processes

Article

An Empirical Investigation to Understand the Issues of Distributed Software Testing amid COVID-19 Pandemic

Abdullah Alharbi [1], Md Tarique Jamal Ansari [2], Wael Alosaimi [1], Hashem Alyami [3], Majid Alshammari [1], Alka Agrawal [2], Rajeev Kumar [4,*], Dhirendra Pandey [2] and Raees Ahmad Khan [2]

1 Department of Information Technology, College of Computers and Information Technology, Taif University, P.O. Box 11099, Taif 21944, Saudi Arabia; amharbi@tu.edu.sa (A.A.); w.osaimi@tu.edu.sa (W.A.); m.alshammari@tu.edu.sa (M.A.)
2 Department of Information Technology, Babasaheb Bhimrao Ambedkar University, Lucknow 226025, Uttar Pradesh, India; tjtjansari@gmail.com (M.T.J.A.); alka_csjmu@yahoo.co.in (A.A.); prof.dhiren@gmail.com (D.P.); khanraees@yahoo.com (R.A.K.)
3 Department of Computer Science, College of Computers and Information Technology, Taif University, P.O. Box 11099, Taif 21944, Saudi Arabia; hyami@tu.edu.sa
4 Department of Computer Science and Engineering, Babu Banarasi Das University, Lucknow 226028, Uttar Pradesh, India
* Correspondence: rs0414@gmail.com

Abstract: Generally, software developers make errors during the distributed software development process; therefore, software testing delay is a significant concern. Some of the software mistakes are minor, but others may be costly or harmful. Since things can still go wrong—individuals encounter mistakes from time to time—there is a need to double-check any software we develop in a distributed environment. The current global pandemic, COVID-19, has exacerbated and generated new challenges for IT organizations. Many issues exist for distributed software testing that prevent the achievement of successful and timely risk reduction when several of the mechanisms on which testing is based are disrupted. The environment surrounding COVID-19 is quickly evolving on a daily basis. Moreover, the pandemic has exposed or helped to develop flaws in production systems, which obstruct software test completion. Although some of these issues were urgent and needed to be evaluated early during the distributed software development process, this paper attempts to capture the details that represent the current pandemic reality in the software testing process. We used a Fuzzy TOPSIS-based multiple-criteria decision-making approach to evaluate the distributed software testing challenges. The statistical findings show that data insecurity is the biggest challenge for successful distributed software testing.

Keywords: software testing; test automation; best practices; remote work; software quality assurance

check for updates

Citation: Alharbi, A.; Ansari, M.T.J.; Alosaimi, W.; Alyami, H.; Alshammari, M.; Agrawal, A.; Kumar, R.; Pandey, D.; Khan, R.A. An Empirical Investigation to Understand the Issues of Distributed Software Testing amid COVID-19 Pandemic. *Processes* **2022**, *10*, 838. https://doi.org/10.3390/pr10050838

Academic Editors: Jan Pitel and Chien-Chih Wang

Received: 3 April 2022
Accepted: 21 April 2022
Published: 24 April 2022

1. Introduction

The execution of functional specifications is taken into consideration when evaluating software system performance. It must ensure that the program code is compatible with the software design at both the unit and system levels. Delivering a high-quality consumer product with special and creative functionality has always been a primary concern for software companies around the globe [1,2]. The development team, on the other hand, cannot assure these factors without testing software applications under a variety of anticipated and unforeseen conditions. As a result, testing is carried out on all device components, big and small. Software testing is a technique for determining whether the real software system satisfies the intended specifications and ensuring that it is defect-free [3,4]. It entails the use of manual as well as automated methods to test one or more desired properties by executing software system modules. In comparison to real specifications, software testing aims to find bugs, holes, and inconsistencies.

The demand for software testing is wide and increasing [5–11]. It is undergoing diversification, which includes increased automation, a faster time to value, a change left and right, API testing, and group layout standardizing. Furthermore, with the increasing development of mobile and smart devices, multi-form component testing is becoming more common. The software testing industry is ripe for exploitation, and testers are specifically interested in new technologies that use machine learning and image processing to enhance QA processes.

According to a MarketsandMarkets study, the software testing products and services market produced USD 32 billion in revenue in 2017 and is expected to reach nearly USD 70 billion by 2023, representing a 14 percent CAGR. The software testing sector is responsible for over 70% of the market, but it is also the slowest rising sub-category, with a CAGR of just 13%. The test automation software segment claims about 24% of the industry, whereas the API testing segment is the most minor, with just 1% of the market [12–19]. Figure 1 shows the graphical representation of the software testing market growth report.

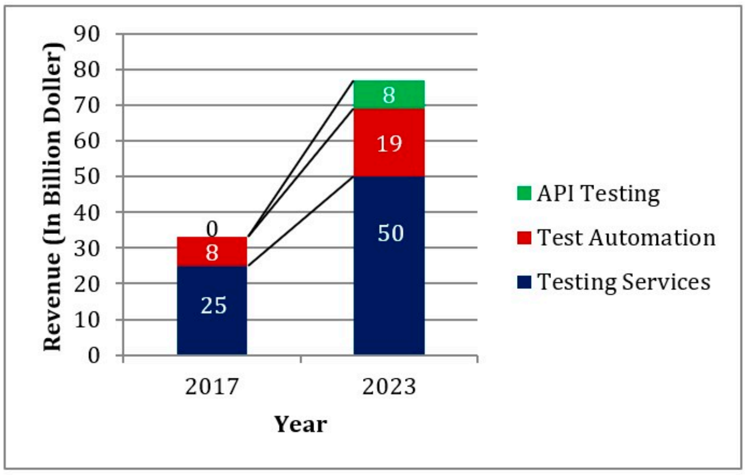

Figure 1. Software testing market growth.

This COVID-19 battle may end up leaving an indelible mark, even establishing and reinforcing a new standard. These are challenging times, with the healthcare sector leading the charge while the worldwide economy is in free fall and every business is experiencing interruption. These are critical times for us to band together in order to combat the disease outbreak as well as the worldwide emergency brought on by the COVID-19 spread. Several software firms are testing software applications based on the 'no-touch' concept, ensuring the consistency of zero-touch implementations for several industries, including retail, supply chain, distribution, and others. The performance of such applications is engineered to be dependable, stable, and safe [20–29].

The new market research study from Technavio is named Global Software Testing Services Market 2020–2024. As per this market research report, the global software testing solutions market is projected to expand by USD 34.49 billion between 2020 and 2024, with a CAGR of over 12 percent. The study examines the effect of the COVID-19 pandemic on the Software Testing solution market globally in three different prediction scenarios: optimistic, probable, and pessimistic [30,31].

The software testing service industry will have a neutral to at par effect during the projected period due to the widespread nature of the COVID-19 deadly virus, according to the devalued and upgraded Software Testing Services Market Study. The Information Technology sector is poised to have both a mixed and clear effect as a result of the virus's wide spread across the world. Moreover, as per Technavio's pandemic-centric study, due to a rise in the disease and decreased economic activity, consumer demand would expand at a pace.

The COVID-19 pandemic profoundly influenced people's working conditions. Many diverse sectors started to take hold of less conventional working practices. This is not only viable but also a more practical solution due to technological advances that allow remote work and cooperation. Distributed software developments are probably the most powerful framework for expanding production, productivity, and providing better business, given that the technology industries are always foremost in the innovation field. This became a model for software development amid COVID-19, allowing the company to remain and prosper for everybody throughout a challenging moment.

When software developers work in several places remotely, that is a distributed software development model. These teams frequently use technology such as media channels and project management tools to guarantee that all who are engaged stay in the process. Sometimes, various teams work on initiatives that make a significant contribution to the bigger context and not on a single project and several areas. Distributed software development can function across geographical boundaries. Testing is critical to understanding where humans are in terms of public health performance in almost every conversation on COVID-19 pandemic. Data from the tests are also used in the preparation, control, and recovery procedures. Of course, the same should be true for software testing. Testing is an important method for detecting software bugs and defects [11–13]. In the era of distributed software development, it is important to strike a balance between pace and quality, particularly given the importance of quality in terms of organizational performance. Although it is obvious why testing is relevant in both contexts, we must recognize the discrepancies in testing consequences. In all instances, the significance of testing is to promote consistency as a result, whether it is associated with public health or application performance.

In the context of COVID-19 testing, the result considered as public health quality is a decrease in the number of illnesses and, more significantly, casualties. Preventing the disease's transmission is one way to do this. In the case of software testing, the result considered as software quality is a reduction in the number of severe bugs that make it into development, as well as an improvement in the software/reliability services. The early identification and mitigation of errors are critical in software testing since the cost of remediation rises when the flaw is discovered later. One can see that early intervention and control are critical in both scenarios [14]. The significant task in quality management, of which testing is an important component, is to ensure overall performance. In the light of this overall perspective, we have to analyse the testing efforts in the current situation [15].

Distributed software testing has also progressed from conventional methods of performance testing on applications to the use of cutting-edge AI and machine learning methods to predict software deficiencies [16,17]. The latter approaches detect potential flaws by inspecting other kinds of data (e.g., program logs, event logs) rather than performing individual tests. Likewise, AIOps methods involve AI and ML to optimize not just the mean time to diagnose, but also to deliver detailed underlying cause analysis knowledge that aids in the prevention of incidence.

The properties of the DSD (Distributed Software Development), as well as the agile methodologies for software development, were integrated by Collins et al. [25]. As per the authors, the prior works did not handle the case of the distribution of testing between geographically dispersed teams. On the foundation of this study, the authors propose the creation and testing of the programme when your team members are frequently dispersed geographically. The outcome of this research is to emphasize the issues facing distributed tests and to create a framework for distributed testing by agile software development methods. A dynamic test tool was used by Eassa et al. [26] to discover time problems in agent and agent-based systems using a time logic assertion language. This tool assesses the behaviour of the agent and detects mistakes associated with the compliance of the agent, agent communication errors, user requirements associated errors as well as web application safety. Di Lucca et al. [27] carried out a study of the functional and non-functional requirements of various web applications. The research emphasizes that functionality relies on some key elements such as test levels, test techniques, test cases, test models as well as test processes for the web application. More complicated functions of the distributed test elements are coordination and communication. Azzouzi et al. [28] concentrated on the time requirements for the exchange of information between different elements of the test applications in the distributed setting. They presented a new design to prevent the problem of tester synchronization. The objective of their project was to present improved coordination across multiple tests and to be able to make the detection of defects in a more controllable and observable manner.

Sher [29] analysed project management (PM) issues in global application products as well as offered solutions to these problems. A comprehensive literature review was used to examine these concerns. Cultural discrepancies, a lack of cooperation and consultation, distinct time zones, language difficulties, different organizational styles and processes, as well as knowledge management among remote workers, were the key challenges they discovered. The author aimed to create a project management framework for global application development. Hsaini et al. [30] proposed a novel algorithm for generating local timed test patterns that specify the test's activity at every port. With this method, the authors believed the number of texts sent between testers would decrease. Furthermore, the authors demonstrated their strategy with a case study that employed a MapReduce architecture with multiple worker modules. They also defined the temporal characteristics of such systems in their specifications. They then used their algorithm to generate the associated timed local test sequences. They also demonstrated how to use their strategy in the context of MapReduce testing to distinguish faulty employees and make arrangements for their tasks to be moved to a better and healthier employee. Another author, Shakya and Smys [31], presented automatic testing to minimize the complications associated with manual testing. However, it was also an application that required variables in order to test the particular products. Automatic testing can be used to attain maximum product tuning. Their research study offered an optimized automated system software testing prototype as a hybrid model using evolutionary algorithms and ant colony efficiency to enhance software testing performance and completeness. Conventional designs such as artificial neural networks as well as particle swarm efficiency were also compared to their proposed approach to verify the consistency.

Among all aspects of the modern age, the internet has now become a crucial element. For daily life, several nations are becoming increasingly reliant on the internet. The growing internet demand has further increased the potential of harmful threats. In the digital world, cybersecurity is the most crucial aspect in combating all cyber dangers, breaches, and fraud due to the hazards of increasing cybersecurity. The growing cyberspace is greatly susceptible to the growing likelihood of endless cyber threats. Many machine learning algorithms are available for detecting cybersecurity risks to the very bottom of all advancements. Several authors have published their works in the area of distributed software testing but none of them have performed the MCDM-based approach to evaluate the various challenges of distributed software testing. In the current COVID-19 pandemic situation, there is a huge need to evaluate the different challenges of distributed software testing for efficient and timely software development. In this paper, we used the Fuzzy TOPSIS-based multiple-criteria decision-making approach to evaluate distributed software testing challenges for making a trustworthy process for testing in a distributed software development setting. The main goal of this research work is to assist the software development team to understand the different challenges of distributed software testing so that they can develop a trustworthy process or produce an efficient testing solution for distributed software testing amid the COVID-19 pandemic and to meet its users' requirements.

The rest of this study is organized as follows: Section 2 discusses several software testing challenges amid the COVID-19 pandemic. Section 3 presents the evaluation of distributed software testing challenges using the Fuzzy TOPSIS method. Section 4 discusses the findings of the empirical investigation in this research study. Finally, the discussion and conclusions are reported in Section 5.

2. Software Testing Challenges amid COVID-19 Pandemic

For most programmers, building software remotely may become difficult. With the chaos and confusion that characterizes every home, it is hard to create a productive work setting [18]. Numerous obstacles will reduce the team's efficiency, lowering the quality of the finished product and increasing production costs. Understanding the complexities of remote project management may help you overcome them quickly and effectively, ensuring that your application is fully operational.

2.1. Lack of Resources

Most IT businesses do not provide their employees with the systems and processes necessary to work from home, considering the remote complexity of developing software. Even in the midst of the coronavirus pandemic, those that do, have an excellent, remote team operating on software development. Many home-based software developers, designers, and testers, on the other hand, lack the tools necessary for rapid and efficient production. While having a personal computer at home is no longer uncommon for a remote software engineer, other systems and techniques ensure that the final product is completely safe. Hardware or software configurations, online tools, or establishing the whole framework for the software required or for testing are examples of these. As a result, one of the most important obstacles to remote app creation is the lack of a clear infrastructure.

2.2. Poor Communication

The absence of decent contact is one of the most important barriers to working remotely. When operating on a project externally, most application developers lack a strategic communication mechanism. Face-to-face contact also lacks the competence and commitment of digital communications. When working remotely, disruptions are normal, which can lead to communication breakdown or a lack of comprehension of a project's technical specifications and key performance indicators.

Ineffective management from the team leader or project coordinator may lead to a loss of influence, a lack of professional morale, and failure to meet the goals. Overcommunication, on the other hand, may be detrimental to the team's morale because it is interpreted as a sign of mistrust. Project leaders also seem to contact their diverse workforce every two hours to get a report. This reveals a lack of confidence, accountability, and collaboration in the process. The software development team members can lose concentration and enthusiasm if they have too many meetings during the day. As a result, having a strong communication strategy in place becomes critical, especially when managing software development projects remotely.

2.3. No Transparency in the Process

There is uncertainty among these team members due to a lack of transparency. They have no idea what is anticipated of them, why it is anticipated of them, or when it is anticipated of them. Workforce frustration can result from this. Since the entire process prevents the clarification of priorities, tasks, and obligations, one group member frequently tends to be overloaded with tasks compared with others. When performing tasks from home, the group's members can have trust challenges because they appear to miss deadlines or feel depressed. Trust takes time to develop, particularly when team members work from home. Efficient communication, teamwork, and inspiration can all aid in this situation.

2.4. Lack of Control

Weak management in remote application development is unsurprising, particularly when COVID-19 is spreading across the world and businesses are being shut down on a regular basis. Moreover, bringing the whole team onto the same network and at the same time can be difficult. When someone operates electronically, it is hard to keep track of anything.

2.5. Data Insecurity

When a team works remotely, they are likely to share ideas, inventions, and data among others in the same way as if they worked in the workplace. The distinction here is a deficiency of network protection and a failure to meet the company's data security requirements [9,10]. The team leaders and team participants should share and receive information in a professional way that does not damage the firm's or the customer's end result.

2.6. Delay in Project Delivery

As already said, among the most important flaws of operating remotely is a lack of motivation. At home, the workplace atmosphere is more relaxed, and disruption is common. It is challenging for software developers to stay focused and finish projects on schedule in this setting.

2.7. Wrong Mindset

The focus of the software testing team is frequently centred on determining the functional requirements of the software system rather than identifying problems in it. This prevents the team from discovering faults in the program. It is the project leader's responsibility to instil the idea that testing is performed to uncover flaws in the software system or programme under diverse scenarios, not to certify that it functions.

2.8. Incomplete Testing

Many businesses today generally favour verbal communication for basic or even complicated modifications in a software product's development. They do not keep the necessary documents, which includes the software's functional and non-functional objectives. This results in an absence of requirement collection for software testers because verbal communication could skip certain important project details.

2.9. Unstable Environment

The requirements definition may vary at times during the quick process of software development and deployment. Furthermore, developers modify the testing environment to address the detected faults or to add new capabilities to be tested. Additionally, when several testers are participating in the testing phase of a specific product, keeping track of all the modifications made by different members becomes difficult. If quality assurance teams are not kept up to date on all of these modifications, the software lifecycle will be disrupted. Ultimately, testing an application with inadequate information becomes challenging.

2.10. Inadequate Schedule

Testing is a time-consuming process. It has always been that way since the purpose of the test is to expose the system's flaws or deficiencies under diverse settings, not to demonstrate that it performs. Testing and development must work in conjunction. This ensures that any deficiencies or problems in a certain system's capability are brought to the attention of the development team and resolved as soon as possible. Unfortunately, most often, project leaders continue to postpone testing in the software development process. This provides much less time for complete testing, resulting in an unsatisfactory process timeline.

Thus, these are all the popular limitations in remote software development that one can overcome with the right approach. The following section will go through the best practises for remote software creation in order to effectively solve these issues.

3. Evaluation of Distributed Software Testing Challenges

Today's many advancements in technology have become quite familiar with a fast-growing digital technology. One of these modern technologies is termed the decision support system. It is built and applied so that decision makers can make variable and challenging decisions at planning, operational, and management levels. We have to evaluate the different challenges that are faced with the help of a decision support system in order to make a trustworthy process for distributed software testing.

3.1. Hierarchy for the Prioritization

Because of the demand to get a high-quality software product to market, the software must go through numerous rounds of testing to guarantee that any flaws are found and corrected as soon as possible. However, building test cases and conducting manual or software testing is not as effective as expected; testers frequently face numerous problems throughout the testing lifecycle, making the process quite stressful. Figure 2 shows the hierarchy for the prioritization of different testing challenges discussed earlier.

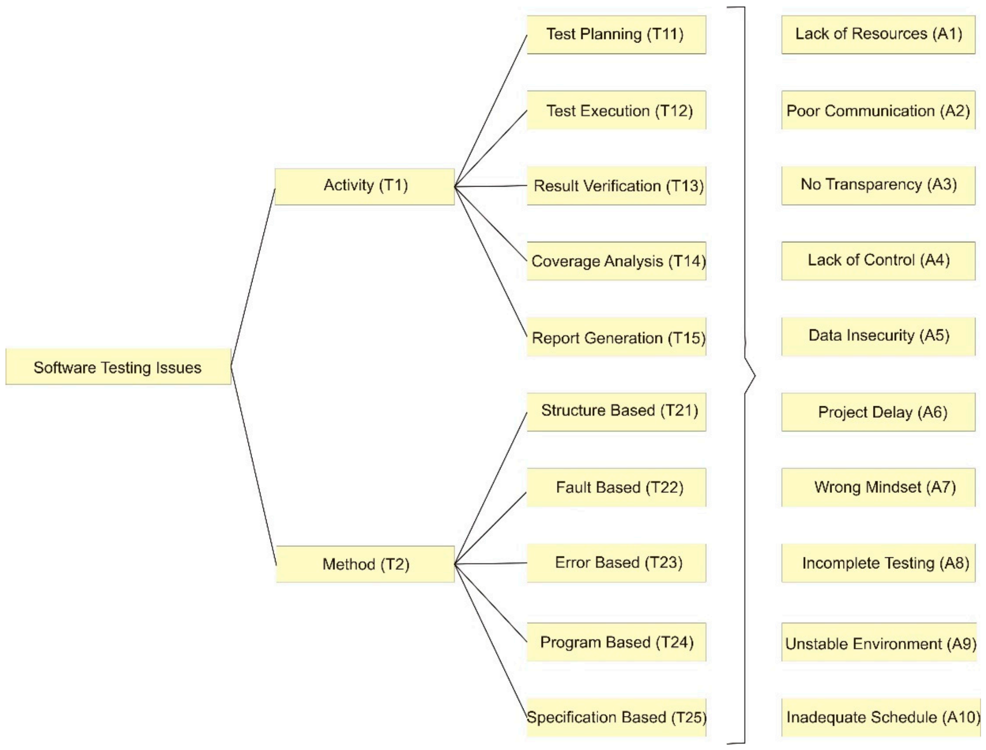

Figure 2. Hierarchy of the prioritization issue.

3.2. Fuzzy TOPSIS Method

Yoon and Hwang [32] presented the Technique for Order Preference by Similarity to Ideal Solution (TOPSIS) approach. Its primary principle is that the selected option should be the closest to the ideal solution while being the furthest away from the negative perfect solution [19]. This paper includes the TOPSIS approach. A positive ideal solution maximizes the benefit standards or qualities while decreasing the cost criteria or features, while a negative ideal solution decreases the benefit standards or features while increasing the cost criteria or qualities. Figure 3 shows the flow chart of the Fuzzy TOPSIS approach.

Several authors have used the Fuzzy TOPSIS method in their research study. Khan et al. [33] adapted the TOPSIS approach to a fuzzy setting replacing numeric language scales for scoring and weighing with triangular fuzzy numbers. Following this, several methods offered modifications to the Fuzzy TOPSIS technique. Ansari et al.'s [34] Fuzzy TOPSIS approach was used in the decision-making process in this work. This strategy is ideal for dealing with group decision-making problems in a fuzzy setting. The significance weights of multiple factors, as well as the evaluations of quantifiable criteria are treated as linguistic terms in this approach [22–24].

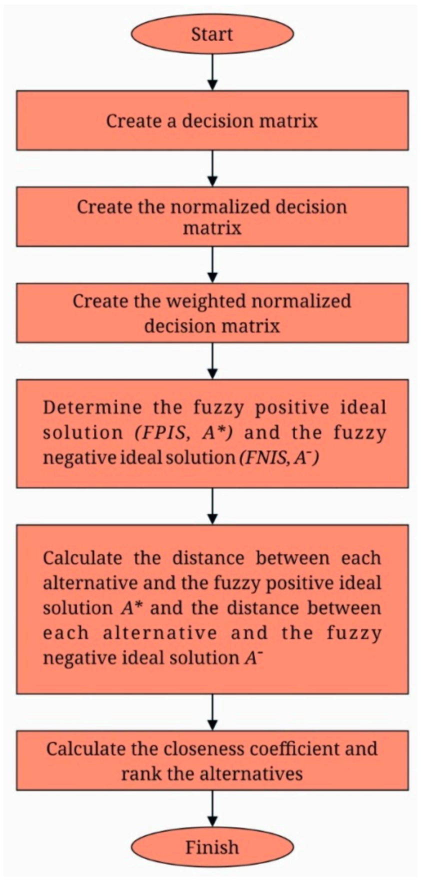
Start
Create a decision matrix
Create the normalized decision matrix
Create the weighted normalized decision matrix
Determine the fuzzy positive ideal solution (FPIS, A*) and the fuzzy negative ideal solution (FNIS, A⁻)
Calculate the distance between each alternative and the fuzzy positive ideal solution A* and the distance between each alternative and the fuzzy negative ideal solution A⁻
Calculate the closeness coefficient and rank the alternatives
Finish

Figure 3. Flow chart of Fuzzy TOPSIS approach.

We gathered multi-criteria expert opinions from various software organisations and academic institutions for this study. Based on defined parameters, we performed behavioural tests to assess the effect of multiple software testing challenges. To this goal, it is necessary to identify and characterise the problematic behaviour of significant groups of integration variables. Experts and academics in the field of software testing face a daunting task in mathematically measuring the consequences of various testing concerns. Therefore, we used Fuzzy TOPSIS, a well-developed and well-established decision-making approach, to achieve the goal of our research study. This method works well for comparing the information and overall level of various software testing challenges during the COVID-19 pandemic. To provide a more thorough result, we accumulated the opinions of 75 experts from different tech businesses and educational institutions. The information acquired from these experts was used to produce our investigation's findings. Activity and Method indicated as T1, T2 respectively indicate the different criteria for the evaluation process throughout the implementation phase. The quantitative findings of the current investigation are presented in Tables 1–4. Table 4 and Figure 4 show the Closeness Coefficients to the aspired level among the various alternatives.

The satisfaction degrees (CCi) of different alternatives are estimated as 0.933200, 0.852100, 0.811500, 0.963800, 0.979600, 0.721800, 0.885700, 0.936700, 0.944600, and 0.956800 for A1, A2, A3, A4, A5, A6, A7, A8, A9 and A10, respectively. As per the findings shown in Figure 4, the fifth alternative, Data insecurity, (A5) is the most challenging among several other software testing challenges. The current remote work environment introduces cyber security risks due to authorized users' distant access to vital IT infrastructure, the usage of collaboration platforms for software development team interactions, the accessibility of company data on remote devices, the lack of physical monitoring of vital IT equipment, and many more reasons.

Table 1. Subjective comprehension tests of linguistic evaluators.

Characteristics/Alternatives	A1	A2	A3	A4	A5	A6	A7	A8	A9	A10
T11	2.450000, 4.270000, 6.270000	1.450000, 3.070000, 4.910000	0.820000, 2.270000, 4.270000	0.910000, 2.450000, 4.450000	2.450000, 4.270000, 6.270000	2.450000, 4.270000, 6.270000	1.450000, 3.070000, 4.910000	0.820000, 2.270000, 4.270000	0.910000, 2.450000, 4.450000	2.450000, 4.270000, 6.270000
T12	2.090000, 3.730000, 5.730000	2.820000, 4.640000, 6.640000	1.910000, 3.730000, 5.730000	2.820000, 4.640000, 6.640000	1.910000, 3.730000, 5.730000	2.090000, 3.730000, 5.730000	2.820000, 4.640000, 6.640000	1.910000, 3.730000, 5.730000	2.820000, 4.640000, 6.640000	1.910000, 3.730000, 5.730000
T13	3.000000, 4.820000, 6.820000	3.000000, 5.000000, 7.140000	2.180000, 4.090000, 6.140000	1.820000, 3.730000, 5.730000	1.640000, 3.550000, 5.550000	3.000000, 4.820000, 6.820000	3.000000, 5.000000, 7.140000	2.180000, 4.090000, 6.140000	1.820000, 3.730000, 5.730000	1.640000, 3.550000, 5.550000
T14	5.120000, 7.140000, 8.720000	3.150000, 5.150000, 6.910000	2.820000, 4.640000, 6.640000	1.550000, 3.180000, 5.180000	1.450000, 3.180000, 5.180000	5.120000, 7.140000, 8.720000	3.150000, 5.150000, 6.910000	2.820000, 4.640000, 6.640000	1.550000, 3.180000, 5.180000	1.450000, 3.180000, 5.180000
T15	4.280000, 6.370000, 8.370000	2.450000, 4.450000, 6.450000	2.910000, 4.640000, 6.550000	1.450000, 3.000000, 4.910000	1.180000, 2.820000, 4.820000	4.280000, 6.370000, 8.370000	2.450000, 4.450000, 6.450000	2.910000, 4.640000, 6.550000	1.450000, 3.000000, 4.910000	1.180000, 2.820000, 4.820000
T21	4.270000, 6.270000, 8.140000	2.820000, 4.820000, 6.820000	3.180000, 5.180000, 7.100000	1.450000, 3.070000, 4.910000	0.820000, 2.270000, 4.270000	4.270000, 6.270000, 8.140000	2.820000, 4.820000, 6.820000	3.180000, 5.180000, 7.100000	1.450000, 3.070000, 4.910000	0.820000, 2.270000, 4.270000
T22	5.360000, 7.360000, 9.120000	3.730000, 5.730000, 7.550000	2.450000, 4.450000, 6.450000	0.910000, 2.450000, 4.450000	2.450000, 4.270000, 6.270000	5.360000, 7.360000, 9.120000	3.730000, 5.730000, 7.550000	2.450000, 4.450000, 6.450000	0.910000, 2.450000, 4.450000	2.450000, 4.270000, 6.270000
T23	2.820000, 4.640000, 6.640000	1.910000, 3.730000, 5.730000	1.180000, 2.820000, 4.820000	4.280000, 6.370000, 8.370000	1.450000, 3.070000, 4.910000	4.640000, 6.640000, 8.550000	3.000000, 5.000000, 7.140000	2.180000, 4.090000, 6.140000	2.820000, 4.640000, 6.640000	1.910000, 3.730000, 5.730000
T24	3.000000, 5.000000, 7.140000	2.180000, 4.090000, 6.140000	0.820000, 2.270000, 4.270000	4.270000, 6.270000, 8.140000	2.820000, 4.640000, 6.640000	3.120000, 5.000000, 7.140000	2.450000, 4.450000, 6.450000	3.550000, 5.550000, 7.450000	1.820000, 3.730000, 5.730000	1.640000, 3.550000, 5.550000
T25	3.550000, 5.550000, 7.450000	1.820000, 3.730000, 5.730000	2.450000, 4.270000, 6.270000	5.360000, 7.360000, 9.120000	3.000000, 5.000000, 7.140000	5.360000, 7.360000, 9.090000	2.640000, 4.640000, 6.640000	2.900000, 4.800000, 6.700000	2.820000, 4.640000, 6.640000	2.550000, 4.450000, 6.450000

Table 2. Normalized fuzzy-decision matrix.

Characteristics/Alternatives	A1	A2	A3	A4	A5	A6	A7	A8	A9	A10
T11	0.560000, 0.780000, 0.950000	0.410000, 0.680000, 0.910000	0.370000, 0.620000, 0.890000	0.230000, 0.470000, 0.780000	0.220000, 0.490000, 0.800000	0.300000, 0.530000, 0.790000	0.178100, 0.377100, 0.603100	0.118600, 0.328500, 0.617900	0.210000, 0.460000, 0.730000	0.120000, 0.350000, 0.660000
T12	0.460000, 0.690000, 0.910000	0.320000, 0.580000, 0.850000	0.390000, 0.620000, 0.870000	0.210000, 0.450000, 0.730000	0.180000, 0.430000, 0.740000	0.260000, 0.470000, 0.720000	0.346400, 0.570000, 0.815700	0.276400, 0.539700, 0.829200	0.130000, 0.360000, 0.670000	0.370000, 0.660000, 0.970000
T13	0.460000, 0.680000, 0.890000	0.370000, 0.630000, 0.900000	0.420000, 0.690000, 0.950000	0.210000, 0.460000, 0.730000	0.120000, 0.350000, 0.660000	0.370000, 0.600000, 0.860000	0.368500, 0.614200, 0.877100	0.315400, 0.591800, 0.888500	0.420000, 0.690000, 1.000000	0.290000, 0.570000, 0.880000
T14	0.580000, 0.800000, 1.000000	0.490000, 0.750000, 1.000000	0.320000, 0.590000, 0.860000	0.130000, 0.360000, 0.670000	0.370000, 0.660000, 0.970000	0.490000, 0.740000, 0.980000	0.436100, 0.681800, 0.927500	0.263300, 0.539700, 0.829200	0.270000, 0.560000, 0.860000	0.250000, 0.550000, 0.860000
T15	0.500000, 0.720000, 0.930000	0.390000, 0.660000, 0.940000	0.290000, 0.540000, 0.820000	0.420000, 0.690000, 1.000000	0.290000, 0.570000, 0.880000	0.320000, 0.560000, 0.810000	0.356200, 0.589600, 0.823000	0.408100, 0.671400, 0.960900	0.420000, 0.690000, 1.000000	0.390000, 0.700000, 1.000000
T21	0.340000, 0.540000, 0.780000	0.320000, 0.580000, 0.850000	0.470000, 0.740000, 1.000000	0.270000, 0.560000, 0.860000	0.250000, 0.550000, 0.860000	0.490000, 0.740000, 1.000000	0.346400, 0.570000, 0.815700	0.369000, 0.643900, 0.933400	0.469200, 0.698400, 0.917700	0.158900, 0.337700, 0.538300
T22	0.580000, 0.800000, 0.990000	0.340000, 0.610000, 0.870000	0.380000, 0.640000, 0.890000	0.420000, 0.690000, 1.000000	0.390000, 0.700000, 1.000000	0.400000, 0.650000, 0.890000	0.574500, 0.770200, 1.000000	0.408100, 0.697500, 0.986900	0.468200, 0.687500, 0.892500	0.394900, 0.649800, 0.929900
T23	0.309000, 0.508000, 0.728000	0.209400, 0.408990, 0.628200	0.129300, 0.309200, 0.528500	0.469200, 0.698400, 0.917700	0.158900, 0.337700, 0.538300	0.089900, 0.248900, 0.468200	0.356200, 0.589600, 0.823000	0.455800, 0.745200, 1.000000	0.587700, 0.807000, 1.000000	0.420100, 0.700200, 1.000000
T24	0.328900, 0.548200, 0.782800	0.252900, 0.494000, 0.813200	0.110000. 0.304600, 0.573100	0.468200, 0.687500, 0.892500	0.394900, 0.649800, 0.929900	0.241400, 0.471500, 0.724300	0.346400, 0.570000, 0.815700	0.354500, 0.643900, 0.933400	0.460000, 0.690000, 0.910000	0.320000, 0.580000, 0.850000
T25	0.389200, 0.608500, 0.816800	0.241000, 0.494000, 0.758900	0.328800, 0.573100, 0.841600	0.587700, 0.807000, 1.000000	0.420100, 0.700200, 1.000000	0.275600, 0.517000, 0.776200	0.524500, 0.770200, 1.000000	0.408100, 0.697500, 0.986900	0.460000, 0.680000, 0.890000	0.370000, 0.630000, 0.900000

Table 3. Weighted normalized fuzzy-decision matrix.

Characteristics/Alternatives	**A1**	**A2**	**A3**	**A4**	**A5**	**A6**	**A7**	**A8**	**A9**	**A10**
T11	0.001310	0.000960	0.000870	0.000540	0.000520	0.000700	0.000420	0.000280	0.000500	0.001500
	0.002060	0.001800	0.001640	0.001240	0.001290	0.001400	0.001000	0.000870	0.001900	0.003600
	0.004790	0.004590	0.004490	0.003940	0.004040	0.003990	0.003040	0.003120	0.006300	0.009100
T12	0.000300	0.000200	0.000300	0.000200	0.000100	0.000200	0.000300	0.000200	0.111900	0.077300
	0.000600	0.000500	0.000600	0.000400	0.000400	0.000400	0.000500	0.000500	0.200200	0.165400
	0.001700	0.001600	0.001600	0.001400	0.001400	0.001300	0.001500	0.001500	0.347400	0.305700
T13	0.010000	0.008000	0.009100	0.004600	0.002600	0.008000	0.008000	0.006800	0.003700	0.003400
	0.018800	0.017400	0.019100	0.012700	0.009700	0.016600	0.017000	0.016400	0.008300	0.008100
	0.036800	0.037200	0.039200	0.030200	0.027300	0.035500	0.036200	0.036700	0.013100	0.013100
T14	0.002300	0.002000	0.001300	0.000500	0.001500	0.002000	0.001800	0.001100	0.053100	0.049300
	0.004300	0.004000	0.003200	0.001900	0.003600	0.004000	0.003700	0.002900	0.105800	0.107300
	0.009400	0.009400	0.008100	0.006300	0.009100	0.009200	0.008700	0.007800	0.193900	0.193900
T15	0.133200	0.103900	0.077300	0.111900	0.077300	0.085200	0.094900	0.108700	0.007800	0.002700
	0.209000	0.191500	0.156700	0.200200	0.165400	0.162500	0.171100	0.194800	0.015600	0.007500
	0.323100	0.326600	0.284900	0.347400	0.305700	0.281400	0.285900	0.333800	0.026100	0.015300
T21	0.004600	0.004300	0.006400	0.003700	0.003400	0.006600	0.004700	0.005000	0.044500	0.037500
	0.008000	0.008600	0.010900	0.008300	0.008100	0.010900	0.008400	0.009500	0.082800	0.078300
	0.011900	0.013000	0.015200	0.013100	0.013100	0.015200	0.012400	0.014200	0.148200	0.154400
T22	0.073400	0.043000	0.048100	0.053100	0.049300	0.050600	0.072700	0.051600	0.177700	0.127000
	0.122600	0.093500	0.098100	0.105800	0.107300	0.099600	0.118100	0.106900	0.279500	0.242500
	0.192000	0.168700	0.172600	0.193900	0.193900	0.172600	0.193900	0.191400	0.369400	0.369400
T23	0.005200	0.003500	0.002200	0.007800	0.002700	0.001500	0.005900	0.007600	0.000300	0.000200
	0.011300	0.009100	0.006900	0.015600	0.007500	0.005600	0.013200	0.016600	0.000600	0.000500
	0.020700	0.017900	0.015000	0.026100	0.015300	0.013300	0.023400	0.028500	0.001700	0.001600
T24	0.031200	0.024000	0.010400	0.044500	0.037500	0.022900	0.032900	0.033700	0.010000	0.008000
	0.066100	0.059500	0.036700	0.082800	0.078300	0.056800	0.068700	0.077600	0.018800	0.017400
	0.130000	0.135000	0.095200	0.148200	0.154400	0.120300	0.135400	0.155000	0.036800	0.037200
T25	0.117700,	0.072900	0.099400	0.177700	0.127000	0.083300	0.158600	0.123400	0.002300	0.002000
	0.210700,	0.171100	0.198500	0.279500	0.242500	0.179000	0.266700	0.241500	0.004300	0.004000
	0.301700	0.280300	0.310900	0.369400	0.369400	0.286700	0.369400	0.364500	0.009400	0.009400

Table 4. Closeness coefficients to the aspired level among the various alternatives.

Alternatives (HS)	di+	di−	Gap Degree of CCi+	Satisfaction Degree
A1	0.838400	0.060000	0.066700	0.933200
A2	0.806700	0.140000	0.147800	0.852100
A3	0.775100	0.180000	0.188400	0.811500
A4	0.916900	0.040000	0.041000	0.963800
A5	0.921300	0.030000	0.031500	0.979600
A6	0.700400	0.240000	0.255200	0.721800
A7	0.860300	0.110000	0.113300	0.885700
A8	0.901300	0.070000	0.072000	0.936700
A9	0.912200	0.050000	0.051900	0.944600
A10	0.915600	0.050000	0.051700	0.956800

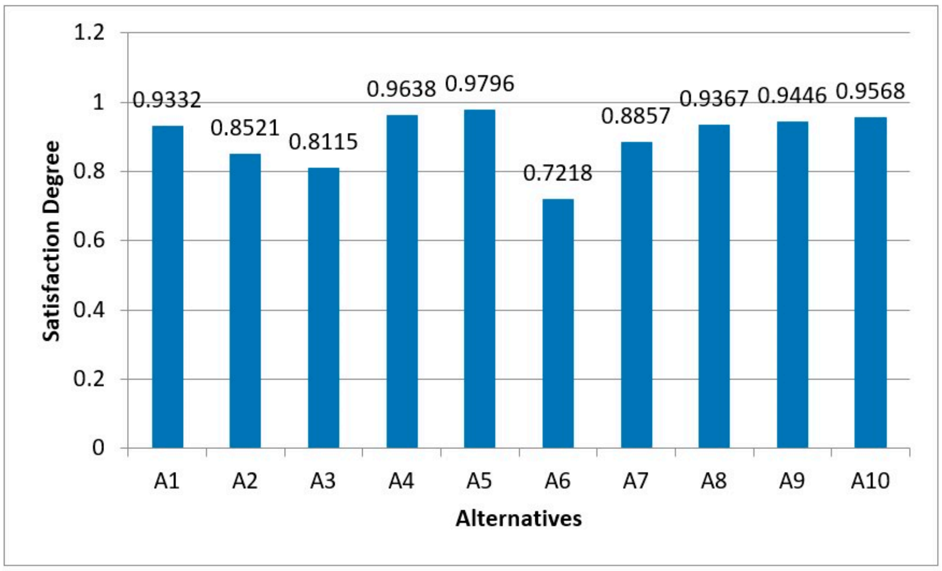

Figure 4. Graphical representation of satisfaction degree to the aspired level among the various alternatives.

3.3. Comparative Analysis

We also applied a variety of symmetrical methodologies to verify the effectiveness of the study's results in this investigation. In this work, we employed the Fuzzy TOPSIS approach to prioritize the various software testing challenges. The data collecting and assessment methodology for that dataset is similar to the classical TOPSIS procedure in Fuzzy TOPSIS. As a result, the Fuzzy TOPSIS approach requires fuzzification and defuzzification. As a result, data for Fuzzy TOPSIS are obtained in their initial statistical values and then transformed into fuzzy values. Moreover, the comparative analysis confirms the findings of the Fuzzy TOPSIS, thus enhancing the validity of the ranking among the ten alternatives. Table 5 and Figure 5 describe the significant differences in the Fuzzy TOPSIS, Classical TOPSIS, Delphi-TOPSIS, and Fuzzy-Delphi-TOPSIS findings.

Table 5. Comparative Results.

Approaches/Alternatives	A1	A2	A3	A4	A5	A6	A7	A8	A9	A10
Fuzzy TOPSIS	0.933200	0.852100	0.811500	0.963800	0.979600	0.721800	0.885700	0.936700	0.944600	0.956800
Classical TOPSIS	0.938800	0.853000	0.807900	0.963500	0.978700	0.720900	0.881500	0.925800	0.951800	0.967900
Delphi-TOPSIS	0.933400	0.852110	0.807740	0.963490	0.978740	0.720810	0.881480	0.925770	0.951810	0.967890
Fuzzy-Delphi-TOPSIS	0.932100	0.855000	0.809500	0.964500	0.979200	0.721400	0.885000	0.926100	0.945800	0.966100

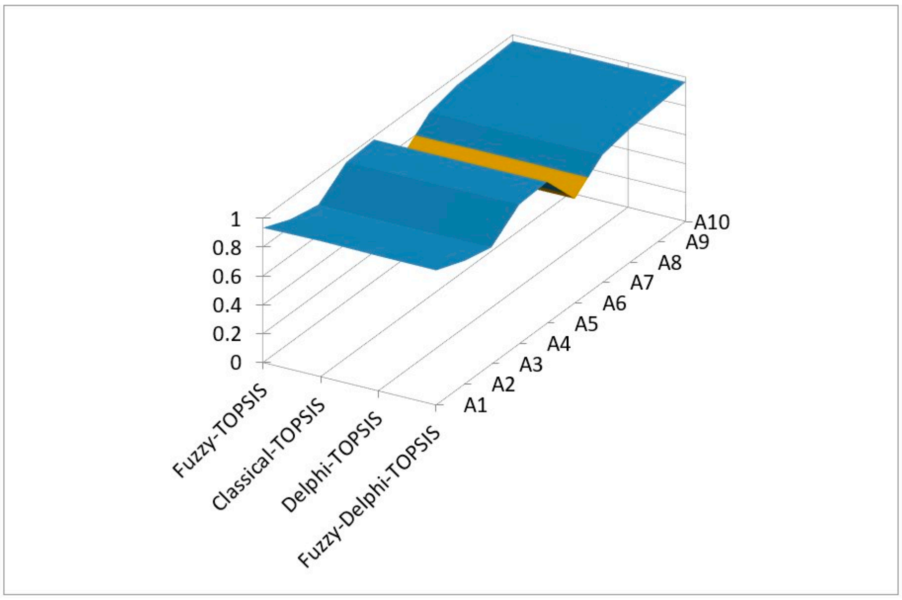

Figure 5. Graphical representation of comparative results.

3.4. Validation of the Results

Sensitivity analysis, as a strategy or instrument, plays an important role in justifying a research study using the same data as well as approach. It is used to determine the consequence or impact of a parameter on a parameter when the numbers of the parameter are changed. This method aids researchers in correlating their findings [35]. The weights derived by the fuzzy-based TOPSIS were treated as parameters. The weight of the attribute value is shifted in each experiment, while the weights of some other characteristics stay unchanged. At the top level of the hierarchical attribute tree, ten characteristics were adopted for this research. As a consequence, ten experiments were conducted out, one for each separately, and also the obtained values are shown in Table 6 and Figure 6.

Table 6 as well as Figure 6 shows the visual illustration of sensitivity experiment results. In the very same table, the weight values of the characteristics are also depicted as initial weights. From Expt.1 to Expt.10, ten experiments were conducted. The final outcome of every experiment after estimating the satisfaction level is shown in Table 6. Furthermore, the validation procedure is performed in objective manner with the assistance of [21–25]. Statistical analysis was performed on the results of the sensitivity assessment to ensure the consequences. The validation procedure [26–29] used by the researchers in this study to determine the numerical mean x. For every investigation, the mean x is estimated, as well as x is the sample average, measured as the cumulative of all observed data divided by the aggregate number described in Equation (1) as shows:

$$\overline{x} = \frac{1}{n}\sum_{i=1}^{n} x_i \tag{1}$$

With the support of [30–33], the alternative A1 receives the maximum importance in in all experiments. Additionally, A4 got lowest values in three experiments Expt-2, Expt-4, and in Expt-8 whereas A6 got lowest values in Expt-1, Expt-3, Exp-5, Exp-6, Exp-7, Exp-9 and Exp-10. The analysis of the results demonstrates that the alternative scores are sensitive to the weights.

Table 6. Variations in results.

Experiments	Weights/Alternatives	A1	A2	A3	A4	A5	A6	A7	A8	A9	A10
	Original Weights	0.933200	0.852100	0.811500	0.963800	0.979600	0.721800	0.885700	0.936700	0.944600	0.956800
Expt-1	S11	0.933524	0.852556	0.885644	0.968567	0.979647	0.721745	0.885777	0.936774	0.944471	0.956779
Expt-2	S12	0.933124	0.852745	0.811784	0.963774	0.979789	0.721999	0.885777	0.936852	0.944854	0.956857
Expt-3	S21	0.933789	0.852654	0.811456	0.963789	0.979741	0.721857	0.885859	0.936965	0.944745	0.956745
Expt-4	S22	0.933745	0.852963	0.811951	0.963957	0.979852	0.721358	0.885854	0.936745	0.944859	0.956958
Expt-5	S23	0.933754	0.852748	0.811754	0.963778	0.979785	0.721254	0.885745	0.936875	0.944425	0.956457
Expt-6	S31	0.963789	0.979741	0.721857	0.963789	0.979741	0.721857	0.885859	0.936965	0.944745	0.956745
Expt-7	S32	0.963957	0.979852	0.721358	0.963957	0.979852	0.721358	0.885854	0.936745	0.944859	0.956958
Expt-8	S33	0.963789	0.979741	0.721857	0.963789	0.979741	0.721857	0.885859	0.936965	0.944745	0.956745
Expt-9	S41	0.963957	0.979852	0.721358	0.963957	0.979852	0.721358	0.885854	0.936745	0.944859	0.956958
Expt-10	S42	0.933747	0.852745	0.811748	0.963778	0.979748	0.721968	0.885885	0.936748	0.944778	0.956758

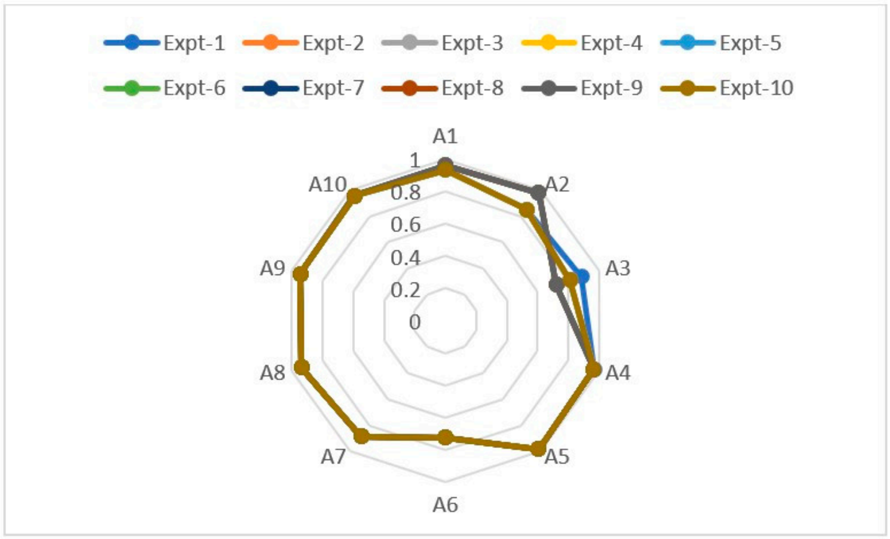

Figure 6. Experiments of sensitivity analysis of ten alternatives.

4. Discussion

COVID-19's risks are having a substantial influence on the technology sector, impacting the supply of materials, interrupting the electrical goods supply chain, as well as posing inflationary potential losses to product lines. More favourably, the interruption has accelerated working from home and an accelerated focus on assessing and de-risking the entire value chain [36–38]. Industries tightened their finances in response to the pandemic; however, as the emergency dragged on with no finish in sight, and realization set in, they modified their business strategies and also broadened out into areas where they had no expertise in the field. These transformations, combined with the need to find new company objectives, have accelerated the interest in acquiring and implementing QA as well as software testing global sourcing. The requirement to do so rapidly has become a primary consideration, and the previous reluctance to invest money has been conquered by the demand to preserve quality in the face of such unpredictability. With a vested financial interest, leading software testing firms have intently gone along with this transformation, quickly learning that the application of testing solutions is the key element in the achievement of this global transformation [39,40].

As a record majority of software firms and organizations across the country conduct business over the internet, software testing is vital to maintain accuracy, pace, and profit growth. The pandemic expanded the limits of firms, making the effectiveness of software, online platforms, as well as other technology, more essential than before. As even more industries want to entrust their software testing as a consequence of COVID-19, they can indeed be optimistic. According to the empirical investigation findings in this paper, data insecurity is the most significant among many other software testing issues. The recent remote work setting has incorporated cyber security consequences as a result of authorized users' virtual access to critical IT architecture, the use of collaborative effort platforms for project management relationships, the ease of access to company information on remote systems, and the absence of physical surveillance of critical IT equipment, as well as many other factors.

Software firms must ensure that their remote employees understand the security procedures, as well as what represents a breach of security and what does not in order to avert prospective data security tragedies. A data security strategy that includes a collection of safety tips can be useful in this regard. Furthermore, organizations should recognize employees who follow the strategy. Simultaneously, they must focus on ensuring that all staff members are prepared for the impacts of a lack of compliance.

5. Conclusions

The COVID-19 disease outbreak has created panic around the world, causing deaths. Business organizations have come to a halt as global downturns continue to prevail. Because lockdowns as well as social separating have become more common around the world, many businesses have resorted to working from home arrangements to keep businesses from stopping. A major challenge for IT businesses, therefore, has been to clear the path for remote software production. The significance of software testing cannot be underestimated. Since it enhances performance and accuracy, software testing is an important aspect of the software development process. The key advantage of testing is the detection and subsequent elimination of mistakes. Testing, on the other hand, assists software developers in comparing real and predicted outcomes in order to enhance efficiency. If software is created without being tested, it can be ineffective or even harmful to consumers. As a result, a tester must be responsible for maintaining the software's stability and making it easy to utilize in real-world scenarios. The Fuzzy TOPSIS method used in this paper is effective at prioritizing the different identified software testing challenges in this pandemic situation.

This research has some limitations, which are important to declare in order to manage the distributed software testing issues efficiently. One of the limitations is that this research recognizes 10 issues of distributed software testing that were identified through a literature review and there may be more or less in practice. In the case of more or fewer issues, the study results could be even more beneficial. One more limitation is the specialists' points of view, which may be individually biased, thus having an impact on the ultimate research findings. The goal for the future is to incorporate this robust tactic on digital platforms. We will anticipate other MCDM approaches for the assessment of multiple issues in distributed software testing when we expand on this work. Moreover, we will also look at some other weight estimation techniques that could be used with TOPSIS, such as the entropy technique as well as least square techniques. Furthermore, different options as well as criteria can be considered in future work.

Author Contributions: All the authors have contributed equally to the manuscript. All authors have read and agreed to the published version of the manuscript.

Funding: Researchers Supporting Project number (TURSP-2020/231), Taif University, Taif, Saudi Arabia.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Not applicable.

Acknowledgments: This research was supported by the Taif University Researchers Supporting Project number (TURSP-2020/231), Taif University, Taif, Saudi Arabia.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. De Michell, G.; Gupta, R. K Hardware/Software Co-Design. *Proc. IEEE* **1997**, *85*, 349–365. [CrossRef]
2. Dustin, E.; Rashka, J.; Paul, J. *Automated Software Testing: Introduction, Management, and Performance*; Addison-Wesley Professional: Boston, MA, USA, 1999.
3. Murphy, C.; Shen, K.; Kaiser, G. Automatic system testing of programs without test oracles. In Proceedings of the Eighteenth International Symposium on Software Testing and Analysis, Chicago, IL, USA, 19–23 July 2009; pp. 189–200.
4. El-Far, I.K.; Whittaker, J.A. *Model-Based Software Testing*; Wiley: Hoboken, NJ, USA, 2002.
5. Myers, A. Introducing Redpoint's Software Testing Landscape. Available online: https://medium.com/memory-leak/introducing-redpoints-software-testing-landscape-3c5615f7eeae (accessed on 11 May 2021).
6. AlHakami, W.; Binmahfoudh, A.; Baz, A.; AlHakami, H.; Ansari, T.J.; Khan, R.A. Atrocious Impinging of COVID-19 Pandemic on Software Development Industries. *Comput. Syst. Sci. Eng.* **2021**, *36*, 323–338. [CrossRef]
7. Technavio. $34.49 Billion Growth in Software Testing Services Market 2020–2024: Insights and Products Offered by Major VENDORS: TECHNAVIO. Available online: https://www.prnewswire.com/news-releases/-34-49-billion-growth-in-software-testing-services-market-2020-2024--insights-and-products-offered-by-major-vendors--technavio-301242103.html (accessed on 11 May 2021).
8. Khan, M.E.; Khan, F. A comparative study of white box, black box and grey box testing techniques. *Int. J. Adv. Comput. Sci. Appl.* **2012**, *3*. [CrossRef]
9. Frankl, P.G.; Hamlet, R.G.; Littlewood, B.; Strigini, L. Evaluating testing methods by delivered reliability [software]. *IEEE Trans. Softw. Eng.* **1998**, *24*, 586–601. [CrossRef]
10. Zarour, M.; Alenezi, M.; Ansari, T.J.; Pandey, A.K.; Ahmad, M.; Agrawal, A.; Kumar, R.; Khan, R.A. Ensuring data integrity of healthcare information in the era of digital health. *Health Technol. Lett.* **2021**, *8*, 66–77. [CrossRef] [PubMed]
11. Planning, S. *The Economic Impacts of Inadequate Infrastructure for Software Testing*; National Institute of Standards and Technology: Gaithersburg, MD, USA, 2002.
12. Amershi, S.; Begel, A.; Bird, C.; DeLine, R.; Gall, H.; Kamar, E.; Nagappan, N.; Nushi, B.; Zimmermann, T. Software Engineering for Machine Learning: A Case Study. In Proceedings of the 2019 IEEE/ACM 41st International Conference on Software Engineering: Software Engineering in Practice (ICSE-SEIP), Montreal, QC, Canada, 27 May 2019; pp. 291–300.
13. Zheng, W.; Bai, Y.; Che, H. A computer-assisted instructional method based on machine learning in software testing class. *Comput. Appl. Eng. Educ.* **2018**, *26*, 1150–1158. [CrossRef]
14. Gunawan, J.; Juthamanee, S.; Aungsuroch, Y. Current Mental Health Issues in the Era of COVID-19. *Asian J. Psychiatry* **2020**, *51*, 102103. [CrossRef]
15. Hwang, C.-L.; Yoon, K. Basic Concepts and Foundations. In *Computer-Aided Transit Scheduling*; Springer Science and Business Media LLC: Berlin/Heidelberg, Germany, 1981; pp. 16–57.
16. Yoon, K. *Systems Selection by Multiple Attribute Decision Making*; Kansas State University: Manhattan, KS, USA, 1980; pp. 1–20.
17. Chen, C.-T. Extensions of the TOPSIS for group decision-making under fuzzy environment. *Fuzzy Sets Syst.* **2000**, *114*, 1–9. [CrossRef]
18. Collins, E.; Macedo, G.; Maia, N.; Dias-Neto, A. An Industrial Experience on the Application of Distributed Testing in an Agile Software Development Environment. In Proceedings of the 2012 IEEE Seventh International Conference on Global Software Engineering, Washington, DC, USA, 27–30 August 2012; pp. 190–194.
19. Eassa, F.E.; Osterweil, L.J.; Fadel, M.A.; Sandokji, S.; Ezz, A. DTTAS: A Dynamic Testing Tool for Agent-based Systems. *Pensee J.* **2014**, *76*, 147–165.
20. Di Lucca, G.A.; Fasolino, A.R. Testing Web-based applications: The state of the art and future trends. *Inf. Softw. Technol.* **2006**, *48*, 1172–1186. [CrossRef]
21. Azzouzi, S.; Benattou, M.; Charaf, M.E.H. A temporal agent based approach for testing open distributed systems. *Comput. Stand. Interfaces* **2015**, *40*, 23–33. [CrossRef]
22. Shaukat, K.; Luo, S.; Varadharajan, V.; Hameed, I.A.; Chen, S.; Liu, D.; Li, J. Performance Comparison and Current Challenges of Using Machine Learning Techniques in Cybersecurity. *Energies* **2020**, *13*, 2509. [CrossRef]
23. Shaukat, K.; Faisal, A.; Masood, R.; Usman, A.; Shaukat, U. Security quality assurance through penetration testing. In Proceedings of the 2016 19th International Multi-Topic Conference (INMIC), Islamabad, Pakistan, 5–6 December 2016; pp. 1–6.
24. Liu, X.; Deng, R.; Yang, Y.; Tran, H.N.; Zhong, S. Hybrid privacy-preserving clinical decision support system in fog–cloud computing. *Future Gener. Comput. Syst.* **2018**, *78*, 825–837. [CrossRef]
25. Shaukat, K.; Shaukat, U.; Feroz, F.; Kayani, S.; Akbar, A. Taxonomy of automated software testing tools. *Int. J. Comput. Sci. Innov.* **2015**, *1*, 7–18.
26. Dar, K.S.; Tariq, S.; Akram, H.J.; Ghani, U.; Ali, S. Web Based Programming Languages that Support Selenium Testing. *Int. J. Foresight Innov. Policy* **2015**, *2015*, 21–25.
27. Corny, J.; Rajkumar, A.; Martin, O.; Dode, X.; Lajonchère, J.P.; Billuart, O.; Buronfosse, A. A machine learning–based clinical decision support system to identify prescriptions with a high risk of medication error. *J. Am. Med. Inform. Assoc.* **2020**, *27*, 1688–1694. [CrossRef]
28. Anooj, P. Clinical decision support system: Risk level prediction of heart disease using weighted fuzzy rules. *J. King Saud Univ.-Comput. Inf. Sci.* **2012**, *24*, 27–40. [CrossRef]

29. Sher, B. Challenges to Project Management in Distributed Software Development: A Systematic Literature Review. In *Evolving Software Processes*; Wiley: Hoboken, NJ, USA, 2022; pp. 241–251.
30. Hsaini, S.; Azzouzi, S.; Charaf, M.E.H. A temporal based approach for MapReduce distributed testing. *Int. J. Parallel. Emergent Distrib. Syst.* **2021**, *36*, 293–311. [CrossRef]
31. Shakya, S.; Smys, S. Reliable automated software testing through hybrid optimization algorithm. *J. Ubiquitous Comput. Commun. Technol. (UCCT)* **2020**, *2*, 126–135.
32. Yoon, K.P.; Hwang, C.L. *Multiple Attribute Decision Making: An Introduction*; Sage Publications: Thousand Oaks, CA, USA, 1995.
33. Khan, S.A.; Alenezi, M.; Agrawal, A.; Kumar, R.; Khan, R.A. Evaluating Performance of Software Durability through an Integrated Fuzzy-Based Symmetrical Method of ANP and TOPSIS. *Symmetry* **2020**, *12*, 493. [CrossRef]
34. Ansari, M.T.J.; Al-Zahrani, F.A.; Pandey, D.; Agrawal, A. A fuzzy TOPSIS based analysis toward selection of effective security requirements engineering approach for trustworthy healthcare software development. *BMC Med. Inform. Decis. Mak.* **2020**, *20*, 236. [CrossRef] [PubMed]
35. Alzhrani, F.A. Evaluating the usable-security of healthcare software through unified technique of fuzzy logic, ANP and TOPSIS. *IEEE Access* **2020**, *8*, 109905–109916. [CrossRef]
36. Ansari, M.T.J.; Agrawal, A.; Khan, R. DURASec: Durable Security Blueprints for Web-Applications Empowering Digital India Initiative. *ICST Trans. Scalable Inf. Syst.* **2022**. [CrossRef]
37. Ansari, M.T.J.; Khan, N.A. Worldwide COVID-19 Vaccines Sentiment Analysis through Twitter Content. *Electron. J. Gen. Med.* **2021**, *18*, 1–10. [CrossRef]
38. Alosaimi, W.; Ansari, M.T.J.; Alharbi, A.; Alyami, H.; Seh, A.; Pandey, A.; Agrawal, A.; Khan, R. Evaluating the Impact of Different Symmetrical Models of Ambient Assisted Living Systems. *Symmetry* **2021**, *13*, 450. [CrossRef]
39. Alyami, H.; Nadeem, M.; Alharbi, A.; Alosaimi, W.; Ansari, M.T.J.; Pandey, D.; Kumar, R.; Khan, R.A. The Evaluation of Software Security through Quantum Computing Techniques: A Durability Perspective. *Appl. Sci.* **2021**, *11*, 11784. [CrossRef]
40. Ansari, M.T.J.; Pandey, D.; Alenezi, M. STORE: Security Threat Oriented Requirements Engineering Methodology. *J. King Saud Univ.-Comput. Inf. Sci.* **2018**, *34*, 191–203. [CrossRef]