

Article

A Survey on Factors Preventing the Adoption of Automated Software Testing: A Principal Component Analysis Approach

George Murazvu ^{1,2}, Simon Parkinson ^{2,*} , Saad Khan ² , Na Liu ³  and Gary Allen ² 

¹ Axia Digital, Unit 57, Batley Business Park, Batley WF17 6ER, UK; u1766085@pgr.hud.ac.uk

² Department of Computer Science, University of Huddersfield, Huddersfield HD1 3DH, UK; saad.khan@hud.ac.uk (S.K.); g.allen@hud.ac.uk (G.A.)

³ Department of Logistics, Marketing, Hospitality and Analytics, University of Huddersfield, Huddersfield HD1 3DH, UK; n.liu2@hud.ac.uk

* Correspondence: s.parkinson@hud.ac.uk; Tel.: +44-1484-256244

Abstract: Automated software testing is a crucial yet resource-intensive aspect of software development. This burden on resources affects widespread adoption, with expertise and cost being the primary challenges preventing adoption. This paper focuses on automated testing driven by manually created test cases, acknowledging its advantages while critically analysing its implications across various development stages that are affecting its adoption. Additionally, it analyses the differences in perception between those in nontechnical and technical roles, where nontechnical roles (e.g., management) predominantly strive to reduce costs and delivery time, whereas technical roles are often driven by quality and completeness. This study investigates the difference in attitudes toward automated testing (AtAT), specifically focusing on why it is not adopted. This article presents a survey conducted among software industry professionals that spans various roles to determine common trends and draw conclusions. A two-stage approach is presented, comprising a comprehensive descriptive analysis and the use of Principal Component Analysis. In total, 81 participants received a series of 22 questions, and their responses were compared against job role types and experience levels. In summary, six key findings are presented that cover expertise, time, cost, tools and techniques, utilisation, organisation, and capacity.

Keywords: software testing; automated testing; attitudes; principal component analysis



Citation: Murazvu, G.; Parkinson, S.; Khan, S.; Liu, N.; Allen, G. A Survey on Factors Preventing the Adoption of Automated Software Testing: A Principal Component Analysis Approach. *Software* **2024**, *3*, 1–27. <https://doi.org/10.3390/software3010001>

Academic Editor: Andreas L. Symeonidis

Received: 28 October 2023

Revised: 23 November 2023

Accepted: 25 December 2023

Published: 2 January 2024



Copyright: © 2024 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<https://creativecommons.org/licenses/by/4.0/>).

1. Introduction

The development of computer science and software engineering and the increasing use of artificial intelligence and data mining technologies have led to the development of a wide range of applications that are critical to operations in business, healthcare, and education. Software development is a complex and expensive process, prone to errors and subsequent failure to meet user requirements [1]. Organisations, therefore, invest significant resources to ensure that software products are tested against set criteria, ensuring that they are of the best quality before being released to their clients and users [2]. Traditionally, testing has been a manual process, involving humans executing applications and comparing their behaviour against certain benchmarks. However, advances in technology and the constant desire to improve quality have introduced and increased the use of automated testing, which uses computer algorithms to detect bugs in software applications [3]. Automated testing can generally be categorised into two types: those in which automated testing tools write and use manual test cases and frameworks in which the testing tools automatically generate test cases. In this study, we focus on the first type in which test cases are created manually. The phrase automated testing is used throughout the rest of the paper to refer to instances of automated testing that involve the manual creation and automated use of test cases.

A key aspect of all software development processes is that they each have testing phases at different points in the development cycle. For example, the Waterfall development process has a distinct test phase after development has taken place [4]. Another example is Agile, which is an iterative development process and has repetitive testing phases [5]. Although there are processes involving iterative and concurrent testing, many development processes assume users can specify a finished set of requirements in advance, ignoring the fact that they develop as the project progresses and change depending on the client's circumstances. Manual testing and correction of errors, as well as the integration of changes, is feasible in small projects, as the code size is easy to manage. However, as client requirements change or more requirements are added, the projects grow in complexity, yielding more lines of code and a higher probability of software faults occurring (commonly named bugs). This results in the need for an increased frequency of manual software testing. Consequently, there has been a shift to more flexible methodologies that combine testing with the completion of each phase to identify software problems before progressing to the next phase.

Automated software testing has many well-established benefits [6,7]; however, several organisations are still not using automation techniques due to the requirement of knowledge, legacy challenges, and reluctance to change [8,9]. The results of the 2018 SmartBear State of Testing Report survey on test automation identified that automation is not yet as common as organisations desire [10]. There are still many factors that hinder the update and use, such as challenges in acquiring and maintaining expertise, cost, and the use of the correct testing tools and frameworks. Although previous studies present the reasons why automated testing may not be used, there is an absence of literature focusing on different job roles and experiences and how they relate to factors preventing the adoption of automated testing. There is also ongoing research and debate among academics and professionals as to the merits of automated testing over traditional manual testing methods [9,11–14]. This research paper presents an empirical study to gain an understanding of the different attitudes of employees working in the software industry. The particular focus of this research is to understand whether there are common patterns surrounding different roles and levels of experience. Furthermore, this research aims to identify common reasons why automation is not being used.

At the end of this research, the following question will be answered: *Do common themes emerge when investigating opinions on why automated testing is not used, with the focus being on the job role and level of experience?* To answer this research question, a twenty-two-question survey has been created to collect attitudes toward automated testing (AtAT) from employees working in the software testing industry. The data are then thoroughly analysed by using quantitative techniques to determine key patterns and themes.

This paper is structured as follows. Section 2 presents and discusses existing work, grounding this study in the relevant literature. Section 3 describes and justifies the process adopted in this paper, which includes using a two-stage analysis approach. This section also presents and discusses the results of the study in detail, identifying common themes relevant to the objective of this study. Section 4 provides a summary of key findings and discusses how these findings motivate future work. Finally, in Section 5, a conclusion of the work is provided. The full set of participant responses is available in Appendix A.

2. Related Work

The purpose of this paper is to identify the attitudes of those working in software testing. This section surveys academic works that tackle this question, comparing any existing approaches and methodologies. In one recent study, the authors defined manual testing as a procedure to test the product to find software bugs [2]. Software is erroneous if it deviates from the system requirements and/or implements any requirement incorrectly. Taipale et al. agree by stating that manual software testing is the procedure of physically testing software for imperfections, and it requires a tester to assume the job of an end-user, whereby they use the application's features to ensure correct functionality [11]. This

view is shared by many researchers [9,13]. There are several types of software testing that target different objectives, such as effectiveness, efficiency, user satisfaction, completeness, defect types, etc. For example, two recent works focus on testing and detecting specific memory issues in Android devices [15] and object replication in Java applications [16]. Following the different testing objectives mentioned by Hynninen et al. [17], one area that has received significant attention lately is that of testing software to examine if there are security vulnerabilities that can be exploited by an attacker [18]. In another study, the authors examine how the objective of usability testing is performed in industry [19]. Software quality is commonly discussed as a testing objective, with recent work discussing the role of Artificial Intelligence [20]. In other work, the authors propose a methodological framework as a set of guidelines and checklists on the type of testing that should be applied to achieve a certain objective based on a given case study [21].

Although software testing usually identifies errors and hence reduces associated costs and maintains quality, evidence suggests that its proportion of the aggregated costs of total development is high. A research study has determined that it contributes between 40% and 80% of total development costs [22]. This could be regarded as contrary to business strategy for profit maximisation, and hence software manufacturers are increasingly looking for ways to reduce their testing costs in order to reduce their overall development costs. In a recent report, process efficiency is described as the ability of a process to produce the desired result with the optimal number of resources [23]. Although automated testing is commonly agreed upon to help identify software faults faster compared to manual testing, the literature questions whether it is capable of significantly reducing the overall costs of a project [24]. Automated software testing is defined as a process where software testing frameworks (such as the Selenium web testing suite [25]) are utilised to conduct prescribed software tests to confirm whether all functionality is working appropriately [26].

2.1. Requirement for Automated Software Testing

There is strong evidence to suggest that expenditure can be reduced using test automation. A report by Infosys [23] states that the manual testing of product features and performance is an expensive, lengthy, and tedious task. A recent survey [27] claims that the cost of software testing is between 30% and 50% of the entire budget, and there is an undeniable requirement for testing methods that can decrease the duration required to guarantee software quality and reliability. In another work, the authors discovered a set of factors that influence the cost of test automation, all of which provide positive outcomes on cost, quality, and time to market [24]. Another research study presented an experiment on an automated test generation tool and proposed a methodology named ‘TestDescriber’, which creates comprehensive documentation for each individual test, thereby improving and aiding the reduction of expert knowledge required to perform the tests [28]. This is an extension of previous work [29] that developed a toolkit to facilitate the automatic generation of test data for structural testing cases. In relation to the financial impacts of automated testing, a study discovered that the cost increases from 1:5 (from requirements to after release) for simple systems to as high as 1:100 for complex systems [30]. This statement confirms that once the bug is found in production, it will cost more to rectify, as the system might need to be taken out of operation in order for the bug to be fixed, which will result in the company losing revenue or even customers migrating to competitors because of lack of confidence with their software systems. A similar study confirms that the longer a software fault is left undetected, the more expensive it will be to fix once it is discovered [31].

A recent research work examined the relative proficiencies of both random and organised methods in automated software testing and identified that proficiency is an imperative property of software testing, conceivably significantly more essential than adequacy [32]. Test automation can provide benefits in many ways, such as test reusability, repeatability, test inclusion, and exertion spared in test executions. Another work added that since complex software faults exist even in basic software systems, engineers are searching for automated systems to identify software faults, resulting in greater trust and accuracy [33].

A similar study states that automated testing is a productive method to gain trust in the accuracy of the software [34]. This observation is well argued and is based on the premise that automated software testing removes the element of human error and is faster to run regression tests, which can take days if they are to be performed manually. Furthermore, another article claimed that when comparing automated software testing versus manual software testing, the impacts and advantages of automated testing are provided in the long term when compared to manual testing [35]. This is because an automated testing tool can consider and process all factors holistically, in an efficient manner, as compared to manual testing.

A research study examined different methods of software testing and concluded that performing manual testing is wasteful and error prone; using automated tests is efficient in reducing software release time [36]. The experiment was based on a mathematical procedure with the intention of increasing the chances of having a resource-effective test automation process. Another paper investigated techniques for enhancing the effectiveness of software test automation. This point is supported by the declaration that automated testing frees testing staff to perform other testing duties [37]. This paper also explored the challenges and best practices related to quality within software development and determined that completing software testing can reduce financial expenditure by catching issues before they progress too far through the product development process.

Another paper reports that the main issue for a tester and/or organisation that wants to automate their software testing process is how much the testing tools cost [38]. Furthermore, the concern is whether it will satisfy the testing requirements. Open-source testing tools are available and free to use, which is seen as a positive aspect and helps organisations automate software testing. Another research study conducted an experiment to investigate the benefits of automated testing techniques using the open-source Ball Aviation Universe testing framework. It concluded that automated testing yields numerous advantages, such as mitigation against client input errors, faster execution times, and decreased client oversight during execution [39].

2.2. Current Limitations of Automated Testing

A study states that during the investigation of the current situation and possible improvements in software test automation, it was observed that the main advantages of test automation were quality improvement, the likelihood of executing more tests in less time, and the familiar reuse of testware [40]. However, other work identified that when investigating the present condition of test automation in software testing, by surveying the perspectives and perceptions of supervisors, testers, and developers in every company, it was concluded that the biggest burdens were the expenses related to implementing test automation, particularly under unique altered conditions [11]. Another paper conducted an experiment using the AutoTest tool, which is a fully automated testing framework running on the Linux system. After combining automated and manual testing, it was realised that software can be tested manually or automatically, and these two methodologies can complement each other [41].

Similarly, another experimental framework is presented to compare testing procedures based on efficiency, effectiveness, and applicability [22]. It employed 70 distinct test design techniques and concluded that automated testing cannot be applied in all cases due to a lack of ability to determine issues and/or increased difficulty in implementation. This agrees with the observations and lessons learnt from automated testing [6] that the use of automated test tools does not improve fault detection when compared to manual testing. Furthermore, it was found that 80% of professionals disagreed with the suggestion that automation testing would serve as a complete replacement for manual testing. This issue seems to be well known, as another paper determined that automated tests found only 26% (on average) of the faults [40]. Furthermore, they state that when an automated test suite has been configured and integrated, it is usually reused in future tests. This makes testing substantially less likely to uncover defects in the product during the next iteration.

Regarding open-source software testing tools, a paper investigated several such tools and concluded that they are not maintained regularly and are difficult to use [42]. In addition, organisations are still likely to use commercial tools due to the level of support available, which can help them fully use the technology. Therefore, due to the aforementioned reasons, automated software testing is not used in some organisations.

The existing literature highlights that there is a known gap between academic and practitioner opinions on automated software testing, and there is a need to close the gap by exploring attitudes about the benefits and restrictions of test automation [43]. However, the appreciation for test automation is unbalanced, as the achievement rate is low and the impediments are always high at the beginning for acquiring the resources to set up automation testing and training tools. Moreover, the automated tests are not well suited for every organisation and are varied in terms of accuracy, applicability, and usefulness factors.

2.3. Survey-Based Research

There are many examples of recent research that involves surveying practitioners in the software industry. A recent survey was conducted to determine the importance of automated bug report management [44]. This research consulted 327 practitioners to gain their insights into automated bug report management techniques. Their study concluded that practitioners value automated bug report management techniques, but many recommendations were identified. In another recent survey, 3000 industry professionals were invited to rate the relevance of research published at leading conferences [45]. Research was carried out to understand how practitioners perceive software engineering research, helping conference organisers and academics understand software engineering research priorities and what elements of their research are favourably perceived, and thus have a stronger end-user impact. Another recent research study focused on acquiring the perception of productivity by software developers [46]. In this survey, the authors consulted 379 software developers, eliciting themes around tasks, activities, and workflow. The authors have also conducted literature-based surveys on impediments to software test automation, identifying the benefits and challenges; however, the authors state that empirical work is needed for further understanding [9], which further motivates the need for the research presented in this article.

3. Materials and Methods

In this work, we used a two-stage analysis to address the research question. In the first phase, a comprehensive yet basic analysis is performed to provide the foundation for understanding individuals' attitudes towards the reasons why automated testing is not used and also to illuminate patterns specific to individuals performing different roles and of different experiences.

Next, we conduct a principal component factor analysis (principal component extraction). This is a standardised and widely used approach, which provides the opportunity to further examine the relationships between the participants' opinions on automated testing as a whole while looking for clustering of certain variables [47]. In particular, we examine the dimensionality of individual responses to investigate whether or not automated testing attitudes comprise a distinct attitudinal dimension. Principal component factor analysis is a standard and widely used technique for data analysis [48].

3.1. Questions and Process

To measure attitudes toward automated testing (AtAT), a scale was constructed based on twenty items asking respondents about their broad feelings relating to automated software functionality, as well as about its adoption. The questionnaire was created in a way that develops a comprehensive analysis of common reasons why automated software testing is not used. To understand this and what facilitates the development of technological mechanisms, practitioners' attitudes and concerns must first be investigated. The questionnaire is mostly derived with the help of existing frameworks and methodologies.

The survey was distributed through professional and social media channels to acquire participants. Groups such as the following will be targeted: Quality Assurance (QA) testers, Software Developers in Test (SDIT), Software Testing Managers, and Automation Engineers. The questionnaire that has been designed consists of 22 questions.

The questions are grouped into the constructs of biographic, time, cost, tools and techniques, utilisation, organisation, and capability. We selected these construct themes as they were repeatedly presented in related research and represent a natural divide between the individual, the technology, and the environment within which both operate. Table 1 provides the mapping of the questions to each construct, including a citation to the literature presented in related work that motivates their inclusion in the questionnaire. A question number is provided, as is used in later discussions. The numbering is provided in the order that they were asked to the participants, and it is evident that the questions for each construct are diversely distributed. The purpose of this is to extract more information from participants on their AtAT to enable a stronger analysis. As this questionnaire aims to deduce the reasons for not accepting automated testing, all items are negatively worded. There is also an open-ended section for the participant to provide further comments. Note that the questions are not asked in a grouped order to try to introduce variation within the questions being asked, making the participant revisit the theme after changing to a different theme. For each question, the participant will be presented with a statement with which they can either agree or disagree. The participant will receive responses based on the Likert scale [49], which are either strongly disagree, disagree, neutral, agree, or strongly agree. Furthermore, free text input is made possible at two points in the survey to acquire additional information. The purpose of these inputs is to acquire comments from the participant that might rationalise their answer or provide further information. The first appears approximately halfway through the survey at Question 10 and the second at the end of the survey at Question 22.

Table 1. Questions and construct.

Construct	Questions	Sources
Biographic	q1 What is your job title?	[32]
	q2 How many years of experience in the IT sector do you have?	
Time	q4 Individuals not having enough time prevents the use of test automation.	[24,36,40]
	q15 Automated testing techniques are time-consuming to learn.	
Cost	q11 Commercial tools are too expensive, which prevents their use.	[22,27,30,31,38]
	q19 Expensive to generate test cases/test scripts.	
	q20 They require high maintenance costs for test cases, test scripts and test data.	
Tools and Techniques	q6 Not having the right automation tools and frameworks is preventing use.	[6,39,44]
	q16 Automated testing tools and techniques lack the necessary functionality.	
	q17 They are not reliable enough to make them suitable for use.	
	q18 They lack support for testing non-functional requirements (usability, safety, security, etc.).	
	q21 Automated testing tools and techniques change too often, introducing problems that need fixing.	

Table 1. *Cont.*

Construct	Questions	Sources
Utilisation	q5 Difficulties in preparing test data and environments prevent their use.	[34,35,37]
	q7 Difficulty in integrating different automation tools/frameworks together is preventing their use.	
	q8 Requirements change too often in software projects resulting in them being too time-consuming when required to quickly react to change.	
	q12 Open-source tools are not easy to use.	
	q22 Difficult to reuse test scripts and data across stages of testing.	
Organisation and Capability	q3 Lack of skilled resources prevents automated testing from being used.	[11,28]
	q9 Not realising and understanding the benefits of test automation is preventing their use.	
	q10 A lack of support from senior management is preventing their use.	
	q13 Test automation tools require a high level of expertise, which is often not available.	
	q14 Automated testing requires strong programming skills.	

As all questions were multiple choice, the responses provided in Appendix A are their numerical versions (strongly agree = 5, agree = 4, neutral = 3, disagree = 2, strongly disagree = 1). Furthermore, the graphs provided in Figures 3 and 4 use a character abbreviation (strongly agree = SA, agree = A, neutral = N, disagree = D, strongly disagree = SD). Because all items are negatively worded, score reverses were not needed.

The survey was created as a digital survey and distributed through special interest groups. More specifically, we used Google Forms to create and host special interest groups in software engineering and testing on LinkedIn. The survey uses a suitable convenient sampling approach [50], where the survey is open to responses from all professionals working in software testing, regardless of whether they are employed or have experience working within an organisation where test automation is applied. This is done to ensure that responses are captured from respondents with different experiences to represent the diversity within the software test community. This is important, as all individuals working in the software lifecycle are in some way influential to the adoption of automated software testing within their organisation. It is also worth noting that we do not ask participants to disclose information regarding organisations they work within. The purpose is to maintain anonymity and follow the survey style used in practitioner-based surveys in the discipline of software engineering [44–46]. The questionnaire took approximately 10 min to complete.

3.2. Participants

Figure 1 illustrates the experience of the participants. Years of experience are used to measure how long a participant has been involved in automated testing, which is also a measure used in other academic work [51]. It is important to make the distinction that the authors are not assuming that years of experience relate to an individual's skill level; rather, an assumption is made that they will have had more interaction with automated testing tools and techniques, therefore forming stronger attitudes. The duration of the experience ranges from 0.5 to 33 years, and as evident in the table, a good variation was surveyed. However, the majority of participants are in the range between 1 and 20 years of age. This is of significant importance as it demonstrates that the survey will not be overly biased toward IT professionals with short or long experience. Table 2 illustrates the variation of roles and the number of respondents. Note that the role title was entered by the user and resulted in a wide variation of roles. It is worth noting that the roles have been placed in themes for ease of comparison. The themes adopted are the same as those of the State of

Testing Report (2018), as discussed in Section 1. In the table, it is evident that the majority of job role themes are Quality Assurance, Software Testing, and then senior versions of each role. Outside of technical roles, there are three Chief Executive Officers, four consultants, seven managers, and one student. Although it can be seen that, in general, the majority of respondents are performing more technical roles, the 15 nontechnical responses account for 18% of the total responses and are not insignificant.

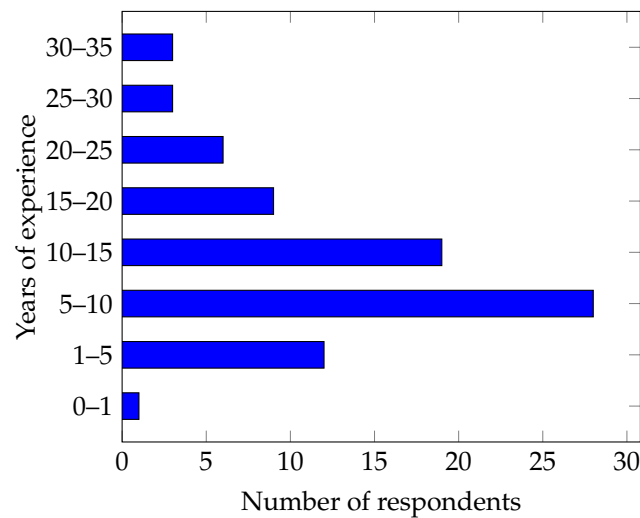


Figure 1. Respondent experience in the IT sector.

Table 2. Participant roles in the IT sector.

Job Role	# of Participants
CEO	3
Consultant	3
Senior Consultant	1
Manager	7
Student	1
QA	7
Senior QA	8
Tester/Engineer/Analyst/Architect/Automation	27
Senior Tester/Engineer/Analyst/Architect/Automation	24
Total	81

3.3. Results: Stage 1

In this section, the responses from the questionnaire are analysed and discussed. Figure 2 provides the numbers of responses for each available response (strongly disagree, disagree, neutral, agree, strongly agree). All response data are available from the authors upon request. Figure 3 provides bar charts for each response in relation to the response choices whilst also showing the response split between different job roles, as provided in Table 2. Furthermore, Figure 4 provides information on how many years of experience the participants have with the responses. The discussion is grouped according to the constructs presented in Table 1, except for the biographic, which is discussed in Section 3.2.

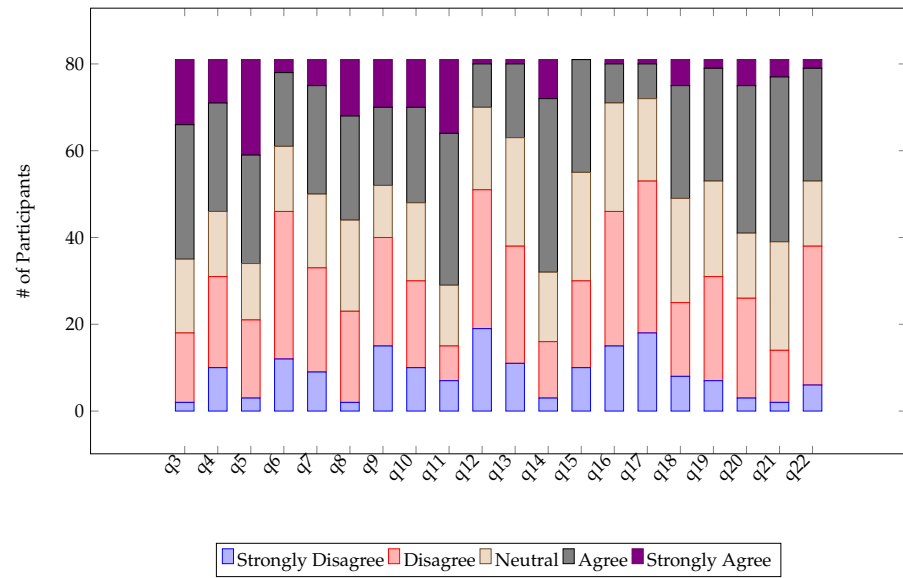


Figure 2. Response to questions.

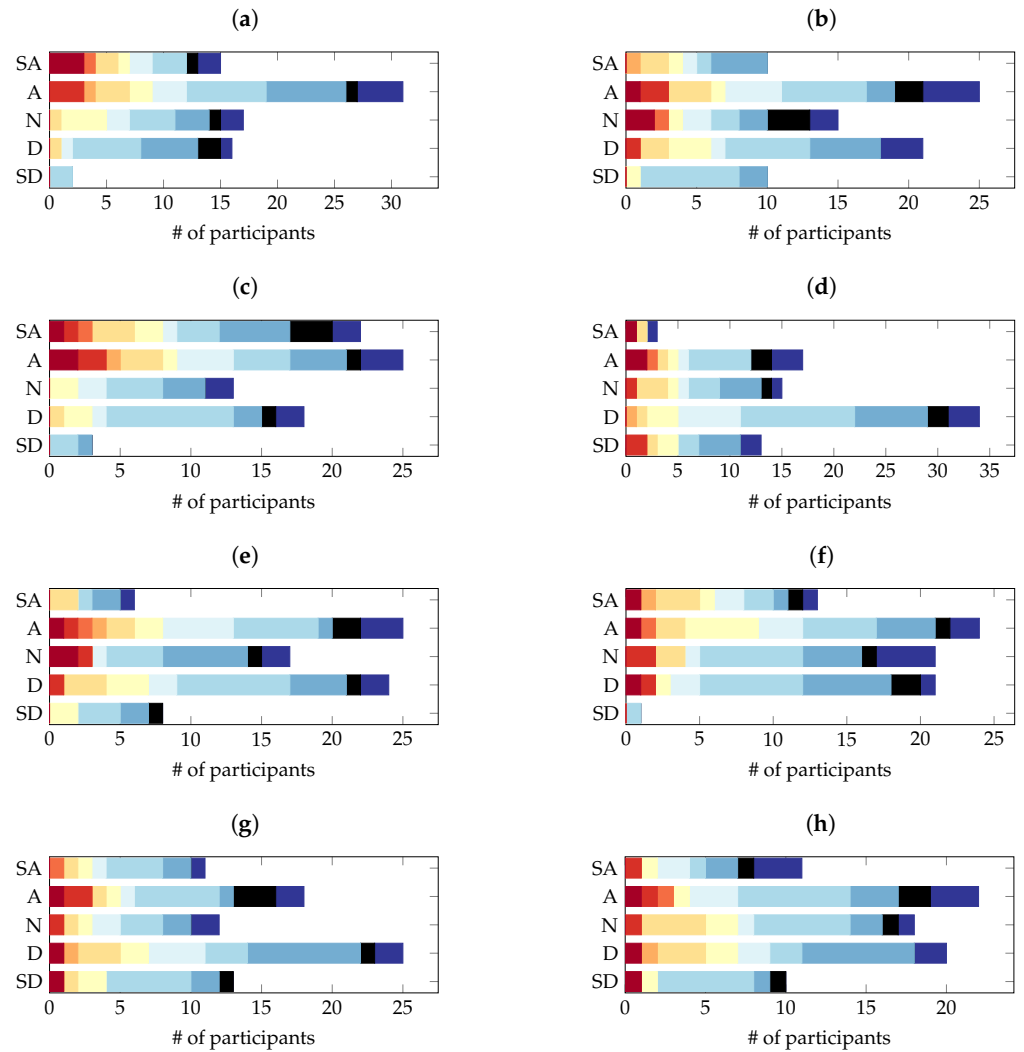


Figure 3. Cont.

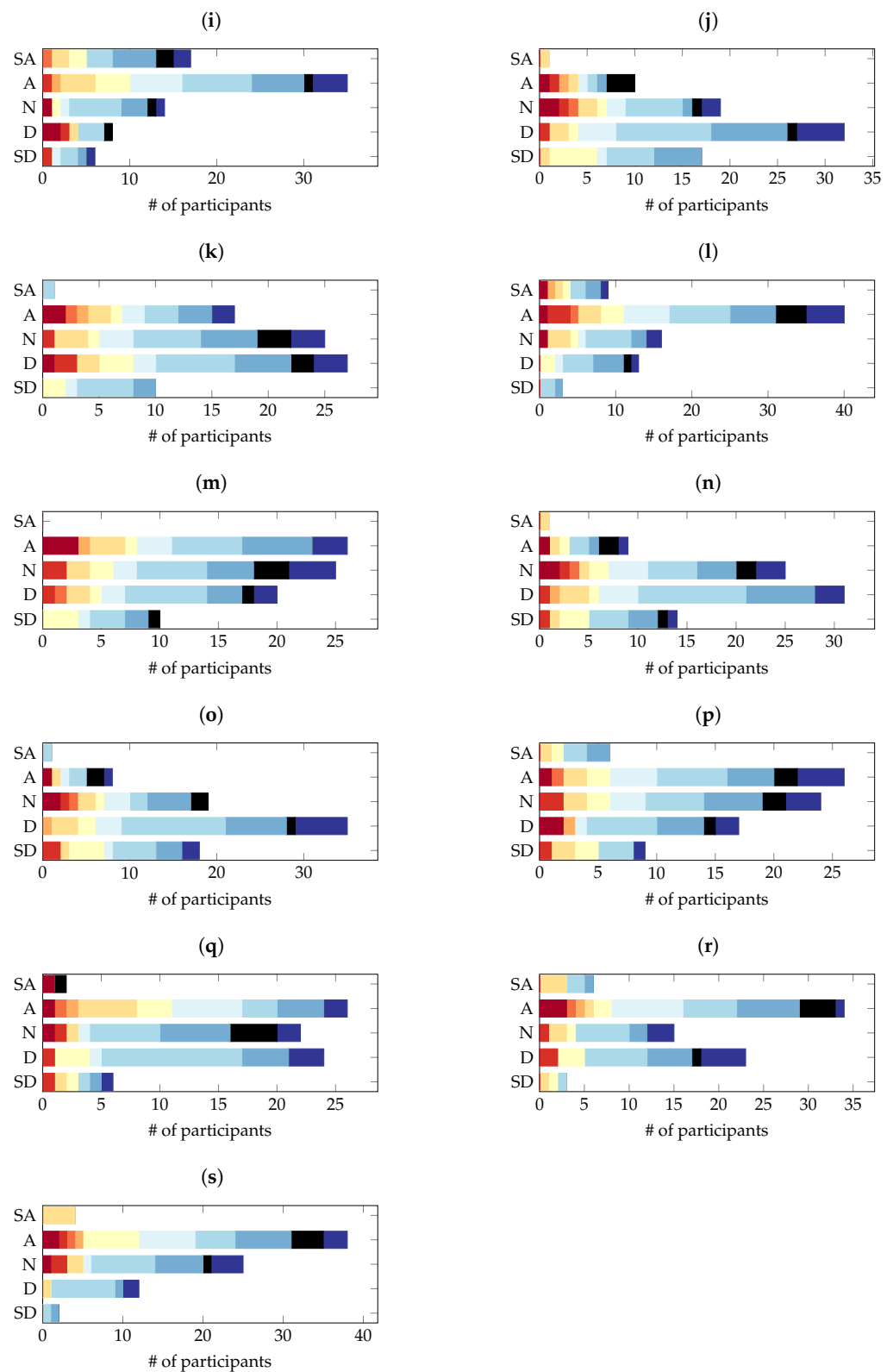


Figure 3. Cont.

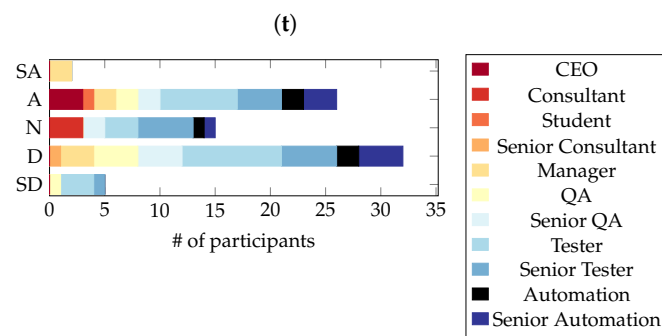


Figure 3. Responses for each survey by question, illustrating the distribution of answers based on job role. (a) Q3 Lack of skilled resources prevents automated testing from being used; (b) Q4 Individuals not having enough time prevents the use of test automation; (c) Q5 Difficulties in preparing test data and environments prevents their use; (d) Q6 Not having the right automation tools and frameworks is preventing use; (e) Q7 Difficult to integrate different automation tools/frameworks together is preventing their use; (f) Q8 Requirements change too often in software projects resulting in them being too time-consuming when required to quickly react to change; (g) Q9 Not realising and understanding the benefits of test automation is preventing their use; (h) Q10 A lack of support from senior management is preventing their use; (i) Q11 Commercial tools are too expensive, which prevents their use; (j) Q12 Open-source tools are not easy to use; (k) Q13 Test automation tools require a high level of expertise, which is often not available; (l) Q14 Automated testing requires strong programming skills; (m) Q15 Automated testing techniques are time-consuming to learn; (n) Q16 Automated testing tools and techniques lack the necessary functionality; (o) Q17 They are not reliable enough to make them suitable for use; (p) Q18 They lack support for testing non-functional requirements (usability, safety, security, etc.); (q) Q19 Expensive to generate test cases/test scripts; (r) Q20 They require high maintenance costs for test cases, test scripts and test data; (s) Q21 Automated testing tools and techniques change too often, introducing problems that need fixing; (t) Q22 Difficult to reuse test scripts and data across stages of testing.

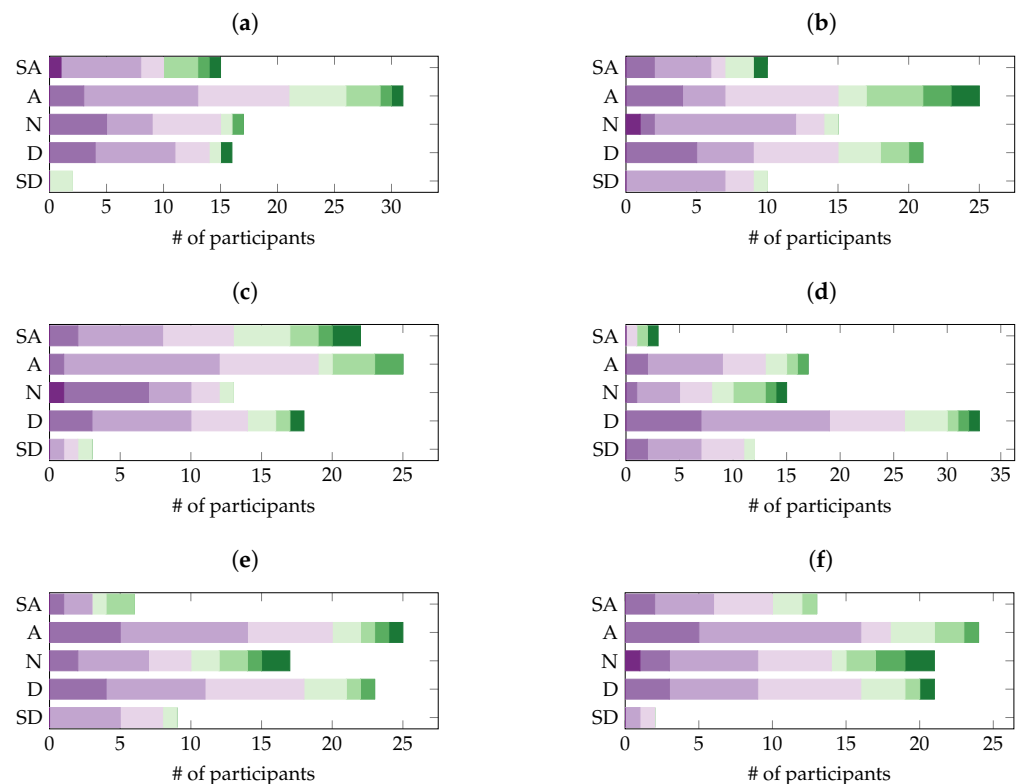


Figure 4. Cont.

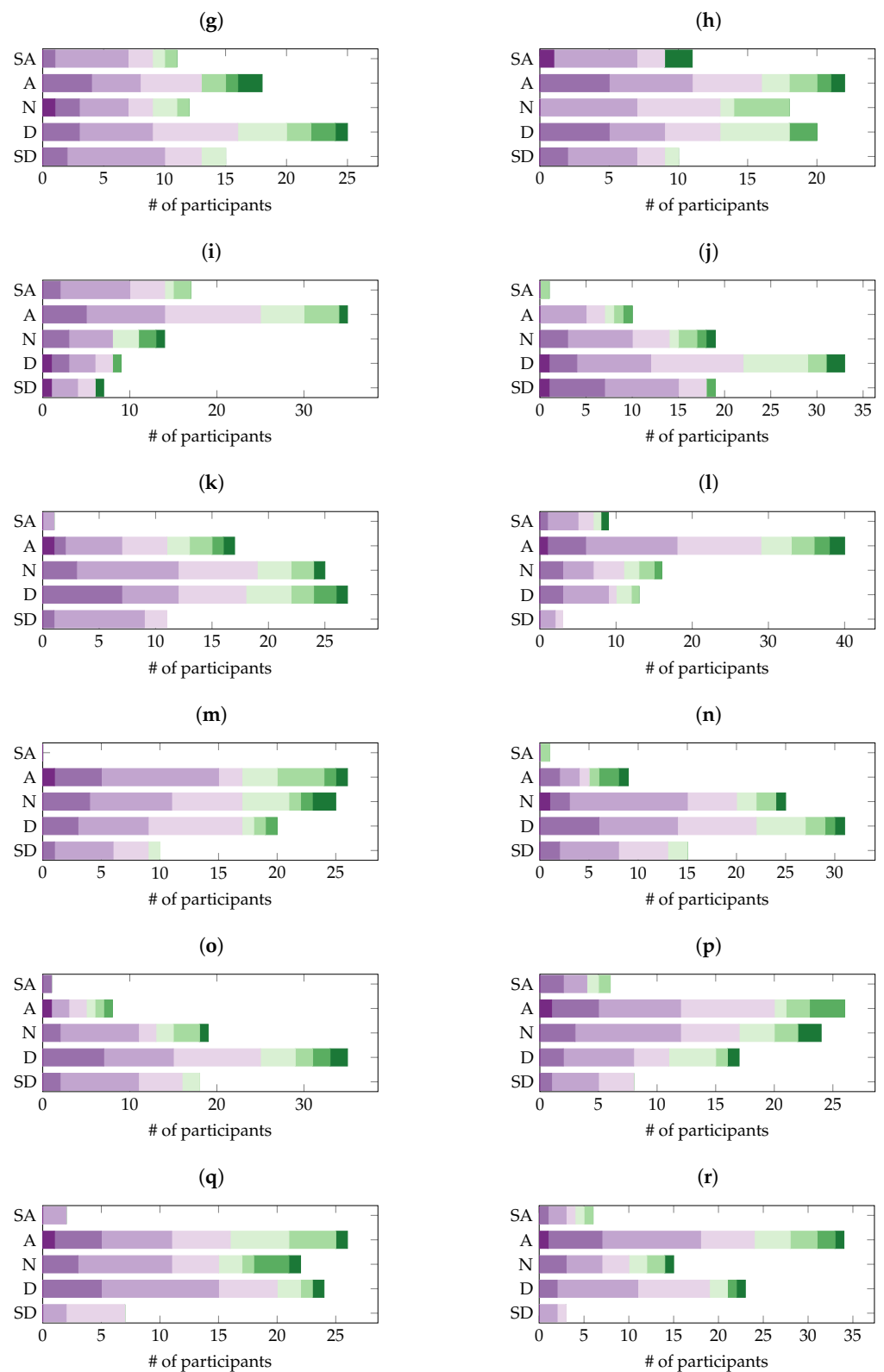


Figure 4. Cont.

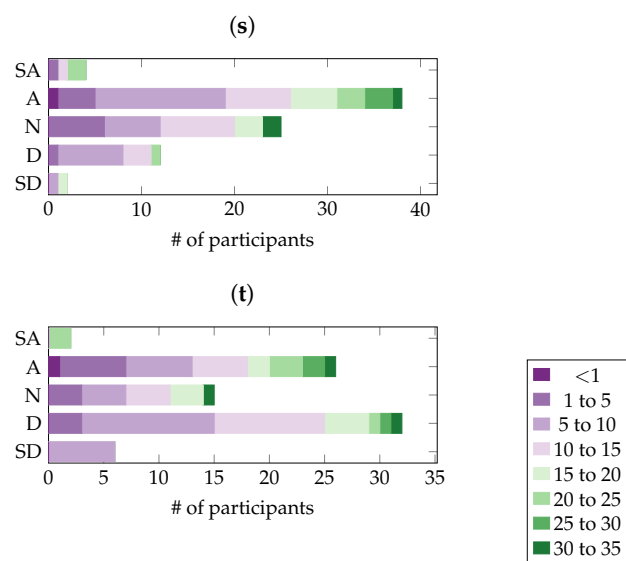


Figure 4. Responses for each survey by question, illustrating the distribution of answers based on number of years of experience. (a) Q3 Lack of skilled resources prevents automated testing from being used; (b) Q4 Individuals not having enough time prevents the use of test automation; (c) Q5 Difficulties in preparing test data and environments prevents their use; (d) Q6 Not having the right automation tools and frameworks is preventing use; (e) Q7 Difficult to integrate different automation tools/frameworks together is preventing their use; (f) Q8 Requirements change too often in software projects resulting in them being too time-consuming when required to quickly react to change; (g) Q9 Not realising and understanding the benefits of test automation is preventing their use; (h) Q10 A lack of support from senior management is preventing their use; (i) Q11 Commercial tools are too expensive, which prevents their use; (j) Q12 Open-source tools are not easy to use; (k) Q13 Test automation tools require a high level of expertise, which is often not available; (l) Q14 Automated testing requires strong programming skills; (m) Q15 Automated testing techniques are time-consuming to learn; (n) Q16 Automated testing tools and techniques lack the necessary functionality; (o) Q17 They are not reliable enough to make them suitable for use; (p) Q18 They lack support for testing non-functional requirements (usability, safety, security, etc.); (q) Q19 Expensive to generate test cases/test scripts; (r) Q20 They require high maintenance costs for test cases, test scripts and test data; (s) Q21 Automated testing tools and techniques change too often, introducing problems that need fixing.; (t) Q22 Difficult to reuse test scripts and data across stages of testing.

3.3.1. Time

Question 4 asks the participants if they believe that an individual not having enough time prevents the use of automated testing. The response to the question is well balanced with only a small majority stating that they agree. As demonstrated in Figure 3b, the distribution of the job roles among the responses is balanced, with both technical and nontechnical roles agreeing and disagreeing. One identified trend is that participants who strongly disagree have identified themselves as performing a technical role, with only two of the ten responses occupying a senior role. This indicates that more junior roles disagree more strongly with the statement presented. It can also be established from Figure 4b that there is an even distribution of years of experience between the responses.

Question 15 then asked the participants if they think automated testing techniques are time consuming to learn. The responses to this question are also quite evenly distributed. The low percentage of participants who strongly agree with this statement results in an average between neutral and disagreement. Figure 3m demonstrates the distribution of job roles amongst response categories, and it is worth noting that, in general, nontechnical roles are responding more closely with agree and neutral replies, which could perhaps be down to their lack of experience with the technology. All CEO responses are in the agreed category.

Figure 4m demonstrates the years of experience for each response category. Interestingly, there is an even distribution apart from agreement, whereby there is the highest quantity of participants with the lowest number of years of experience, which could demonstrate that those with a lower amount of experience could believe that automated testing takes more time to learn, which would most likely originate from the fact that they will have more to learn during earlier years of employment.

3.3.2. Cost

Question 11 asked whether the participant agreed or disagreed that commercial tools are too expensive, thus preventing their use. Most of the participants agreed with this statement. Figure 3i illustrates the number of responses in relation to each job role. It is observed that most of the senior management, consultants, and QA and senior QA roles agree with the statement, whereas the CEOs are neutral or disagree. Figure 4i shows how years of experience are evenly distributed among the available responses.

Question 19 asked the participants whether they believed that automated test scripts and cases are expensive to generate. The responses to this question are balanced, with only a slight emphasis on disagreement. Figure 3q illustrates that the general trend is that managerial roles are more likely to agree with this statement. It is also worth noting from Figure 4q that the small number of responses that strongly agree and disagree have more than 5 years of experience, while the other categories have an even distribution. This could indicate that the views of experienced employees are on average neutral, with a minority having polarised views.

Question 20 asked the participants whether they agreed that automated testing requires high maintenance costs. Overall, the participants agreed with this statement, but Figure 3r illustrates that, in general, nontechnical roles are more likely to agree with this statement, which is perhaps to be expected considering their daily interaction with financial operations. There is a slight emphasis on technical staff not agreeing with this statement, which is perhaps down to their lack of involvement with the financial side of their employer's activities. Furthermore, Figure 4r shows that those strongly agreeing or disagreeing have a higher number of years of experience.

3.3.3. Tools and Techniques

Question 6 asks the participants if they believe that not having the right automation tools and frameworks is preventing use. It is evident that, in general, the majority of participants do not believe the use of automated testing is prohibited by the inability to identify and use the correct tools. Interestingly, Figure 4d illustrates that participants who are strongly agreeing have 10 to 15 and 30 to 35 years of experience. However, overall, there is an even distribution of years of experience among the responses.

Question 16 asked participants whether they believed that automated testing tools and techniques lack the necessary functionality. In general, most of the participants disagree with this statement. Figure 4n shows that there is a slight increase in the portion of responses from participants with an increased number of years of experience in the categories agree and strongly agree. This could perhaps indicate that more experienced employees have a stronger belief that current techniques and tools lack functionality, which could be down to the fact that they have in-depth experience and knowledge of missing functionality.

Question 17 asked the participants whether they believed that automated testing tools and techniques were not reliable enough, making them unsuitable for use. Respondents overwhelmingly disagreed with this statement. Figure 3o illustrates that there is a diverse distribution of job roles between each response category. It is worth noting that only one participant strongly agreed and that they are performing a technical role, which could indicate that their dissatisfaction originates from working closely with automated testing tools and techniques. Furthermore, as evident in Figure 4o, there is no relationship between years of experience and response, except for the observation that there is a higher proportion of participants with a lower number of years of experience agreeing or strongly agreeing.

This could indicate that they have not yet mastered their craft and utilised the full potential of automated tools, or even their dissatisfaction with their chosen career.

Question 18 asked the participants whether they agreed that automated testing lacks support for testing nonfunctional requirements (usability, safety, security.) The responses to this statement are close, but the majority agree with this statement. Figures 3p and 4p show that there is an even distribution between roles and length of experience within the response categories.

Question 21 asked the participants whether they agree that automated testing tools and techniques change too often, introducing problems that need fixing. Most of the responses agree with this statement. Figure 3s illustrates that nontechnical employees are more likely to agree with the statement, with only management roles submitted as strong accept. Interestingly, it also illustrates that QA and senior QA roles only responded as agreed. The majority of technical testing, engineering, and automation roles are neutral or disagree, and they are also the only roles that strongly disagree. Furthermore, Figure 4s also shows that, in general, participants with a higher number of years of experience are more likely to respond with a neutral or disagreeing response. This indicates a different point of view between nontechnical and technical roles, as well as the number of years of experience that an individual has. Experienced individuals may have gained sufficient experience in how to maintain their scripts and keep them up-to-date with new versions of testing tools.

3.3.4. Utilisation

Question 5 asks participants if they believe that difficulties in preparing test data and environments are responsible for preventing the use of automated testing. In general, most of the responses agree with this statement. Figure 3c presents the breakdown of responses versus job roles. Interestingly, the results demonstrate that nontechnical roles (CEO, consultant, management) mostly agree with this statement and there is only one response from a manager that disagrees.

Question 7 asked the participants whether they agreed with the statement that difficulties in integrating different tools/frameworks together prevent their use. A small majority were in favour of disagreeing with the statement. Figure 3e illustrates that the majority of nontechnical roles agree with this statement and the balance based on technical roles is almost even, with a slight emphasis on disagreeing with the statement. Figure 4e shows an even distribution of years of experience among responses, although the responses with a higher number of years of experience are, in balance, more in agreement than in disagreement.

Question 8 asks the participants whether they agree or not with the statement that frequent requirement changes often prevent the use of automated testing. The responses provided agree in general with the statement. In Figure 3f, it is evident that the job role distribution is mostly even with the majority of respondents operating as a(n) software tester, engineer, analyst, and test architect, whereas respondents with quality assurance roles are majority agreeing. Interestingly, nontechnical positions, such as CEOs, are either neutral or disagree with this statement, which could indicate a misalignment between both nontechnical and technical employee experiences with automated testing when it comes to the impact of changing software requirements.

Question 12 asked the participants whether they think that open-source automation tools are difficult to use, and the majority of the participants disagreed. As illustrated in Figure 3j, the number of nonsenior and technical roles is low for both agreeing and strongly agreeing. In relation to years of experience, Figure 4j illustrates that a higher number of individuals with a lower number of years of experience disagree with the statement, which could be due to the fact that those with fewer years of experience received dedicated training on the tools they are using, that is, they could be recent graduates who have received training specifically on the technology used.

Question 22 is the final question and asked participants if they agree that it is difficult to reuse test scripts and data at different stages of testing. Responses in general agree with this statement. The lower number of neutral responses indicates polarised views on this statement. From the analysis of the different roles presented in Figure 3t, it is evident that nontechnical employees are more likely to agree with the statement; however, this is a weak correlation, as some nontechnical staff disagree. In addition, technical personnel are distributed across all categories. However, only those who perform QA and technical roles strongly disagree. Figure 4t illustrates that of the users who strongly agree, all have a large number of years of experience. The different categories of experience are evenly distributed among the different response categories, apart from those that strongly disagree and have between 5 and 10 years of experience only.

3.3.5. Organisation and Capabilities

Question 3 asked the participants whether they agreed with the statement that a lack of skilled resources prevents automated testing from being adopted within an organisation. This demonstrates that the majority of the responses agree that the lack of skilled resources is a problem. Furthermore, as demonstrated in Figure 3a, most of those who agree with this statement perform nontechnical roles, while most of those who disagree perform more technical roles. This is important because it highlights the different points of view when considering whether there is a resourcing issue. In addition, Figure 4a highlights that the majority of the responses provided by participants with more than 20 years of experience agree or strongly agree. However, it is also worth noting that participants who strongly disagree are only in the 15–20 years of experience category.

Question 9 asked whether people believed that automated testing is often not used because people do not realise and understand the potential benefits. The overall trend is that a majority disagree with this statement.

Question 10 asked the participant whether they believed that lack of support from senior management was preventing their use. The results from this question are well balanced, with the number of participants agreeing with the statement being slightly higher than those disagreeing. In Figure 3h, it is evident that there is an even distribution of job roles among the responses; however, the role of consultant only appears in the neutral, agreeing, and strongly agreeing responses, whereas managers and CEOs on average disagree with the statement. The difference with consultants could be due to the fact that they are not directly employed by an organisation and provide independent observation.

Question 13 asked the participants whether they agreed or disagreed with the statement that test automation requires a high level of expertise, which is often not available. Overall, the trend is that the respondents disagree. Figure 3k illustrates how different roles selected their answers. It is evident that there is an even distribution of roles. It is therefore a fair assumption to state that only those with good technical understanding disagree with the statement. Figure 4k illustrates that the number of years of experience within each category is well distributed; however, strongly disagree has the highest average years of experience when compared to the other categories.

Question 14 asked if participants believed that automated testing requires strong programming skills. Overall, the responses strongly agreed with this statement. Figure 3l shows that there is an even distribution of roles that provide responses within each response category, and Figure 4l illustrates that there is an even distribution of years of experience within each response category.

3.4. Results: Stage 2

In this stage, the Principal Component Analysis (PCA) is performed using SPSS (version 24) on the responses (Q3–Q22). Best-practice guidance informs that ideally a minimum of five cases (participant responses) per each question is a good ‘rule of thumb’ but that a minimum of four is also sufficient [52]. However, it should be noted that this can reduce the quality of the analysis, and this section presents appropriate measures

to establish adequacy and reliability. All responses are on the same Likert scale and do not require normalisation. PCA is a statistical analysis technique that uses linear algebra techniques (specifically orthogonal transformation) to convert a data set believed to contain correlations into subsets of correlated data, known as principal components. In this process, the twenty items (questions) were used, and these comprised the final attitude scale. When performing PCA, the level of variance (known as *Cohen's alpha*) is calculated and used as a measure of how suitable the data are for identifying principal components. Nunnally and Bernstein [53] state that 0.70 is an acceptable minimum for a scale that is newly developed. In our results, the reliability of these 20 items in the sample produced a *Cronbach alpha* of 0.86 and is presented in Table 3. It is important to note that the alpha coefficient did not increase when items were eliminated. Ferketich [54] recommended that the corrected item-total correlations should range between 0.30 and 0.70 for a very good scale [55]. In our result, the 20 questions had significant corrected item-total correlations and were retained (ranging from 0.24 to 0.60). Although some of the questions provided a correlation of 0.24, values between 0.2 and 0.39 are often regarded as indicating good discrimination [56].

Table 3. Reliability statistics.

Cronbach's Alpha	Number of Items
0.860	20

Table 4 presents the results using the Kaiser–Meyer–Olkin (KMO) and Bartlett's measure of sampling adequacy. As is evident in the results, the KMO value is very close to 0.8, which is generally considered to determine whether the sample is adequate. A value below 0.6 determines that the sampling would be inadequate, but anything above 0.7 is acceptable [53,57]. Furthermore, Bartlett's test of sphericity is used to measure the correlation between the questions within a theme. In our results, we obtain a value of 553.333. The closer the value is to 0, the higher the correlation between two variables [58].

Table 4. KMO and Bartlett's test.

Test Technique	Result
Kaiser–Meyer–Olkin Measure of Sampling Adequacy	0.778
Bartlett's Test of Sphericity Approx. Chi-Square	553.333
Significance	0.000

AtAT with oblique rotation (nonorthogonal) was used to investigate the components that prevent people from adopting automated testing. Analysis of the scree plot and eigenvalues led to the extraction of two components, which together represented 39% of the variance in the data. In Table 5, the individual items and their relationship to the two factors can be seen. The table shows the item loadings for each of the two factors. We termed component one *nonsoftware factors*, which comprises items relating to finance, expertise, and time. The second component that we call *software factors*, which comprises ten items related to effectiveness, efficiency, completeness, and adaptability, was called the software factor.

Our analyses of the twenty items reveal a two-component structure (Table 5). The *nonsoftware component* consists of ten items explaining 29% of the variance and yields an eigenvalue of 5.8. Eigenvalues are a measure of the magnitude of a component. The component of *nonsoftware factors* is highly correlated with high internal consistency (Cronbach alpha = 0.813). The *software factors* consist of ten items explaining 10% of the variance and yield an eigenvalue of 2.0. The *software factors* also highly correlate (Cronbach's alpha = 0.790). The results present the commonality scores, indicating how well each item fits the components. Furthermore, all existing theoretical guarantees for PCA assume that the data and the corrupting components (all items grouped together into one theme) are mutually independent (uncorrelated).

Table 5. Pattern matrix. Rotation method: Oblimin with Kaiser normalization. The rotation converged in 8 iterations.

Question	Nonsoftware Factors	Software Factors	Commonalities
Q20	0.786		0.606
Q19	0.708		0.568
Q13	0.688		0.499
Q14	0.623		0.384
Q21	0.608		0.375
Q8	0.572		0.294
Q15	0.515		0.347
Q11	0.492		0.217
Q18	0.483		0.397
Q4	0.447		0.278
Q22	0.419	0.362	0.404
Q6		0.741	0.496
Q3		0.662	0.401
Q9		0.618	0.371
Q7		0.617	0.452
Q10		0.504	0.243
Q5		0.500	0.386
Q16		0.487	0.419
Q17		0.469	0.303
Q12	0.343	0.380	0.346
Eigenvalues	5.815	1.971	
Percent variance explained	29.076	9.857	

Table 6 presents the average percentage response grouped by role type and also by identified components from the analysis of the main component factor. It is evident that based on the identified factor, participants undertaking a technical role more strongly agree that nonsoftware reasons are preventing their use, and they more strongly disagree that software reasons are preventing their use. It is also evident that they more strongly agree with the nonsoftware factor being responsible for not adopting automated testing. Participants undertaking a nontechnical role more strongly agree that both nonsoftware and software factors are preventing the use of automated testing.

Table 6. Average % responses to questions, grouped by role type and factor.

Component	% Disagree	Technical Role		% Disagree	Nontechnical Role	
		% Neutral	% Agree		% Neutral	% Agree
Nonsoftware	35	25	41	22	23	55
Software	50	21	29	31	24	45

4. Discussion and Findings

The purpose of this study was to identify key themes when investigating opinions on why automated testing is not used. Through this two-stage study, the nature of the relationship between a set of predictors including software characteristics, nonsoftware issues, and those reasons relating to practitioner support and opposition to AT adoption were investigated. In this spirit, scholars have found that AT characteristics, e.g., functionality, usability, and adaptability, can have a strong effect on practitioners' support or opposition. In particular, we sought to test these predictors in different scenarios to gain an understanding of how perceptions of individuals operating in different roles and with different levels of experience differ. To that end, it has been established that there are key identifiable patterns surrounding attitudes toward automated testing from employees who

assume different roles and have different levels of experience. These key findings can be used by employers in the software industry to better understand the viewpoints of their employees.

Based on the values in Table 6, the responses for technical roles are asymmetric, as technical roles believe that the reasons for not adopting AT are due to nonsoftware factors. However, the responses for nontechnical roles are symmetric and agree that both nonsoftware and software reasons are the factors preventing adoption. We deduce that this could be down to the following reasons: (1) questions in nonsoftware factors related to cost that all (i.e., not just nontechnical) employees agree with; (2) based on common practice in the IT sector, technical employees are often promoted to nontechnical (managerial) roles, meaning that they have both technical and nontechnical attitudes; and (3) nontechnical might have less understanding of how capable technical people are, i.e., management lacks understanding of their employees' skills.

Based on the combination of the comprehensive basic analysis and the principal component analysis, we draw the key findings presented in the remainder of this section. Throughout this section, the original questions and their responses are cross-referenced by adding the question number in parentheses (e.g., q3 for question 3). In this section, optional free-text responses provided by the user are analysed alongside the previously discussed quantitative information. The complete free-text responses provided by 19 of the participants can be seen in Table A1, and since this section is trying to establish key findings from the data, they are used to support the quantitative patterns. A summary of the key themes of the free-text submissions can be seen in Table A1.

Summary Point 1. *Although technical employees are more likely to believe that testers require a high level of expertise and that open-source tools are challenging, this is not identified as a factor preventing their adoption. However, on the contrary, nontechnical roles agree that an absence of expertise is preventing the use of automated testing.*

When asking participants if they believe that a lack of skilled resources is preventing automated testing from being used, it is evident that management staff believe this to be true, while those with more technical expertise do not (q3). This is an interesting finding, as it confirms that there is a different perception between technical and nontechnical staff in regard to whether a lack of skilled resources is a prevention factor. This could be because nontechnical staff are unable to determine the requirements of expertise and match them with the capabilities within their organisation. Furthermore, it could also be because technical staff overstate their ability without having significant experience in automated testing.

It is also evident that people do not believe that automated testing is not fully utilised because people do not realise its benefits (q9). Furthermore, technical roles do not believe that there is an issue with open-source tools; however, less technical roles are more likely to support this argument (q12). Additionally, there is a weak indication that those with technical expertise believe that a high level of expertise is required (q13). It is, however, evident that the majority of the participants believe that strong programming skills are required to undertake automated testing (q14). However, when relating this to the results of the analysis of the principal components, it is evident that the technical staff do not believe that technical reasons prevent the use of automated testing.

It is perhaps not too surprising that technical roles are more likely to believe that a high level of expertise is required. This is because they are working closely with technology and will have a comprehensive understanding of what knowledge is required. However, as demonstrated, technical roles are less likely to believe that skilled resources are preventing the use of automated testing, as they have already gone through the learning process and become competent testers. On the contrary, management is more likely to view the capability within their organisation versus what is to be delivered, and therefore, a lack of skilled resources might refer to there being insufficient resources available to deliver a

project on time, rather than the absence of expertise, from preventing thorough software testing. In contrast, it is possible for those in technical roles to report that they have the expertise required to utilise automated testing tools; however, this raises the question of why they are not always used if the necessary expertise is available.

In terms of comments provided by the participants, 7 of the 19 responses were directed at the necessity and lack of expertise. All of these 7 responses, shown in Table A1, are provided by individuals performing technical roles. Interestingly, all responses agree that technical knowledge is important, but an interesting observation is that some responses draw attention to the fact that there is a lack of training and mentorship within testing roles. One response even highlights the importance of individuals being able to learn the necessary skills independently. It is also interesting that a couple of responses directly state that the management of people is extremely important to help remove any skill and expertise gaps, resulting in a more thorough and robust testing process.

Summary Point 2. *Those with less experience are more likely to agree that individuals do not have enough time to participate in automated tests. Furthermore, employees with less technical experience with automated testing and greater management responsibilities do not agree that they are time consuming to learn.*

Whether individuals have enough time to perform automated testing is polarised, with an even split agreeing and disagreeing. However, it has been identified that those with more junior roles are more likely to agree with this statement (q4). Furthermore, when considering how difficult they are to learn, the majority of people disagree that they are time consuming to learn. However, in general, the least experienced employees tend to agree, and so do managers and CEOs (q15). This agrees with the results of the principal component analysis, whereby technical staff are identified to agree that nontechnical reasons are behind not adopting automated testing.

This finding agrees with the fact that work levels and deadline pressures will be different in different organisations, and, furthermore, people will respond to and handle these pressures differently. The fact that junior employees are more likely to state that they do not have sufficient time to perform automated testing duties is explained by the fact that junior employees might take longer to perform testing duties. This could also be due to a lack of experience as a result of the employee learning new skills necessary for their role, which could slow down testing. It is also possible that those with less experience are burdened with learning the necessary knowledge and expertise to perform their entire role and therefore have little capacity to take on improvement activities. This may change as an individual gains more experience, becoming more efficient in their role and creating more space for learning and improvement.

Summary Point 3. *Most of the participants agree that automated testing is expensive, with nontechnical roles more likely to agree that they are expensive to use and maintain.*

Most of the participants agree that commercial tools are expensive to use, but there is no discernible pattern (q11). However, there is a weak correlation that management roles are more likely to agree with the statement that test scripts are more expensive to generate (q19). This is further compounded when nontechnical roles agree that there are high maintenance costs for test cases and scripts (q20). This agrees with the presented principal component analysis, as both technical employees agree with nontechnical reasons being responsible for not adopting automated testing. Furthermore, nontechnical roles are split between believing that software and nonsoftware factors are responsible for not adopting automated testing.

It is not surprising that most of the users agree that the costs of automated testing are expensive. Furthermore, the pattern that management staff more strongly agree with this statement is explainable through their closeness to the financial operations of the business. It is, however, quite surprising that managerial staff believe that automated testing has

high maintenance costs. A fundamental aspect of automated testing is its reuse and ease of maintenance. This difference in perspective is likely to originate from management's lack of understanding when it comes to fundamental aspects of automated testing.

The comments provided by the participants also reflect the fact that automated testing is expensive to perform and maintain, largely due to the cost of the testing team. One participant (# 75) states that management does not see the wasted amount of time in automated software testing, and this could provide justification as to why nontechnical roles agree that they are expensive to maintain. If they saw the amount of wasted time, they might have a better understanding of the true cost.

Summary Point 4. *All but the most experienced employees disagree that automated testing tools and techniques lack functionality. Furthermore, experienced employees are more likely to disagree that problems arise due to fast revisions, whereas those with managerial roles agree.*

When considering whether automated testing tools and techniques lack functionality, in general, the more experienced employees are likely to agree, but overall the majority disagree (q16). When asked whether people believe that automated tools are reliable enough, there was a very strong tendency to disagree (q17). There is a slight agreement that people believe that automated testing tools lack support for testing nonfunctional requirements (q18). When asked whether automated testing tools and techniques change too often, introducing problems that need fixing, the general trend is that a greater number of years of experience leads to an increased chance of disagreement. Furthermore, of the response categories, nontechnical roles agree/strongly agree (q21). This aligns with the findings from performing principal component analysis where nontechnical roles more strongly believe that software reasons are preventing the use of automated testing, whereas those undertaking technical roles believe it is nonsoftware issues.

The reason why more experienced employees disagree that automated testing tools and techniques lack functionality is most likely down to the fact that more experienced employees either have fully mastered the tools, or they have developed sufficient workaround or alternative techniques. Furthermore, experienced staff do not believe that updates cause significant problems, which could be put down to the fact that they are experienced in how to handle revisions within automated testing frameworks. Nonfunctional requirements are a secondary feature set of automated testing tools and techniques and, as such, are not the primary feature set integral to their core use. This is most likely the reason why the majority of participants do not see an issue with their lack of support for nonfunctional requirements.

Many comments were received in regard to the capabilities of tools and techniques, and in general, they state that the tools, techniques, and frameworks do not lack functionality. Rather, they justify the complexity of tightly integrating the functionality within a project and how this can make it hard to reuse and fix revisions. Furthermore, it is evident that technical employees also believe that those in managerial roles do not understand what is involved in the implementation of automated testing. It is also interesting that one response from an individual performing a technical role (#64) even states that test scripts breaking is a good sign, as it clearly demonstrates that they are working. A comment from an individual in management (#81) states that product delivery is more important than testing, demonstrating that for management, their emphasis is on project completion rather than testing.

Summary Point 5. *Only managerial staff believe that test preparation and integration inhibit their use. Furthermore, only management staff do not believe that software requirements change too frequently, having negative impacts on automated testing.*

In terms of utilisation, when considering whether difficulties in preparing test data and scripts inhibit their use, only nontechnical staff agree, and there is a balanced response from technical roles (q5). Furthermore, the majority of the participants do not believe that not having the right automation tools and available frameworks prevents use (q6). When asking

staff specifically about whether the difficulty of integrating tools is a problem, nontechnical roles agree, and technical roles are balanced with a slight emphasis on disagreement (q7). The majority of participants also agreed that requirements changing too frequently are impacting their use (q8); however, it is also the case that nontechnical roles do not agree. There is also strong disagreement among the technical staff that test scripts are difficult to reuse in different testing stages (q22). This finding also agrees with the performed principal component analysis, where it was identified that nontechnical staff more strongly believe the reasons for not adopting automated testing to be technical.

The fact that nontechnical employees believe that there are difficulties, both in setting up and maintaining automated tests, which are prohibiting the use of automated testing tools is most likely down to the disconnect between nontechnical and technical staff when it comes to understanding limitations with software testing. All participants believe that there are sufficient frameworks to meet their individual testing requirements. Interestingly, only management believes that changing requirements do not impact automated testing techniques. This difference could most likely originate from a managerial misunderstanding of the impact of changing requirements throughout the software development cycle.

Comments provided by the participants do support the argument that those in testing roles understand the technical complexities involved and why automated testing might not be fully utilised. However, there is a lack of responses from management staff to justify that this is only a point of view of technical staff. There are many reasons specified for poor utilisation, from formal training and guidance, a preference to view automated testing as second to manual, and that automation might be used for the wrong reasons, i.e., to replace manual rather than complement.

Summary Point 6. *Whether a lack of support prevents the use of automated testing is polarised.*

There is no agreement or disagreement that lack of support prevents the use of automated testing. However, there is an observation that consultants tend to agree with this statement (q10). This is interesting, as it demonstrates that there is no majority, either in terms of role or experience, stating that lack of support is preventing them from adopting and using automated testing within their organisation. However, it is also worth noting that the responses to this question are rather polarised, with people agreeing and disagreeing, but overall there are few holding strong views on this. This is consistent with the performed principal component analysis, which determined that nontechnical roles and technical roles agree (technically more strongly) that nonsoftware factors, such as finance, expertise, and time, are preventing the adoption of automated testing.

Comments received from participants detail that training is a common limiting factor to their uptake, but the biggest theme is that nontechnical either do not understand or value test automation. This means that automation is seen as an afterthought from manual testing and thus will not be well supported by their employer.

5. Conclusions

Six key findings have been established, which demonstrate key differences in the perceptions of both technical and managerial employees, as well as of employees of different levels of experience. The two-stage analysis approach presented in this paper demonstrated that an overarching two-factor split can be established when considering the attitudes towards automated testing of both technical and nontechnical staff. It has been established that technical employees strongly believe that preventing factors to automated testing use are those of a nonsoftware nature, whereas nontechnical roles believe that it is for both software and nonsoftware reasons. These attitudes have been further analysed and explained considering different roles and years of experience. In addition to the implication of these findings in software development organisations, they also have significance and implication more broadly throughout Information Technology. For example, any reduced efficiency or effectiveness within software development organisations has the potential

to reduce software quality and increase costs. Furthermore, the difference in perspective between those in technical and nontechnical roles has the potential to introduce additional costs and delays. Those in managerial roles could invest in training and resources that may not be required and add further delays. Failure to adequately address these challenges might result in an organisation and its software products gaining a competitive disadvantage, and the organisation may suffer, negatively affecting its ability to operate. The findings presented in the article highlight that a culture change is required that could involve training programmes focused on both technical and nontechnical roles to improve awareness and create a supportive environment for learning and adopting automated testing practices.

Although the study is based on responses from 81 different users, future work should focus on gaining a larger number of samples with a more even distribution across the different role types. However, it is important to note that 81 responses from those working in software testing are significant and worthy of investigation and analysis. Another limitation of the study is that the questionnaire focused on reasons why automated testing is not adopted and is therefore negatively formulated. This means that the positive aspects of using automated testing have been ignored. Although this was a deliberate design choice in this work, collecting positive attitudes can also help to gain a deeper understanding.

The use of the two-stage analysis ensures that each question is interpreted before investigating the responses to identify the relationships between factors. Although this provides a systematic analysis approach, it does have limitations. One of the main limitations of this survey is that it is performed on a relatively small (81) dataset, which makes it difficult to form a generalised view and opinions. Although this was a deliberate design choice to incentivise busy people to undertake the survey, we acknowledge the potential impact. However, loadings with a high *Cronbach's alpha* that have several high loading marker variables (>0.80) do not require such large sample sizes as solutions with lower loadings [59]. Our results produced a *Cronbach alpha* of 0.86, justifying the reliability of the survey. Another limitation of the study is that, by using a convenient sampling approach, there is the potential to introduce sampling error or bias. As we use LinkedIn and special interest groups to identify participants, it is possible that we have not captured representative and unbiased views of the data, as those who participated in the study could have stronger negative opinions against the adoption of software testing. Another limitation is that the questionnaire does not cover all factors involved in the automated software testing process, and therefore, the key findings may not be true or applicable in all cases. However, this research has achieved its goal of developing an understanding of why people are not adopting automated testing, which establishes a suitable position for further research.

Another limitation of our study is that we consider large-scale AT software in a general sense rather than focusing on any specific AT software. Research finds that public opposition tends to be highest when projects are proposed and then decreases once construction is complete [60]. However, we believe this limitation to be fairly minor because we are trying to understand practitioners' attitudes about AT generally rather than any relation to any testing software adoption. The selection of questionnaire items always restricts the potential structure that can emerge from innovation adoption studies. We designed the questionnaire to include items related to a broad range of potential experiences, motivated both theoretically and by prior qualitative research.

Author Contributions: Conceptualization, G.M. and S.P.; methodology, G.M., S.P., S.K. and N.L.; formal analysis, G.M., S.P., S.K. and N.L.; data curation, G.M.; writing—original draft preparation, G.M., S.P., S.K. and N.L.; writing—review and editing, G.M., S.P., S.K., N.L. and G.A.; supervision, S.P. and G.A.; project administration, S.P.; All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Informed Consent Statement: Informed consent was obtained from all subjects involved in the study.

Data Availability Statement: Data are contained within the article.

Conflicts of Interest: Author George Murazvu was employed by the company Axia Digital. The remaining authors declare that the research was conducted in the absence of any commercial or financial relationships that could be construed as a potential conflict of interest.

Appendix A

Table A1 provides the comments provided at the two designated points when performing the test. In the table, responses to Q1 have been abbreviated to save space. The response of ‘Tester’ means that they have one of the following job titles: Tester, Engineer, Analyst, Architect, Automation. ‘Senior Tester’ means having the title of a senior version of one of the aforementioned roles.

Table A1. Summary of free-text responses.

Response	Summary Points
2	<ul style="list-style-type: none"> Problems are with people and not technology.
3	<ul style="list-style-type: none"> Skilled team will prevent issues.
5	<ul style="list-style-type: none"> False expectations from management. Lack of consideration to test data and scripts. People are the cost and over expectation of automation. Setting and using testing tools requires knowledge.
15	<ul style="list-style-type: none"> Company recognise value of automation.
20	<ul style="list-style-type: none"> High maintenance due to when testing is performed.
24	<ul style="list-style-type: none"> Management do not understand automation.
33	<ul style="list-style-type: none"> People management is poor.
35	<ul style="list-style-type: none"> Lack of mentorship and guidance. Self-teaching is important and widely performed.
39	<ul style="list-style-type: none"> Management does not understand the time/effort required. Automation is always seen as nice to have after the manual is performed. Benefits of automated testing are significant.
43	<ul style="list-style-type: none"> Implementation of open-source frameworks is key to their value.
49	<ul style="list-style-type: none"> Training is the most challenging aspect.
51	<ul style="list-style-type: none"> Management sees value in automated testing for the stability of the end product.
55	<ul style="list-style-type: none"> Training is the most challenging point.
56	<ul style="list-style-type: none"> Once automated testing is used, it may become tricky to understand its value.
64	<ul style="list-style-type: none"> Automation still viewed as secondary. The need to maintain scripts as new tools and techniques develop is a good sign.
66	<ul style="list-style-type: none"> Automated testing is more of a process than a skill. Test automation needs to be pre-planned to consider software development factors.

Table A1. Cont.

Response	Summary Points
75	<ul style="list-style-type: none"> • New experienced testers are likely to make mistakes. • Automation engineers lack product knowledge and are disconnected from the project they are working on. • Those involved in automation spend lots of time making scripts and maintaining them. • Managers often do not see the level of waste in automated testing. • Managers invest heavily without seeing or understanding the benefit. • Management pushing advice on automation without knowledge is a bad thing. • People often build their own frameworks, but spending too much time here can be disadvantageous for the project. • Translation of testing output to management is currently underperformed. • Changing requirements is normal and a necessary part of a product's life cycle. • Automation used for the wrong reasons. • Automated testing can make it hard to see the true benefits, and therefore management is likely to want it until they do not see any positive impact. • Commercial tools are too expensive. • Open-source tools are not hard to utilise unless the person is unfamiliar with the area. • Knowing when and how to use automated testing is important. • Level of programming ability is less important than the ability to learn when needed. • A wrongly utilised tool is expensive. • Challenges with maintenance stem from a lack of understanding. • Bespoke and low-level test scripts can be translated to new projects, but this is necessary as they encode application-specific behaviour.
77	<ul style="list-style-type: none"> • A deep understanding is required.
81	<ul style="list-style-type: none"> • Product delivery is more important than testing.

References

1. Charette, R. Why software fails. *IEEE Spectr.* **2005**, *42*, 42–49. [[CrossRef](#)]
2. Ammann, P.; Offutt, J. *Introduction to Software Testing*; Cambridge University Press: Cambridge, UK, 2016.
3. Dustin, E.; Rashka, J.; Paul, J. *Automated Software Testing: Introduction, Management, and Performance*; Addison-Wesley Professional: Boston, MA, USA, 1999.
4. Elghondakly, R.; Moussa, S.; Badr, N. Waterfall and agile requirements-based model for automated test cases generation. In Proceedings of the 2015 IEEE Seventh International Conference on Intelligent Computing and Information Systems (ICICIS), Cairo, Egypt, 12–14 December 2015; IEEE: Piscataway, NJ, USA, 2015; pp. 607–612.
5. Al-Saqqa, S.; Sawalha, S.; AbdelNabi, H. Agile software development: Methodologies and trends. *Int. J. Interact. Mob. Technol.* **2020**, *14*, 246–270. [[CrossRef](#)]
6. Rafi, D.M.; Moses, K.R.K.; Petersen, K.; Mäntylä, M.V. Benefits and limitations of automated software testing: Systematic literature review and practitioner survey. In Proceedings of the 2012 7th International Workshop on Automation of Software Test (AST), Zurich, Switzerland, 2–3 June 2012; IEEE: Piscataway, NJ, USA, 2012; pp. 36–42.
7. Asfaw, D. Benefits of Automated Testing Over Manual Testing. *Int. J. Innov. Res. Inf. Secur.* **2015**, *2*, 5–13.
8. Collins, E.F.; De Lucena, V.F. Software test automation practices in agile development environment: An industry experience report. In Proceedings of the 2012 7th International Workshop on Automation of Software Test (AST), Zurich, Switzerland, 2–3 June 2012; IEEE: Piscataway, NJ, USA, 2012; pp. 57–63.
9. Wiklund, K.; Eldh, S.; Sundmark, D.; Lundqvist, K. Impediments for software test automation: A systematic literature review. *Softw. Test. Verif. Reliab.* **2017**, *27*, e1639. [[CrossRef](#)]
10. Bear, S. *State of Testing*; Technical report; Smart Bear: Somerville, MA, USA, 2018.
11. Taipale, O.; Kasurinen, J.; Karhu, K.; Smolander, K. Trade-off between automated and manual software testing. *Int. J. Syst. Assur. Eng. Manag.* **2011**, *2*, 114–125. [[CrossRef](#)]
12. Nass, M.; Alégroth, E.; Feldt, R. Why many challenges with GUI test automation (will) remain. *Inf. Softw. Technol.* **2021**, *138*, 106625. [[CrossRef](#)]
13. Khan, A.Z.; Iftikhar, S.; Bokhari, R.H.; Khan, Z.I. Issues/challenges of automated software testing: A case study. *Pak. J. Comput. Inf. Syst.* **2018**, *3*, 61–75.
14. Evans, I.; Porter, C.; Micallef, M. Scared, frustrated and quietly proud: Testers' lived experience of tools and automation. In Proceedings of the 32nd European Conference on Cognitive Ergonomics, Siena Italy, 26–29 April 2021; pp. 1–7.
15. Li, B.; Zhao, Q.; Jiao, S.; Liu, X. DroidPerf: Profiling Memory Objects on Android Devices. In Proceedings of the 29th Annual International Conference on Mobile Computing and Networking, Madrid Spain, 2–6 October 2023; pp. 1–15.

16. Li, B.; Xu, H.; Zhao, Q.; Su, P.; Chabbi, M.; Jiao, S.; Liu, X. OJXPerf: Featherlight object replica detection for Java programs. In Proceedings of the 44th International Conference on Software Engineering, Pittsburgh, PA, USA, 21–29 May 2022; pp. 1558–1570.
17. Hynninen, T.; Kasurinen, J.; Knutas, A.; Taipale, O. Guidelines for software testing education objectives from industry practices with a constructive alignment approach. In Proceedings of the 23rd Annual ACM Conference on Innovation and Technology in Computer Science Education, Larnaca, Cyprus, 2–4 July 2018; pp. 278–283.
18. Felderer, M.; Büchler, M.; Johns, M.; Brucker, A.D.; Breu, R.; Pretschner, A. Security testing: A survey. In *Advances in Computers*; Elsevier: Amsterdam, The Netherlands, 2016; Volume 101, pp. 1–51.
19. Larusdottir, M.K.; Bjarnadottir, E.R.; Gulliksen, J. The focus on usability in testing practices in industry. In *Human-Computer Interaction, Proceedings of the Second IFIP TC 13 Symposium, HCIS 2010, Held as Part of WCC 2010, Brisbane, Australia, 20–23 September 2010*; Proceedings; Springer: Berlin/Heidelberg, Germany, 2010; pp. 98–109.
20. Hourani, H.; Hammad, A.; Lafi, M. The impact of artificial intelligence on software testing. In Proceedings of the 2019 IEEE Jordan International Joint Conference on Electrical Engineering and Information Technology (JEEIT), Amman, Jordan, 9–11 April 2019; IEEE: Piscataway, NJ, USA, 2019; pp. 565–570.
21. Vos, T.E.; Marin, B.; Escalona, M.J.; Marchetto, A. A methodological framework for evaluating software testing techniques and tools. In Proceedings of the 2012 12th International Conference on Quality Software, Xi'an, China, 27–29 August 2012; IEEE: Piscataway, NJ, USA, 2012; pp. 230–239.
22. Eldh, S.; Hansson, H.; Punnekkat, S.; Pettersson, A.; Sundmark, D. A framework for comparing efficiency, effectiveness and applicability of software testing techniques. In Proceedings of the Testing: Academic & Industrial Conference-Practice And Research Techniques (TAIC PART'06), Windsor, UK, 29–31 August 2006; IEEE: Piscataway, NJ, USA, 2006; pp. 159–170.
23. Infosys. Infosys Test Automation Accelerator. 2019. Available online: <https://www.infosys.com/IT-services/validation-solutions/Documents/infosys-test-automation-accelerator.pdf> (accessed on 20 November 2023).
24. Kumar, D.; Mishra, K. The Impacts of Test Automation on Software's Cost, Quality and Time to Market. *Procedia Comput. Sci.* **2016**, *79*, 8–15. [[CrossRef](#)]
25. Mittal, V.; Garg, N. *Test Automation using Selenium Webdriver 3.0 with C#*; AdactIn Group Pty Limited: Parramatta, Australia, 2018.
26. Vogel-Heuser, B.; Fay, A.; Schaefer, I.; Tichy, M. Evolution of software in automated production systems: Challenges and research directions. *J. Syst. Softw.* **2015**, *110*, 54–84. [[CrossRef](#)]
27. Zhou, Z.Q.; Sinaga, A.; Susilo, W.; Zhao, L.; Cai, K.Y. A cost-effective software testing strategy employing online feedback information. *Inf. Sci.* **2018**, *422*, 318–335. [[CrossRef](#)]
28. Panichella, S.; Di Sorbo, A.; Guzman, E.; Visaggio, C.A.; Canfora, G.; Gall, H.C. How can i improve my app? Classifying user reviews for software maintenance and evolution. In Proceedings of the 2015 IEEE International Conference on Software Maintenance and Evolution (ICSME), Bremen, Germany, 29 September–1 October 2015; IEEE: Piscataway, NJ, USA, 2015; pp. 281–290.
29. Tracey, N.; Clark, J.; Mander, K.; McDermid, J. An automated framework for structural test-data generation. In Proceedings of the Proceedings 13th IEEE International Conference on Automated Software Engineering (Cat. No. 98EX239), Honolulu, HI, USA, 13–16 October 1998; IEEE: Piscataway, NJ, USA, 1998; pp. 285–288.
30. Fewster, M.; Graham, D. *Software Test Automation: Effective Use of Test Execution Tools*; ACM Press: New York, NY, USA; Addison-Wesley Publishing Co.: Boston, MA, USA, 1999.
31. Graham, D.; Fewster, M. *Experiences of Test Automation: Case Studies of Software Test Automation*; Addison-Wesley Professional: Boston, MA, USA, 2012.
32. Böhme, M.; Paul, S. A probabilistic analysis of the efficiency of automated software testing. *IEEE Trans. Softw. Eng.* **2015**, *42*, 345–360. [[CrossRef](#)]
33. Rahman, A.A.; Hasim, N. Defect Management Life Cycle Process for Software Quality Improvement. In Proceedings of the 2015 3rd International Conference on Artificial Intelligence, Modelling and Simulation (AIMS), Kota Kinabalu, Malaysia, 2–4 December 2015; pp. 241–244. [[CrossRef](#)]
34. Garrett, T. Useful Automated Software Testing Metrics. *Softw. Test. Geek* **2011**.
35. Rex, B. *Managing the Testing Process: Practical Tools and Techniques for Managing Hardware and Software Testing*; Rex Black Inc.: Dallas, TX, USA, 2002.
36. Berner, S.; Weber, R.; Keller, R.K. Observations and lessons learned from automated testing. In Proceedings of the 27th International Conference on Software Engineering, St. Louis, MO, USA, 15–21 May 2005; ACM: New York, NY, USA, 2005; pp. 571–579.
37. Jansing, D.; Novillo, J.; Cavallo, R.; Spetka, S. Enhancing the Effectiveness of Software Test Automation. Ph.D Thesis, State University of New York Polytechnic Institute Utica, New York, NY, USA, 2015.
38. Dustin, E.; Garrett, T.; Gauf, B. *Implementing Automated Software Testing: How to Save Time and Lower Costs While Raising Quality*; Pearson Education: London, UK, 2009.
39. Melton, J.R. The Hidden Benefits of automated Testing. In Proceedings of the 2015 Aerospace Testing Senior, CVENTS, Los Angeles, California, 27–29 October 2015;
40. Kasurinen, J.; Taipale, O.; Smolander, K. Software test automation in practice: Empirical observations. *Adv. Softw. Eng.* **2010**, *2010*, 620836. [[CrossRef](#)]

41. Leitner, A.; Ciupa, I.; Meyer, B.; Howard, M. Reconciling manual and automated testing: The autotest experience. In Proceedings of the 2007 40th Annual Hawaii International Conference on System Sciences (HICSS'07), Big Island, HI, USA, 3–6 January 2007; IEEE: Piscataway, NJ, USA, 2007; p. 261a.
42. Monier, M.; El-mahdy, M.M. Evaluation of automated web testing tools. *Int. J. Comput. Appl. Technol. Res.* **2015**, *4*, 405–408. [\[CrossRef\]](#)
43. Garousi, V.; Felderer, M. Worlds Apart: Industrial and Academic Focus Areas in Software Testing. *IEEE Softw.* **2017**, *34*, 38–45. [\[CrossRef\]](#)
44. Zou, W.; Lo, D.; Chen, Z.; Xia, X.; Feng, Y.; Xu, B. How practitioners perceive automated bug report management techniques. *IEEE Trans. Softw. Eng.* **2018**, *46*, 836–862. [\[CrossRef\]](#)
45. Lo, D.; Nagappan, N.; Zimmermann, T. How practitioners perceive the relevance of software engineering research. In Proceedings of the 2015 10th Joint Meeting on Foundations of Software Engineering, Bergamo, Italy, 30 August 2015; pp. 415–425.
46. Meyer, A.N.; Fritz, T.; Murphy, G.C.; Zimmermann, T. Software developers' perceptions of productivity. In Proceedings of the 22nd ACM SIGSOFT International Symposium on Foundations of Software Engineering, Hong Kong, China, 16–21 November 2014; pp. 19–29.
47. Abdi, H.; Williams, L.J. Principal component analysis. *Wiley Interdiscip. Rev. Comput. Stat.* **2010**, *2*, 433–459. [\[CrossRef\]](#)
48. Jolliffe, I.T.; Cadima, J. Principal component analysis: A review and recent developments. *Philos. Trans. R. Soc. A Math. Phys. Eng. Sci.* **2016**, *374*, 20150202. [\[CrossRef\]](#)
49. Boone, H.N.; Boone, D.A. Analyzing likert data. *J. Ext.* **2012**, *50*, 48. [\[CrossRef\]](#)
50. Etikan, I.; Musa, S.A.; Alkassim, R.S. Comparison of convenience sampling and purposive sampling. *Am. J. Theor. Appl. Stat.* **2016**, *5*, 1–4. [\[CrossRef\]](#)
51. Faraj, S.; Sproull, L. Coordinating expertise in software development teams. *Manag. Sci.* **2000**, *46*, 1554–1568. [\[CrossRef\]](#)
52. Gaskin, C.J.; Happell, B. On exploratory factor analysis: A review of recent evidence, an assessment of current practice, and recommendations for future use. *Int. J. Nurs. Stud.* **2014**, *51*, 511–521. [\[CrossRef\]](#) [\[PubMed\]](#)
53. Nunnally, J.C.; Ira, H.B. *Psychometric Theory*; McGraw-Hill: New York, NY, USA, 1994.
54. Ferketich, S. Focus on psychometrics. Aspects of item analysis. *Res. Nurs. Health* **1991**, *14*, 165–168. [\[CrossRef\]](#) [\[PubMed\]](#)
55. Cortina, J.M. What is coefficient alpha? An examination of theory and applications. *J. Appl. Psychol.* **1993**, *78*, 98. [\[CrossRef\]](#)
56. Streiner, D.L.; Norman, G.R.; Cairney, J. *Health Measurement Scales: A Practical Guide to Their Development and Use*; Oxford University Press: Oxford, UK, 2015.
57. Ferguson, E.; Cox, T. Exploratory factor analysis: A users' guide. *Int. J. Sel. Assess.* **1993**, *1*, 84–94. [\[CrossRef\]](#)
58. Tobias, S.; Carlson, J.E. Brief report: Bartlett's test of sphericity and chance findings in factor analysis. *Multivar. Behav. Res.* **1969**, *4*, 375–377. [\[CrossRef\]](#)
59. Tabachnick, B.G.; Fidell, L.S.; Ullman, J.B. *Using Multivariate Statistics*; Pearson: Boston, MA, USA, 2007; Volume 5.
60. Warren, C.R.; Lumsden, C.; O'Dowd, S.; Birnie, R.V. 'Green on green': Public perceptions of wind power in Scotland and Ireland. *J. Environ. Plan. Manag.* **2005**, *48*, 853–875. [\[CrossRef\]](#)

Disclaimer/Publisher's Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.