to the fact that the criterion is accepted to be one of the quality criteria of test cases.

One of the respondents commented on this factor (documentation quality) as: “I agree with all these points but put in mind if these implemented completely, delivery will be late, but it will be solid and reduce bugs”. This comment refers to the importance of quality content for quality test cases, although it implementation can hinder fast delivery of the product.
(2) **Management Quality**: Test case management is concerned with the control processes of the test cases (Kao et al., 2016). Due to the requirements changes in ASD, a test case like other development documents must be flexibly changed or adjusted and easily retrieved and recovered (Lai, 2017). The features of a test case should be manageable to create room for quick recovery of previous versions, easy retrieval of the different levels of the test cases, and version control which is the detailed record of the tester, date of revision, reasons, and other related activity (Lai, 2017). The respondents (experts) agreed that a management quality factor could provide good quality of test cases in ASD with a mean score of 4.06, which refers to the agreed scale. Table 3 also shows that the respondents accept the criteria of this factor.

(a) **Retrieval Quality**: Test cases can be retrieved quickly by their identifications at different levels (Lai, 2017). This feature is significant for the quality of test cases, and to be sure about this point, the respondents were asked to rate it. Their rating was 3.97, which is acceptable as one of the test case quality criteria.

(b) **Recoverability**: It means the degree to which the previous state of the test case can be re-established quickly in the event of an interruption or a failure (Lai, 2017). The respondents gave this criterion the lowest rate (3.92) among management quality criteria, but it is still acceptable as one of the test case quality criteria.

(c) **Version Control Mechanism**: A good test case contains a detailed record of the revision date, reasons, tester, and related activity (Lai, 2017). Any changed test case has the complete documents in the version control system (Lai, 2017). The version control systems store and reconstruct past versions of program source code (Ball et al., 1997). “Version control mechanism” had the highest agreement from most respondents, with 4.31.

(d) **Prioritization and Organization**: Test cases should be arranged in chronological order based on criticality. According to Afzal (2007), not all test cases are equally important; therefore, the test cases need to be prioritized. The test case prioritization is needed because full regression testing requires a lot of time and system resources; therefore, there is a need to identify which test cases should be run first (Berberyana & Ali, 2019). Hence, the respondents gave this criterion a high score of 4.06, which is more than agree. In addition, one of the respondents commented on this factor (management quality) as “Sometimes when we don’t have much time, so we are covering importing test cases”, which means that they test the most critical test cases first in case of limited time. This comment supports the prioritization of test cases “prioritization and organizable”.

(3) **Maintainability**: Zeiss et al. (2007) define test case maintainability as the degree to which the specified test cases could be modified with ease due to changes in the software. Test cases should be written so that it is easy to maintain. Therefore, for any changes in requirements, the tester should effortlessly manage the test cases (Adlemo et al., 2018; Bowes et al., 2017; Lai, 2017). According to the findings of this study, the academic and industry experts confirmed that maintainability influences test case quality. Consequently, this factor was deemed important to measure test case quality in ASD. Therefore, the average criteria scale of maintainability is 4.04, which means the experts insist on test case maintainability. Most of the criteria (i.e., changeability, update regularly, traceability, peer review) had more than 4 rating scales, making them acceptable to measure test case maintainability. Some respondents commented on test case maintainability, such as “Add the negative impact of writing unnecessary or duplicate test cases” and “Whenever requirements or code change, test cases should be fixed”. These two comments insist on not including the unnecessary test cases and updating the test cases in case any changes occur, perspectivity.

(a) **Simplicity**: A test case must be simple to understand so that the tester does not have to do any research or invention of what is to be done next (Limaye, 2009). Li et al. (2016) claim that developers do not need help on simple test cases compared to complex test cases. The
simplicity can lead to less chance of making mistakes and ease a test case’s maintainability (Adlemo et al., 2018; Bowes et al., 2017; Lai, 2017). Simplicity was the only acceptable criterion of test case maintainability, where 11 out of 36 respondents disagreed with. While the average rating is still up to 3.42, this criterion is still acceptable.

(b) **Changeability**: The test case steps should be easily modified based on the changing requirements. A test case is changeable if its structure and style are such that any changes can be made easily, completely, and consistently (Juan et al., 2010). Since agile welcomed the changes, this criterion (i.e., changeability) is critical in the test case quality. Therefore, the experts gave it a high score (4.28).

(c) **Traceability**: The ability to trace requirements in a specification to their origin from higher to lower-level requirements in a set of documented links (Ali, 2006). To nimble test a software system during its agile development, it is crucial to identify what to test (e.g., requirements, code and development artefacts) and how to test it (i.e., test cases). The traceability of test cases will help to easily change the impacted test cases if any of the requirements get changed (Beer & Felderer, 2018). Traceability links should be maintained between test cases, code, and requirements (Kochhar et al., 2019). The traceability rating scale is 4.19, implying that this criterion is accepted in the test case maintainability criteria.

(d) **Peer Review**: The test case should be corrected by the test case reviewer. Whenever requirements are altered, test cases need to be updated. Testers should plan and update test cases for sprint stories (Collins et al., 2012). Therefore, test cases need to be regularly reviewed and revised, and new tests need to be written to exercise different parts of the software to potentially find more defects (Quadri & Farooq, 2010). Reviewing test cases and regular communication between developers and testers were also highly recommended by practitioners (Tran et al., 2019). In our findings also, the respondents recommended it by giving it a high score (4.08).

(e) **Update Regularly**: A test case must be updated when the underlying code changes (Adlemo et al., 2018). According to Li et al. (2016), developers can more easily identify test cases that relate to some new or modified functionality of the system by relying on up-to-date documentation. Due to the importance of updating test cases, the respondents provide a high score of 4.25.

(4) **Reusability**: Reusable test case is a test case that can be used at various application-levels (Juan et al., 2010). Some test cases in basic testing levels assist in combining into test cases that are of higher levels (Lai, 2017). Hence, test cases should be reusable to improve the quality and efficiency of agile testing, where time is a rare resource that makes reusable test cases very attractive (Adlemo et al., 2018; Lai, 2017). For this, the respondents confirmed that the reusability of a test case should be one of the TCQ factors through the average Likert scores for the proposed criteria of test case reusability. The respondents agreed that the test cases should have the characteristics such as automaticity, extensibility, simplicity, repeatable, and combinability to be reusable.

(a) **Automaticity**: The automaticity of a test case is the ability of the test case to be run frequently by recording test data and test results as the format of testing tools (Lai, 2017). The automated test cases can help deliver more effectively and in shorter timescales (Rajasekhar & Shafi, 2014). Therefore, it would be better to automate the test cases (Aamir & Khan, 2017). Our respondents rate this criterion 4.08, which is more than agree.

(b) **Extensibility**: Each test case should have the ability of easy adjustment and modification to meet the extension requirements and increase the efficiency of maintenance operation (Lai, 2017, 2019). The rating scale that this criterion gains is 4.14, which show the importance of this criterion to reusability.
(c) **Combinability:** Test cases in higher levels of testing need to combine test cases at base levels to increase the testing efficiency (Lai, 2017). However, this criterion is not highly favoured by the experts based on their rating (3.92), close to being agreed, but still acceptable.

(d) **Simplicity:** The needed extensions and application rules should be easily adapted to the current testing procedures. The more items a test case tests, the more complex it would be, and thus the less reusable (Rava & Daengdej, 2014). Therefore, the test case must be simple to be reusable. The experts confirm this by providing a 4.08 rating scale, which is more than agree with this criterion.

(e) **Repeatable:** A test case should provide the same result if run many times for the same inputs (Adlemo et al., 2018; Kamde et al., 2006; Kochhar et al., 2019; Sundmark et al., 2005; Tran et al., 2019). If a test case is repeatable, it will be fast and efficient to execute (Adlemo et al., 2018; Eldh et al., 2011; Kamde et al., 2006). One of the respondents commented on this factor “Repeat test leads to higher cost in development. Therefore, good and specific test cases can reduce the time and effort”. This comment is negative on repeatability but positively impacts on the specific criterion. However, the mean of repeatability is high (4.11), which means the respondents unanimously agreed. The repeatable test cases reduce the time, and effort as they help to reduce the time by not rewriting the same test case many times whenever needed.

(5) **Requirement Quality:** Requirement is a statement describing the proposed system where all stakeholders agree must be made true for the customers’ problems to be genuinely solved (Hussain et al., 2016). It is essential to create test cases based upon the requirements (Craig & Jaskiel, 2002). The requirements should be well known before designing test cases (Beer et al., 2017) because the aim of testing is requirements verification (ensure that the requirement is correct) and validation (confirm that the solution meets the requirements), by defining test cases and designing a rationale (Aliala et al., 2019; Limaye, 2009; Schledlauer, 2012). The respondents were asked about the rating of impact on the quality of test cases by the quality of requirements. According to Mannion and Keepence (1995), the criteria to measure requirement quality should be specific, measurable, attainable, realizable, and traceable (SMART). Other criteria were accepted as well, but they have a mean less than 4 in the range of 3.75 to 3.94 (see, Table 4) which is close to 4. The respondents sometimes suffer in analyzing the customer requirements as one of the respondents commented about this issue “We really facing issues with requirements we receive from a business analyst. As we have a lot of discussions and questions because requirements are not ready for implementation. Also, the lack of understanding of the existing implementation which results difficulties in the transition to new requirements or modifications.”

(a) **Specific:** The requirement should say what is required, and this includes that it should be clear, complete, and accurate (Mannion & Keepence, 1995; Quadri & Farooq, 2010). The requirement should not have unnecessary redundancy of information. The extraneous words should be removed. Most respondents agreed or strongly agreed about specific where its rating scale becomes 4.33, which is the most significant scale among other criteria of requirement quality. As a result, the story requirements should be specific to provide good-quality test cases in ASD. Twenty of respondents strongly agreed that requirements should be specific.

(b) **Measurable:** It is possible to verify that the requirement has been met. The level of detail required to describe and set up the corresponding test is a strong indicator of whether the requirement should be broken down into sub-requirements. The requirement should specify a fixed performance against a predefined set of test cases (Mannion & Keepence, 1995). In addition, the requirement should be specific to be measurable (Mannion & Keepence, 1995). The respondents agreed to this criterion by giving it an average of Likert Scale 3.94, which is the exact value of “realisable”.

---

Barraood et al., Cogent Engineering (2022): 9: 2082121
https://doi.org/10.1080/23311916.2022.2082121
(c) **Attainable**: The attainable requirement means that implementation of the requirements can be achieved on time, within budget (Mannion & Keepence, 1995). We asked the experts whether this criterion was important for test case quality, but they were unsure about it, so they gave it the lowest scale of 3.75. Though this value is low, it is still acceptable.

(d) **Realisable**: The realisable requirement can achieve the requirement given what is known about the constraints under which the system and the project must be developed (Mannion & Keepence, 1995).

(e) **Traceable**: The ability to trace a requirement from its conception through its specification to its subsequent design, implementation, and test, that to say, from the requirement to test cases across design and implementation artefacts. This implies, from the requirement to test case across design and implementation artefacts (Bucaioni et al., 2018; Mannion & Keepence, 1995). The traceability between requirements and test cases can provide valuable feedback on how the verification and validation have been operated and its progress status in the sprints (Bucaioni et al., 2018). Based on our survey findings, 16 and 15 of respondents strongly agreed and agreed, respectively, that test cases should be traceable. One of the respondents opined that “Traceability of test cases is a very important topic for customer”. Based on this importance, the respondents provided a high score (4.19) for traceability, which is considered the second-highest score related to requirement quality after Specific (4.33).

(6) **Performance Efficiency**: Performance efficiency is one of the ISO/IEC 25010 quality characteristics (ISO-IEC 25010:2011) which is defined as “the capability of the software product to provide appropriate performance, relative to the number of resources used, understated conditions.” Efficiency is frequently used as a quality factor for assessment (Yan et al., 2019). A good test case quality is more likely to discover severe bugs (Adlemo et al., 2018; Kaner, 2003), which refers to test case efficiency (Limaye, 2009). The respondents were asked to provide their scales about the proposed criteria (i.e., bug detection, resources utilization, and time behaviour) to measure the performance efficiency of test cases in ASD. The results showed that the three criteria are accepted.

(a) **Bug Detection**: A test case should possess the ability to discover bugs easily. Bug detection is one of the main reasons for writing test cases (Kochhar et al., 2019). When practitioners write a new functionality or add a piece of code, they need to test whether that code is working fine or not (Kochhar et al., 2019). A good test case quality is more likely to discover severe bugs (Adlemo et al., 2018; Kaner, 2003). This should help to identify issues and weaknesses of features/functions (Tran et al., 2019). The results shown in Table 4 refers that the respondents strongly agree about bug detection. The mean rating for bug detection is 4.42.

(b) **Resource Utilization**: The number of computing resources and code required by a program to perform its function should be enough (ISO-IEC 25010:2011). The respondents closely agreed about this criterion for good quality of test cases, so they scaled it as 3.92.

(c) **Time Behavior**: Test cases should use tags or categories, such as slow tests, fast tests, and so far, to run a specific set of tests easily at a time (Adlemo et al., 2018). The respondents were asked to rate this criterion as one of the performance efficiency criteria for test case quality. The result was 3.94, which is acceptable.

(7) **Test Case Effectiveness**: A test case effectiveness means that the test case covers the expected requirements (Tran et al., 2019). It shows how to determine whether a set of test cases sufficiently and effectively reveal defects (Chernak, 2001). To design a good and practical test case, a test case should cover all features and the expected requirement, but the developer should not make too many test cases (Tran et al., 2019; Yamaura, 1998). The value of test case effectiveness as a result of calculating the average of coverage and accuracy of a test case is 4.47, which is the highest value of the proposed factors of test
case quality (refer to Table 4). The results showed that the proposed criteria of test case effectiveness (i.e., coverage and accuracy) are very important for high TCQ.

(a) **Coverage**: Coverage should be used to understand what feature(s) is/are missing or covered in test cases as expected in the requirements. Coverage information can help developers and testers find parts of the code that are not covered and might contain bugs (Kochhar et al., 2019). According to Grano (2019), a test case is more effective when it has a high statement coverage and does not contain test smells. The results showed that the proposed criterion of test case effectiveness (i.e., coverage) is essential for good TCQ. The average ranking is 4.67, which is the highest scale among other criteria of test case quality.

(b) **Accuracy**: A test case is accurate when it produces results that correspond to the expected outputs (Adlemo et al., 2018; Kamde et al., 2006). It must describe the desired result correctly to avoid confusion on whether the test case has passed or failed (Limaye, 2009). If the tester does not know the expected output and assumes the result given by the computer is correct, the tester will not be able to detect these logic errors (Atmadja & Shuai, 2011). The respondents strongly agreed about this criterion with a high Likert scale (4.28) given.

5. **Conclusion and future work**

Software testing is an essential activity in Agile methods. Writing test cases, therefore, has a great significant impact on the testing process’s activity. This paper presented an exploratory study on the investigation of test case quality and its importance in ASD from experts’ perspectives. A survey was conducted on 36 academic and industry experts from different countries to examine the importance of test case quality in ASD. The survey investigates 32 criteria, grouped into seven factors, comprising documentation quality, management quality, maintainability, reusability, requirement quality, performance efficiency, and effectiveness of TCQ in ASD. The experts insist on the importance of good quality of test cases in ASD to improve the software quality, meet customers’ needs, improve the adequacy of testing, reduce defects and bugs, decrease delivery time, and reduce the development cost. The positive response of the experts on the factors and criteria indicates its suitability for measuring the quality of test cases in ASD. This study is part of research on designing a measurement model for the quality of test cases in Agile Software Testing. At the moment, none of the test case studies constructed the same factors related to documenting its quality, together with related criteria and metrics. Because the system is tested using the quality assurance technique indicated in the proposed test case quality model, the research could help to produce high-quality system applications utilising the Agile testing approach. Furthermore, it may contribute to the software quality auditor and self-quality assessment in software industries. The future work will focus on defining metrics and measurements for measuring these verified factors and criteria of test case quality in Agile projects.

**Acknowledgements**

The authors would like to express their gratitude to Hadhramout University in Yemen, the Awang Had Salleh Graduate School, School of Computing, and Human-Centered Computing Research Lab Universiti Utara Malaysia, for their continuous support and encouragement to achieve up to this level in this research.

**Funding**

The authors received no direct funding for this research.

**Author details**

Samer Obaid Barraood1,2  
E-mail: sammorahobaid@gmail.com  
ORCID ID: http://orcid.org/0000-0002-6106-7394  
Haslina Mohd2  
Fauziah Baharom2  
1 College of Computers and Information Technology, Hadhramout University, Hadhramout, Yemen.  
2 School of Computing Universiti Utara Malaysia, Sintok, Kedah, 06010 Malaysia.

**Disclosure statement**

No potential conflict of interest was reported by the author(s).

**Citation information**

Cite this article as: An initial investigation of the effect of quality factors on Agile test case quality through experts’ review, Samer Obaid Barraood, Haslina Mohd & Fauziah Baharom, Cogent Engineering (2022), 9: 2082121.

**References**

Aamir, M., & Khan, M. N. A. (2017). Incorporating quality control activities in scrum in relation to the concept of test backlog. Sādhanā, 42(7), 1051-1061. https://doi.org/10.1007/s12046-017-0688-7  
Adlemo, A., Tan, H., & Tarassov, V. (2018). Test case quality as perceived in Sweden. 5th International Workshop
Barraood et al., Cogent Engineering (2022), 9: 2082121
https://doi.org/10.1080/23311916.2022.2082121

International Workshop on Requirements Engineering and Testing Measuring, 25–32.

Beniwal, R. (2015). Analysis of testing metrics for object oriented applications. 2015 IEEE International Conference on Computational Intelligence & Communication Technology, 41–46. https://doi.org/10.1109/CICT.2015.34

Berberyan, H., & Ali, S. (2019, November). Automation regression testing for SasAm website. In CS & IT Conference Proceedings (Vol. 9, pp. 13). CS & IT Conference Proceedings.

Boghdady, P. N., Badr, N. L., Hashem, M., & Tolba, M. F. (2011). A proposed test case generation technique based on activity diagrams. International Journal of Engineering & Technology JET-IJENS, 11(3), 17–57.

Bowes, D., Hall, T., Petric, J., Shippey, T., & Turhan, B. (2017). How good are my tests? In International Workshop on Emerging Trends in Software Metrics, WETSoM (pp. 9-14). IEEE. https://doi.org/10.1109/WETSoM.2017.2

Bucanici, A., Cicchetti, A., Ciccotizzi, F., Kodali, M., & Szöj J., M. (2018). Alignment of requirements and testing in Agile: An industrial experience. In S. Latifi (Ed.), Information Technology-New Generations, Advances in Intelligent Systems and Computing 738 (pp. 225–232). Springer.

Causveic, A., Punnekatt, S., & Sundmark, D. (2012). Quality of test design in test driven development. 2012 Eighth International Conference on the Quality of Information and Communications Technology, 224, http://urn.kb.se/resolve?urn=urn:nbn:se:mdh:dio-18773

Causveic, A., Sundmark, D., & Punnekatt, S. (2012). Test case quality in test driven development. A study design and a pilot experiment. The EASE 2012, 223–227. https://doi.org/10.1049/ic.2012.0029

Chauhan, N. (2010). Software testing: principles and practices. Oxford university press.

Chern, Y. (2001). Validating and improving test-case effectiveness. IEEE Software, 18(1), 81–86. https://doi.org/10.1109/52.903172

Collins, E., Dias-Neto, A., & de Lucena, V. F., Jr. (2012). Strategies for Agile software testing automation: An industrial experience. Computer software and Applications Conference Workshops (COMPSACW), 2012 IEEE 36th Annual, 440–445.

Craig, R. D., & Jaskel, S. P. (2002). Systematic software testing. Artech House.

Crispin, L., & Gregory, J. (2009). Agile testing: A practical guide for testers and agile teams (1st ed.). Pearson Education, Inc.

Daka, E., & Fraser, G. (2014). A survey on unit testing practices and problems. 2014 IEEE 25th International Symposium on Software Reliability Engineering, 201–211.

Do, H. (2016). Recent Advances in Regression Testing Techniques. Advances in Computers, 103, 53–77. https://doi.org/10.1016/bse.acom.2016.04.004

Elidh, S., Hansson, H., & Punnekatt, S. (2011). Analysis of mistakes as a method to improve test case design. Fourth IEEE International Conference on Software Testing, Verification and Validation Analysis, 70–79. https://doi.org/10.1109/ICST.2011.52

Fischbach, J., Femmer, H., Mendez, D., Fucci, D., & Vogelsang, A. (2020). What makes agile test artifacts useful? An activity-based quality model from
a practitioners’ perspective. ESEM, ‘20. https://arxiv.org/abs/2009.01722
Garousi, V., Felderer, M., Kuhrmann, M., & Herkilo, K. (2017). What industry wants from academic software testing? Hearing practitioners’ opinions. EASE’17, 65–69. https://doi.org/10.1145/3084226.3084264
Gay, G., Rojan, A., Staats, M., Whalen, M., & Heimdahl, M. P. E. (2016). The effect of program and model structure on the effectiveness of mc/dc test adequacy coverage. ACM Transactions on Software Engineering and Methodology (TOSEM), 25(3), 1–34. https://doi.org/10.1145/2934672
Goeschl, S., Herp, M., & Wais, C. (2010). When agile meets OO testing: A case study. Proceedings of the 1st Workshop on Testing Object-Oriented Systems, 10.
Gómez, O. S., Monte, B., & Monte, B. (2016). Impact of CS programs on the quality of test cases generation: an empirical study categories and subject descriptors. ICSE ’16 Companion, 374–383. https://doi.org/10.1145/2889160.2889190
Grano, G., Scalasbinio, S., Gall, H. C., & Oliveto, R. (2018). An empirical investigation on the readability of manual and generated test cases. Proceedings of the 26th Conference on Program Comprehension, 348–351.
Grano, G. (2019). A new dimension of test quality: Assessing and generating higher quality unit test cases. The 28th ACM SIGSOFT International Symposium on Software Testing and Analysis (ISSTA ’19), July 15–19. https://doi.org/10.1145/3293882.3339884
Gupta, N., Yadav, V., & Singh, M. (2018). Automated regression test case generation for web application: A survey. ACM Computing Surveys (CSUR), 51(4), 87. https://doi.org/10.1145/3232520
Hollowell, M. R., & Gambatese, J. A. (2009). Qualitative research: Application of the Delphi method to CEM research. Journal of Construction Engineering and Management, 136(1), 99–107. https://doi.org/10.1061/(ASCE)CO.1943-7862.0000137
Hariharan, S., Pando, N., & Acharya, A. A. (2014). Scrum testing with backlog management in Agile development environment. International Journal of Computer Science and Engineering, 2(3), 187–192. http://www.ijcseonline.org/public_paper/38-IJCSSE-00144.pdf
Hayhurst, K. J. (2001). A practical tutorial on modified condition/decision coverage. DIANE Publishing.
Horváth, F., Vancsic, B., Vidacs, L., & Beszedes, Á. (2015). Test suite evaluation using code coverage based metrics. SPLST’15, 46–60.
Humble, J., & Farley, D. (2010). Continuous delivery: Reliable software releases through build, test, and deployment automation (Adobe reader). Pearson Education.
Hussain, A., Mpkojogu, E. O. C., & Kamal, F. M. (2016). The role of requirements in the success or failure of software projects. International Review of Management and Marketing, 6(57), 306–311.
Hyyninen, T., Kasurinen, J., Knutas, A., & Taipale, O. (2018). Software testing: Survey of the industry practices. 2018 41st International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO), 1449–1454. https://doi.org/10.23919/MIPRO.2018.8400261
IEEE 829. (2008). IEEE Standard for Software and System Test Documentation. IEEE Computer Society.
ISO/IEC 25010:2011. (2011). ISO/IEC 25010: 2011. In Systems and software engineering-systems and software quality requirements and evaluation (SQuARE)-System and software quality models. ISO. https://doi.org/10.34033/30215101
ISO/IEC 9126-1. 2000. Information technology — Software product quality — Part 1: Quality model. In ISO/IEC Fdis 9126-1. 2001-126.ISO. IEC. http://www.cse.unsw.edu.au/~cs3710/PMaterials/Resources/9126-1Standard.pdf
ISO/IEC25010. (2011). ISO/IEC25010: 2011 Systems and software engineering-Systems and software Quality Requirements and Evaluation (SQuARE)-System and software quality models. International Organization for Standardization. https://www.iso.org/obp/ui/#iso:std:iso-iec:25010:ed-1:v1:en
ISO/IEC25023. (2016). INTERNATIONAL STANDARD ISO/ IEC 25023. 2016.
Javed, K., Khan, A. H., & Tubbasm, L. (2019). Critical analysis of software development methodologies based on project risk management. International Journal of Academic Research in Business and Social Sciences, 9(12), 248–261. http://dx.doi.org/10.6007/IJARBS/v9-i12/6720
Jovanovikj, J., Narasimhan, V., Engels, G., & Sauer, S. (2018). Context-specific quality evaluation of test cases. Modelsvard, 594–601. https://doi.org/10.5220/000672405940601
Juan, Z., Lizihi, C., Weiqing, T., & Song, Y. (2010). Test case reusability metrics model. Iccctd, 294–298. https://doi.org/10.1109/IICCTD.2010.5645869
Juhnke, K., Tichy, M., & Houdek, F. (2018). Quality Indicators for Automotive Test Case Specifications. SEERTS 2018: Workshop on Software Engineering for Applied Embedded RealTime Systems @ SE18, 96–100.
Juhnke, K., Tichy, M., & Houdek, F. (2021). Challenges concerning test case specifications in automotive software testing: Assessment of frequency and criticality. Software Quality Journal, 29(1), 39–100. https://doi.org/10.11219/020-09523-0
Kamide, P. M., Nandavadekar, V. D., & Pawan, R. G. (2006). Value of test cases in software testing. Management of Innovation and Technology, 2006 IEEE International Conference On, 2, 668–672. https://doi.org/10.1109/IMICT.2006.262303
Kaner, C. (2003). What is a good test case? Software Testing Analysis & Review Conference (STAR) East., 1–16.
Kao, C. H., Lin, C. C., & Lu, H. T. (2016). Towards automatic performance testing for rest-based web applications. The Eleventh International Conference on Software Engineering Advances-ICSEA, 81–87.
Kärklinä, K., & Pirto, R. (2018). Quality metrics in Agile software development projects (pp. 21). Information Technology & Management Science (RTU Publishing House).
Kayes, I., Sarker, M., & Chakareski, J. (2016). Product backlog rating: A case study on measuring test quality in scrum. Innovations in Systems and Software Engineering, 12(4), 303–317. https://doi.org/10.1007/s11334-016-0271-0
Kochhar, P. S., Xia, X., & Lo, D. (2019). Practitioners’ views on good software testing practices. 2019 IEEE/ACM 41st International Conference on Software Engineering: Software Engineering in Practice (ICSE-SEIP), 61–70. https://doi.org/10.1109/ICSE-SEIP.2019.00015
Kumar, S., & Ranjan, P. (2017). A phase wise approach for fault identification. Journal of Information and Optimization Sciences, 39(1), 223–237. https://doi.org/10.1080/02522667.2017.1380420
Lai, S.-T. (2017). Test case quality management procedure for enhancing the efficiency of IID continuous testing. Journal of Software, 12(10), 794–806. https://doi.org/10.17706/jsw.12.10.794-806

Lai, S.-T. (2019). Applying continuous integration for increasing the maintenance quality and efficiency of web app. International Journal of Software Engineering & Applications (IJSEA), 10(1), 37–50. https://doi.org/10.5121/ijsea.2019.10103

Leffingwell, D. (2010). Agile software requirements: Lean requirements practices for teams, programs, and the enterprise. Addison-Wesley Professional.

Lewis, W. E. (2009). Software testing and continuous quality improvement (3rd ed.). Taylor & Francis Group.

Li, B., Vendome, C., Linares-Vásquez, M., Poshyananda, D., & Kraft, N. A. (2016). Automatically documenting unit test cases. 2016 IEEE International Conference on Software Testing, Verification and Validation (ICST), 341–352.

Limaye, M. G. (2009). Software testing. Tata McGraw-Hill Education.

Lin, C., Kang, K., & Kapfhammer, G. M. (2014). Test suite reduction methods that decrease regression testing. Information And Software Technology, 56(10), 1322–1344. https://doi.org/10.1016/j.infsof.2014.04.013

Lynn, M. R. (1988). Determination and quantification of content validity. Nursing Research.

Mannion, M., & Keepence, B. (1995). SMART requirements. ACM SIGSOFT Software Engineering Notes, 20(2), 42–47. https://doi.org/10.1145/224155.224157

Matharu, G. S., Mishra, A., Singh, H., & Upadhyay, P. (2015). Empirical study of agile software development methodologies: A comparative analysis. ACM SIGSOFT Software Engineering Notes, 40(1), 1–6. https://doi.org/10.1145/2691208.2691233

Meszaros, G., Smith, S. M., & Andreo, J. (2003). The test automation manifesto. Conference on Extreme Programming and Agile Methods, 73–81.

Meszaros, G. (2007). XUnit test patterns: Refactoring test code. Pearson Education.

Mohamed, S. F. P. (2015). A process based approach software certification model for agile and secure environment. Universit Utara Malaysia.

Mohi-alddeen, S. M., Deris, S., & Mohamed, R. (2014). Systematic mapping study in automatic test case generation. IOS Press, 703–720. https://doi.org/10.3233/978-91-631499-4-3-7-703

Nawaz, A., & Malik, K. M. (2008). Software testing process in agile development. In Computer Science Master Thesis, Blekinge Institute of Technology. Citeeseer.

Olaussen, M., Rossbreg, J., Ehn, J., & Sköld, M. (2013). Pro team foundation service. In Illustrate, Apress (pp. 404). Apress.

Parveen, T., Tilley, S., & Gonzalez, G. (2007). A case study in test management. Proceedings of the 45th Annual Southeast Regional Conference, 82–87.

Pennetta, J. R. (2017). Agile testing. In H. Mohanty, J. R. Mohanty, A. K. Balakrishnan (Eds.), Trends in software testing (pp. 19–33). Springer Science+Business Media. https://doi.org/10.1007/978-981-10-1415-2

Petunova, O., & Bérzilo, S. (2017). Test case review processes in software testing. Information Technology and Management Science, 20(1), 48–53. https://doi.org/10.1515/itms-2017-0008

Pfaller, C., Wagner, S., Universit, T., & Wiemann, M. (2008). Multi-dimensional measures for test case quality. Software Testing Verification and Validation Workshop, 2008. ICSTW’08. IEEE International Conference On, 364–368.

Pressman, R. S. (2005). Software engineering: A practitioner’s approach. Palgrave Macmillan.

Pressman, R. S. (2019). CS605-Software Engineering Practitioner’s Approach. https://doi.org/10.1017/CBO978110745324.004

Qamar, N., & Malik, A. A. (2019). Evaluating the impact of pair testing on team productivity and test case quality – A controlled experiment. Pakistan Journal of Engineering and Applied Sciences, 25(7), 80–88. https://journal.uet.edu.pk/ojs_old/index.php/pjeas/article/view/1766/328

Quadri, S. M. K., & Farooq, S. U. (2010). Software testing – goals, principles, and limitations. International Journal of Computer Applications, 6(9), 7–10. https://doi.org/10.5120/1343-1648

Rajaram, H. K., Loane, J., Macmahon, S. T., & Caffery, F. M. (2021). Expert review of taxonomy based testing: A testing framework for medical device software. The 16th International Conference on Evaluation Of Novel Approaches to Software Engineering (ENASE 2021), Enase, 331–339. https://doi.org/10.5220/0010458503310339

Rajasekhar, P., & Shafi, R. M. (2014). Agile software development and testing: Approach and challenges in advanced distributed systems. Global Journal of Computer Science and Technology, 14(1), http://computerresearch.org/index.php/computer/article/view/38

Rava, M., & Doangdej, A. P. D. J. (2014). Measuring test case reusability based on simplicity and Independence. ACIT 2014, 66–71. https://doi.org/10.3850/978-981-07-8859-9_15

Rogers, M. R., & Lopez, E. C. (2002). Identifying critical cross-cultural school psychology competencies. Journal of School Psychology, 40(2), 115–141. https://doi.org/10.1016/S0022-4405(02)00093-6

Romil, R., Sarkar, S., Omar, M., & Mahmood, M. (2020). Automated test cases and test data generation for dynamic structural testing in automatic programming assessment using MC/DC. International Journal on Advanced Science, Engineering and Information Technology, 10(1), 120. https://doi.org/10.18517/ijaseit.10.10166

Salmanah, S., Gates, A., Roach, S., & Mondragon, O. (2005). Verifying pattern-generated LTL formulas: A case study. International SPIN Workshop on Model Checking of Software, 200–220.

Sanoo, H., Affifi, W., & Darwish, N. (2016). The goal questions metrics for Agile business intelligence. Egyptian Computer Science Journal, 40(2), 24–42.

Scheidbauer, M. (2012). Testing web applications (version 1.1). Northeastern University, CCIS.

Setiawan, N., Ferdiona, R., & Hartanto, R. (2020). Test case understandability model. IEEE Access, 8, 169036–169046. https://doi.org/10.1109/ACCESS.2020.3022876

Shamshiri, S., Rojas, J. M., Galeotti, J. P., Walkinshaw, N., & Fraser, G. (2019). How do automatically generated unit tests influence software maintenance? 2018 IEEE 11th international conference on software testing, verification and validation (ICST), 250–261.

Shrivastava, D. P., & Jain, R. C. (2010). Metrics for test case design in test driven development. International Journal of Computer Theory and Engineering, 2(6), 952–956. https://doi.org/10.7763/IJCTE.2010.V2.269
Sundmark, D., Pettersson, A., & Thane, H. (2005). Regression testing of multi-tasking real-time systems: A problem statement. ACM SIGBED Review, 2(2), 31–34. https://doi.org/10.1145/1121788.1121798

Tran, H. K. V., Ali, N. B., Börstler, J., & Unterkalmsteiner, M. (2019). Test-case quality-understanding practitioners’ perspectives. International Conference on Product-Focused Software Process Improvement, 37–52.

Tran, H. K. V., Unterkalmsteiner, M., Börstler, J., & Bin Ali, N. (2021). Assessing test artifact quality—A tertiary study. Information and Software Technology, 106620, 1–27. https://doi.org/10.1016/j.infsof.2021.106620

Tudjarova, S., Chorbev, I., & Joksimoski, B. (2017). Software quality metrics while using different development methodologies. International Conference on ICT Innovations, 240–250.

Waltz, C. F., Strickland, O. L., & Lenz, E. R. (2010). Measurement in nursing and health research. Springer publishing company.

Yadav, H. B., & Yadav, D. K. (2015). A fuzzy logic based approach for phase-wise software defects prediction using software metrics. Information and Software Technology, 63, 44–57. https://doi.org/10.1016/j.infsof.2015.03.001

Yamaura, T. (1998). How to design practical test cases. IEEE Software, 15(6), 30–36. https://doi.org/10.1109/52.730835

Yan, M., Xia, X., Zhang, X., Xu, L., Yang, D., & Li, S. (2019). Software quality assessment model: A systematic mapping study. Science China Information Sciences, 62(9), 1–18.

Yu, J. (2018). Design and application on agile software exploratory testing model. 2018 2nd IEEE Advanced Information Management, Communicates, Electronic and Automation Control Conference (IMCEC), 2082–2088.

Zander-Nowicka, J., Mosterman, P. J., & Schieferdecker, I. (2008). Quality of test specification by application of patterns. PLoP08 - 15th Conference on Pattern Languages of Programs, Proceedings, January. https://doi.org/10.1145/1753196.1753200

Zeiss, B., Vega, D., Schieferdecker, I. K., Neukirchen, H., & Grabowski, J. (2007). Applying the ISO 9126 quality model to test specifications – exemplified for ttcn-3 test specifications. Software Engineering, 15(6), 231–242.

Zhu, H. (1999). Axiomatic assessment of control flow-based software test adequacy criteria. Software Engineering Journal, 10(5), 194–204. https://doi.org/10.1049/sej:19990025
