Hindawi
Mathematical Problems in Engineering
Volume 2022, Article ID 6153744, 14 pages
https://doi.org/10.1155/2022/6153744

Research Article

Large-Scale Agile Transformations for Software Quality Assurance: An Empirical Case Study from Pakistan

Kamran Wadood,[1] Natasha Nigar,[2] Muhammad Kashif Shahzad,[1] Shahid Islam,[2] Abdul Jaleel,[2] and Douhadji Abalo[3]

[1]*Power Information Technology Company (PITC), Ministry of Energy, Power Division, Government of Pakistan, Lahore 54890, Pakistan*
[2]*Department of Computer Science (RCET Campus, GRW), University of Engineering and Technology, Lahore 52250, Pakistan*
[3]*University of Lomé, P.O. Box 1515, Lomé, Togo*

Correspondence should be addressed to Douhadji Abalo; douhadjiabalo@gmail.com

Received 30 April 2022; Revised 29 July 2022; Accepted 5 August 2022; Published 29 August 2022

Academic Editor: Ghous Ali

Software quality plays an important role in the easy and quick adoption of a software product by the end users. Agile methods have proven to play an effective role in ensuring software quality; however, the public sector is hesitant in its adoption. This study evaluates the adoption of agile methods in the public sector in-house software development and capitalization of potential benefits. A quantitative approach is used with 28 survey questionnaires based on budget, time, quick debugging and correction, maintenance, easy testing, and installation as software quality components to assess the employability of agile methods. The questionnaire was served to 216 information technology (IT) professionals (project managers: 6%, developers: 61%, quality assurance (QA) leads: 8%, and testers: 25%) from the public sector companies having experience in software development using agile and waterfall methods. The quality components and hypotheses are evaluated using the *T*-test and chi-square test, respectively, with a 95% confidence interval. The results highlight the benefits of using agile methodologies for software quality components in the public sector. Additionally, the findings demonstrate how agile software development approaches significantly affect the quality of software products and successful delivery within budget and deadline.

1. Introduction

In the modern software industry, the prime objective is the delivery of high-quality software in a shorter time. The software project's success and the satisfaction of customers' expectations are widely measured by the quality of the software [1]. A non-systematic software development approach applied to a large-scale software project will result in software products that have high costs with low quality. Therefore, an approach towards software development is a key contributor in the decision related to the quality of the software [2]. IT initiatives, about 5 to 10% of the organizations' total revenues, have made this even more important [3].

As per De Feo [4], software with high-quality has potential benefits for both the customer and the organization because the organization becomes more responsive, gains an edge over its competitors, and reduces its cost for development and marking time. Delivering low-quality software limits the company's growth. Consequently, organizations could risk their reputation, and survival will be at stake in dynamic business settings.

Traditional software methodologies are plan-driven, starting with the requirements, elicitation, and documentation, which leads to the architectural and high-level design development and inspection. The waterfall, V-model, and rational unified process (RUP) are examples of processes that follow a series of steps, including defining the requirements, building the solution, testing, and deployment. These methods cannot deal with changing customer requirements, raising concerns about quality problems [5].

Since the last decade, agile methods have become a popular trend for companies to improve their performance, focusing on software quality. Many companies' transformation from traditional software development to agile software development has dealt with complex projects with ill-defined requirements, high customer satisfaction, low defect rates, and fast development time with evolving customer needs. In these methods, software quality is ensured by customer collaboration. Now, the evolution of the end software product or service is actively shaped and guided by the customers/stakeholders; rather, they prefer to stay at the fringes of software development. However, its benefits are still unknown for successful project delivery in large public software development organizations while confirming quality. This is because of nasty attitudes, hierarchical, bureaucratic management styles, and willingness for change in these organizations [6].

This work has assessed the adoption of agile methods in public sector software development to improve the quality of the software and successful delivery. The key objective is to present a sustainable e-Governance model to ensure service delivery to the citizens. We selected nine public-sector companies with in-house software development to study the effect of quality factors, including "budget," "time," "quick debugging and correctness," "maintenance," "easy testing," and "installation." The results show that agile software development methods significantly impact the software product quality and successful delivery within budget and time. The key contributions of this study are as follows:

(i) Employability of the Agile method: we employ and access the agile method in a large-scale public sector environment to ensure quality

(ii) Result generation: a survey was performed, based on 28 questions and focusing on software quality components, by IT professionals from nine companies (involved in software development, under the umbrella of the Water and Power Development Authority (WAPDA), which is one of the largest public organizations in Pakistan)

This paper is structured into six sections. Section 2 reviews the background and related work. Section 3 proposes a conceptual framework following the results and discussions in Section 4. Section 5 presents the threats to validity. Finally, Section 6 concludes with future directions.

2. Literature Review

2.1. Background. The section is categorized into software quality, process models, and traditional and agile development methods. The key focus is to find the factors that influence software quality and develop a conceptual framework for it.

2.2. Software Quality. It is observed that a software product that is delivered within time and budget while performing its target functions correctly and efficiently still lacks quality. Consequently, the software product is hard to understand, difficult to use and maintain, easy to misuse, machine-dependent, and difficult to integrate with other software. The software quality is defined in the literature from different views, which are stated as follows:

(i) Customer view: it is characterized as an extent to which the product, process, or service fulfils the requirements [7–9]

(ii) Product view: in this context, the quality measures the unvalued features contained in every valued feature [9]

(iii) Engineering view: the quality is characterized as the extent to which a particular item affirms an architecture or requirement [9]

(iv) Value view: the level of tended fulfillment of customer expectations at an affordable cost under variation [9]

Software quality assurance is more customer-centric and can be attributed to a software product that is free of defects, delivered on time within budgetary constraints, fulfills requirements and/or desires, and can be maintained [2].

2.3. Quality Models. In this section, the popular quality models are presented. These are employed along with quality standards to ensure compliance with high-quality software requirements. Jim McCall introduced a quality model that fundamentally revolves around system and process developers. The emphasis is on efforts to bridge the difference between customers and software developers by ensuring quality aspects or features and paying considerable attention to the requirements [10]. Another model is Boehm's model, which complements McCall's quality model. In these models (Figures 1(a), 1(b), and 2), the subjective methodology is used to characterize (with focus on three levels) the primitive attributes, which complete the quality definition for high-quality software products. In addition to these, International Organization for Standardization (ISO) 9126 is the quality standard that projects quality features to be used to assess six important areas, as mentioned in Figure 3.

2.4. Traditional and Agile Software Development Approaches

2.4.1. Waterfall Model. The waterfall is one of the oldest software development models proposed by Royce in 1970 [11]. According to this model, the development phases are sequential, where a new phase starts upon completion of the current phase. The project manager expects tasks to be completed as soon as possible once they are started; however, it has been observed that project information and developers' knowledge improve with time. The inherent compliance to this model requires phases to be started even with missing or incomplete information. This model best fits the software projects where user requirements are properly documented and locked. It should be avoided for large-scale enterprise software projects where requirements regularly evolve. Moreover, this model does not incorporate a quality assurance loop between completed phases.

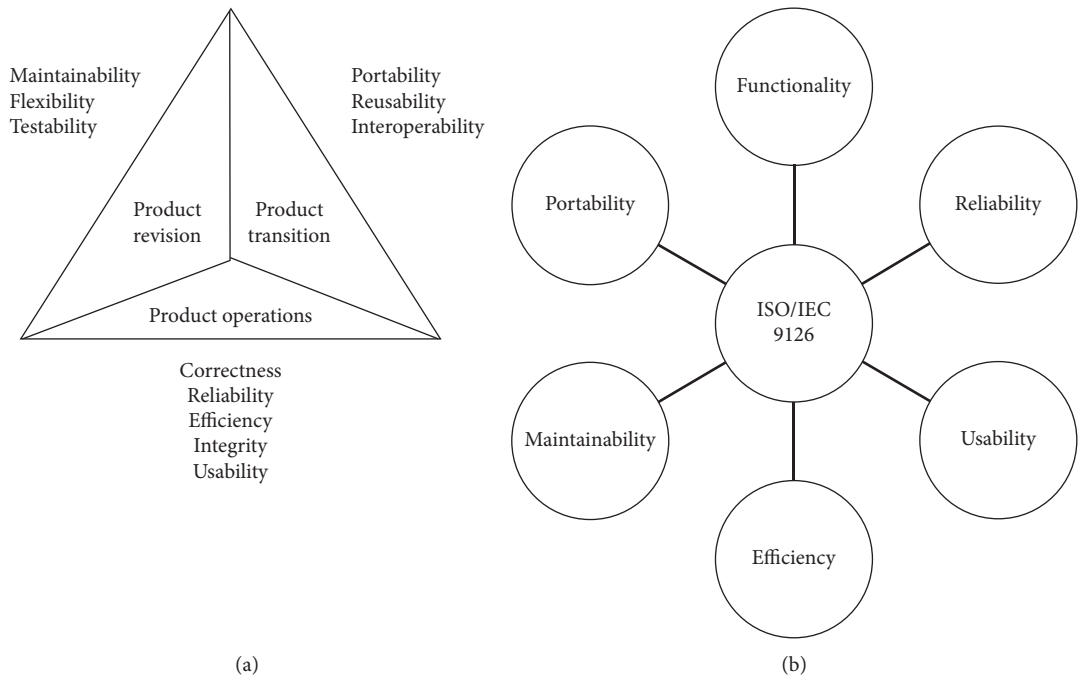

Figure 1: Software quality models: (a) McCall quality model and (b) ISO9126 quality model.

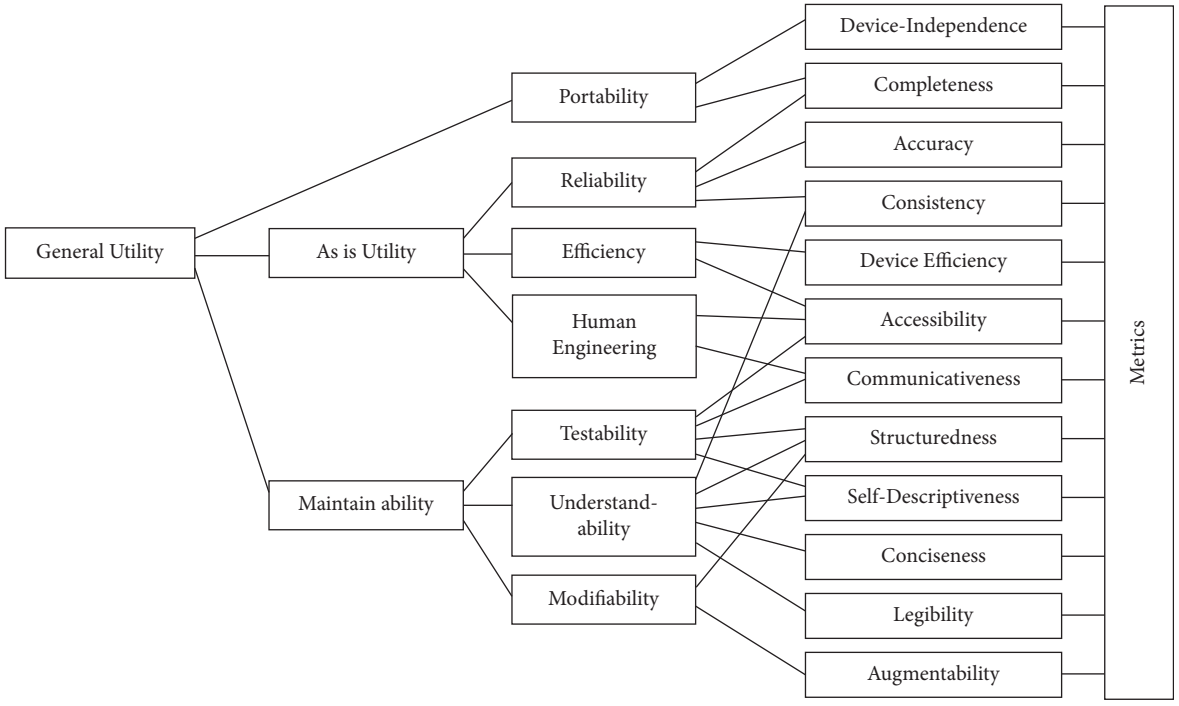

Figure 2: Software quality model: Boehm quality model.

2.4.2. Agile Methods. The "agile" concept refers to the methods and practices based on values and principles presented in the Agile Manifesto 2001 [12]. The software development approaches derived from "agile" have the ability to adapt under uncertainty. They have replaced the traditional "waterfall" approach based on being time-boxed and iterative, where software is developed in increments called sprints, compared to delivering it as a complete package.

The key elements of the "agile" manifesto are presented as follows:

(i) user engagement over complex processes

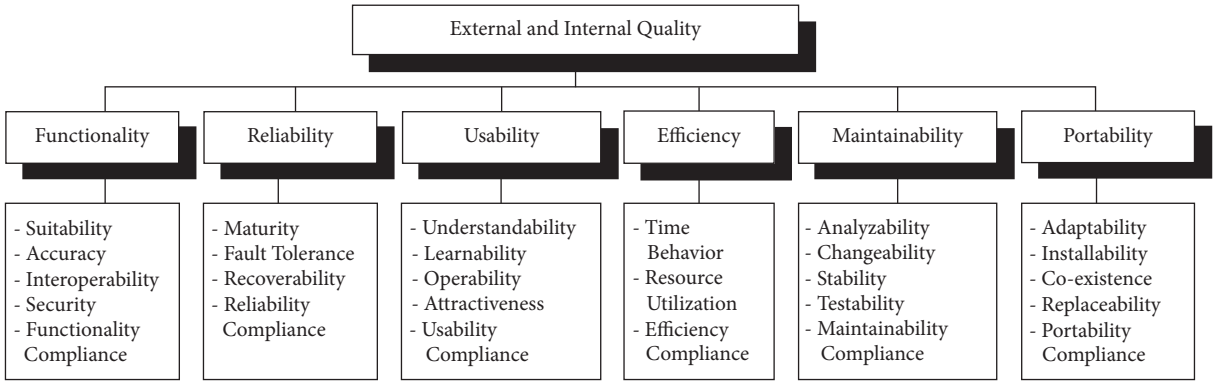

Figure 3: Quality attributes of ISO 9126.

(ii) functional software product against detailed documented product

(iii) user engagement in contractual negotiation

(iv) quick adoption of change

The "agile" approach emphasizes people and communication rather than processes and tools. The focus is on functional software that meets the expectations of the customers rather than comprehensive documentation. In the "agile" approach, tasks are broken into smaller activities to be completed incrementally with daily meetings to ensure that the project is on track. There exist many methodologies under "agile," e.g., scrum [13], extreme programming (XP) [14], feature-driven development [15], and many more. The scrum and XP methodologies are widely used. In the light of one survey, 66.7% of software houses in Pakistan use the scrum software process model as shown in Figure 4.

The software houses are adopting agile software development with feasible inclusion of changes, even in the last phases of the project [16]. This offers a competitive edge among all the software industry competitors. This highlights that the public sector in-house software development has yet to benefit from the "agile" methodologies. They currently rely on traditional software development approaches like "waterfall" to support the sustainable e-Governance model to serve the citizens under the same model.

2.5. Related Work. Nerurkar and Das [17] discussed and analysed the need for an agile project management framework for large scale projects in government and public sectors. However, they did not discuss any quality attribute. Bolhuis [18] evaluated that how large-scale agile can be effectively adopted and scaled up in Dutch public sector organizations. They concluded that their results might not be applicable for all Dutch public sector organizations. Mohagheghi and Lassenius [19] presented an organizational approach in adopting the agile in a large public organization. The study in [20] presented the challenges collected from the past studies of IT project implementation in the government sector considering the agile method. They identified 20 challenges and categorized into technology, organization, environment, and individual context. The work in [21] addressed the agile transformation in large companies with

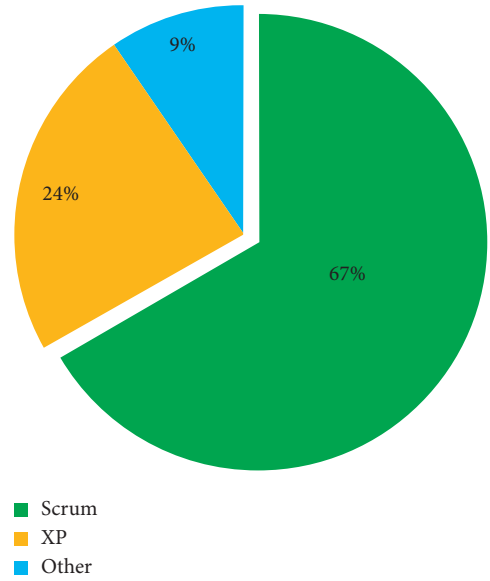

Figure 4: Software models usage in Pakistan software industry.

existing software product lines and proposed a transformation model. Fangmann et al. [22] also identified the challenges posed by the adoption of agile practices in public administration and how these challenges can be overcome. Wadood et al. [23] analysed the software quality components, e.g., budget and time for agile development in a public sector environment. Bousdekis and Kardaras [24] identified the challenges of adopting digital technologies (particularly agile) in the public sector and in local governments. In [25], authors focused on agile developments for mission critical systems in the public sector. Vacari and Prikladnicki [26] presented a systematic literature review for adopting agile methods in the public sector. They concluded that agile methods could be adopted in the public sector. However, not all the implications of adopting agile methods in the public sector are widely known.

The above literature highlights the fact that most of the studies have identified the challenges in adopting agile methods in the public sector without consideration of quality characteristic. It is imperative to identify the quality attribute implications in agile methods for large public sector organization; therefore, it motivates our research to

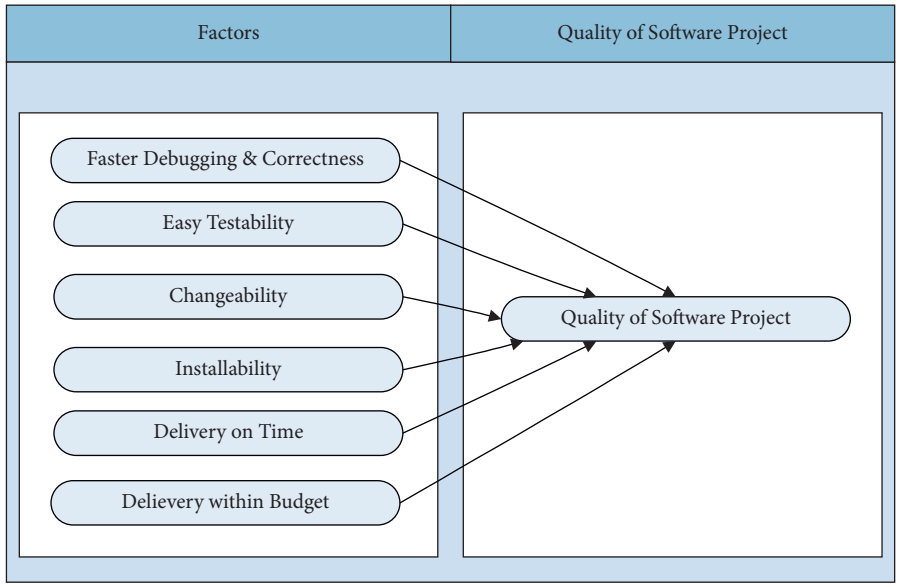

Figure 5: Conceptual framework.

Table 1: Population data.

Company	No of employees				Total
	Software project managers	Software developers	QA leads	Software testers	
Power information technology company	10	140	16	50	216

Table 2: Sample data.

Company	No of employees				Total
	Software project managers	Software developers	QA leads	Software testers	
Power information technology company	10	103	14	44	171

consider this factor for adopting agile in large public sector organization ensuring software quality.

3. Research Methodology

3.1. Proposed Conceptual Framework. The quality of a software project is assessed as defects-free, completed within budget and time, and meeting customer requirements [2]. Moreover, the software quality characteristics in public sector in-house software development are quick debugging, correctness, easy testing, and swift single click installation. These factors are evaluated on the software quality for agile approaches in public sector in-house development. A conceptual framework is proposed based on these characteristics in Figure 5.

3.2. Population and Sample Size. The population in this case study consists of software project managers, software developers, QA testers, and team leads from nine distribution company (DISCO) computer centers. The target designations are chosen because they can provide necessary data and information during a survey for this research. A survey is sent to 216 IT professionals out of 850 employees (Table 1).

It is pertinent to note that respondents have completed at least one software project using agile and waterfall software process models. The results show that 171 respondents have qualified the set initial criteria (Table 2).

3.3. Summary of Questionnaire Design. The researcher gathered data through meetings and by giving questionnaires to QA leads, developers, and testers. The questionnaires helped the researcher acquire data or information from a substantial number of individuals. The questionnaire concentrated on catching information or data in respect of quality factors such as correctness, testability, changeability, and installability (Table 3).

3.4. Analysis Method. Intending to improve the software project quality, we formulate the hypotheses in Table 4 to access the applicability of the agile software development methodology in large public sector organizations. We follow

Table 3: Design of the questionnaire.

Attribute	Quality factors	Questions
Agreement with specifications Consistency in functionality Defects	Correctness	05–09
Percentage of test Ratio of effectiveness of test Easiness or simplicity Consistency of code	Testability	10–12 and 15
Modification ability Errors subsequent to modifications Modification efforts Level of interconnection and link	Changeability	13–14 and 16–17
After installation level of stability	Installability	18–20

Table 4: List of hypotheses.

No	Type	Description
1	H0	There does not exist any difference in faster debugging and correctness ($\mu AC - \mu WC = 0$)
	H1	There does exist a difference in faster debugging and correctness ($\mu AC - \mu WC \neq 0$)
2	H0	There does not exist a difference in easy testability ($\mu AT - \mu WT = 0$)
	H1	There does exist a difference in easy testability ($\mu AT - \mu WT \neq 0$)
3	H0	There does not exist a difference in changeability ($\mu ACh - \mu WCh = 0$)
	H1	There does exist a difference in changeability ($\mu ACh - \mu WCh \neq 0$)
4	H0	There does not exist a difference in installability ($\mu AI - \mu WI \neq 0$)
	H1	There does exist a difference in easy installability ($\mu AI - \mu WI \neq 0$)
	H1	There does exist a difference in time to deliver or complete ($\mu A\ D - \mu W\ D \neq 0$)
6	H0	There does not exist a difference in the estimated budget and actual budget ($\mu AB - \mu WB = 0$)
	H1	There does exist a difference in the estimated budget and actual budget ($\mu AB - \mu WB \neq 0$)
7	H0	There does not exist a difference in software projects developed using agile and waterfall methods ($\mu AQ - \mu WQ = 0$)
	H1	There does exist a difference in software developed with agile and waterfall methods ($\mu AQ - \mu WQ \neq 0$)

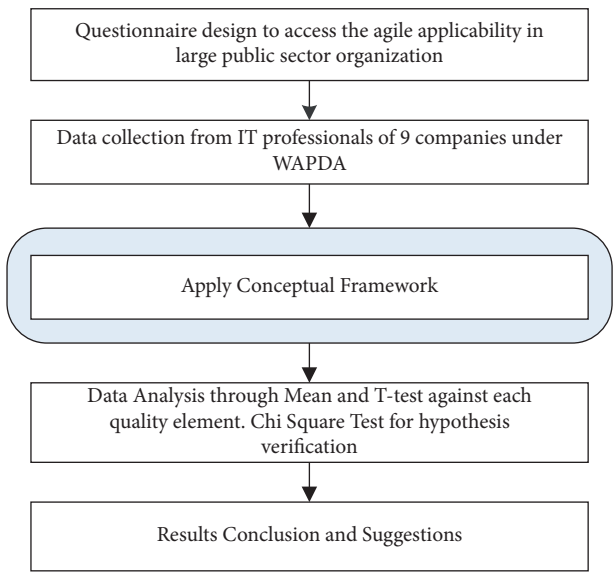

Figure 6: Research methodology summary.

Table 5: Survey response statistics.

Company	Software developers			QA leads			Software testers			Total
	S	R	A	S	R	A	S	R	A	
PITC	103	100	95	14	12	12	44	35	32	
Total S	103			14			44			161
Total R		100			12			35		147
Total A			95			12			32	139
				R% age						91.30%
				A% age (with reference to S)						86.34%
				A% age (with reference to R)						94.56%

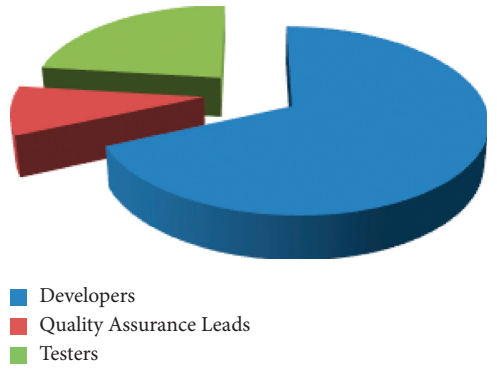

Figure 7: Respondent statistics as per valid questionnaires.

Table 6: *T*-test statistics for questions (5–9).

Statement/Question	*t*	df	Sig. 2-tailed	Mean diff	95% confidence interval of difference	
					Lower	Upper
Developers						
User_Expectation	29.631	94	0	3.611	3.37	3.85
Requirement_Capture	34.231	94	0	3.737	3.52	3.95
System_Design	32.608	94	0	3.747	3.52	3.98
System_Implementation	32.259	94	0	3.642	3.42	3.87
Faults_Free	38.662	94	0	3.789	3.59	3.98
QA Leads						
User_Expectation	13.053	11	0	4.083	3.39	4.77
Requirement_Capture	17.11	11	0	3.917	3.41	4.42
System_Design	15.654	11	0	3.583	3.08	4.09
System_Implementation	17.838	11	0	4.083	3.58	4.59
Faults_Free	14.199	11	0	4.083	3.45	4.72
Testers						
User_Expectation	20.943	31	0	3.813	3.44	4.18
Requirement_Capture	15.774	31	0	3.469	3.02	3.92
System_Design	20.586	31	0	3.781	3.41	4.16
System_Implementation	17.517	31	0	3.656	3.23	4.08
Faults_Free	19.395	31	0	3.719	3.33	4.11

the below steps to analyze data to support or reject the formulated hypotheses for software developed using agile and waterfall methods.

(i) Organized the data collected through a questionnaire using the statistical package for the social sciences (SPSS) tool, i.e., software developers, QA leads, and software testers

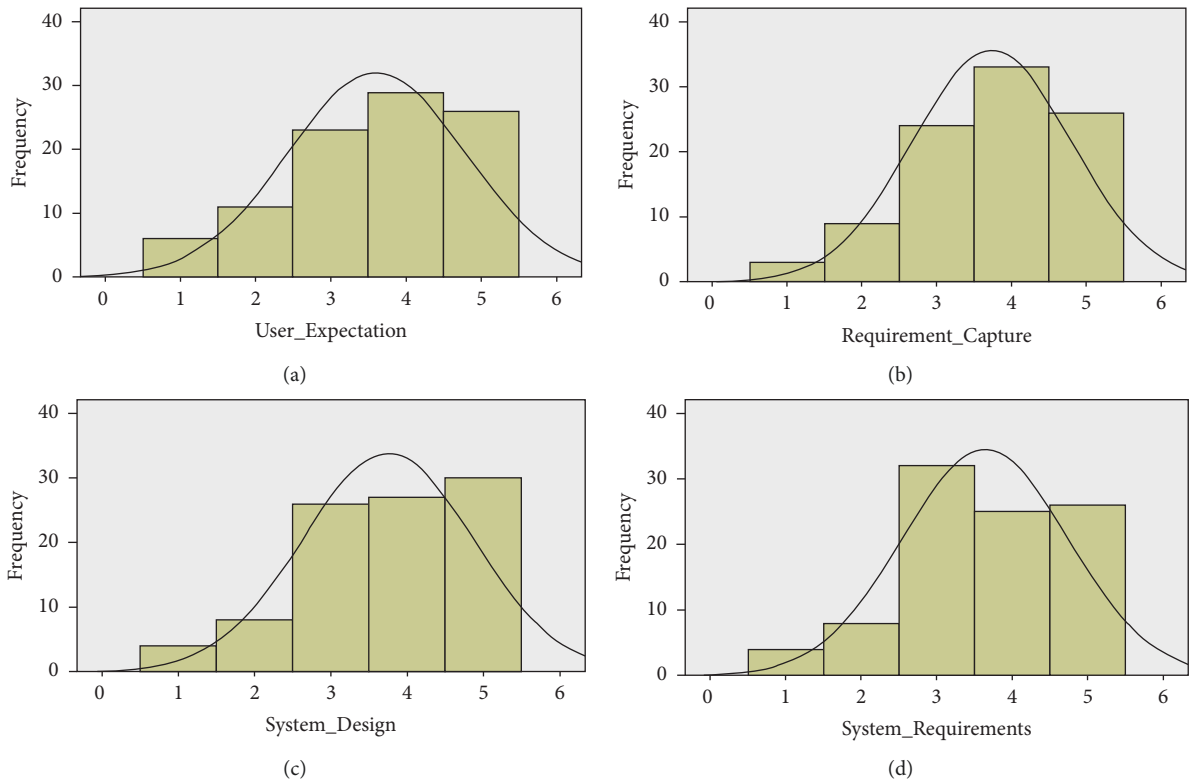

Figure 8: Histograms for the correctness quality factor from developers' point of view: (a) user expectation, (b) requirement capture, (c) system design, and (d) system requirement.

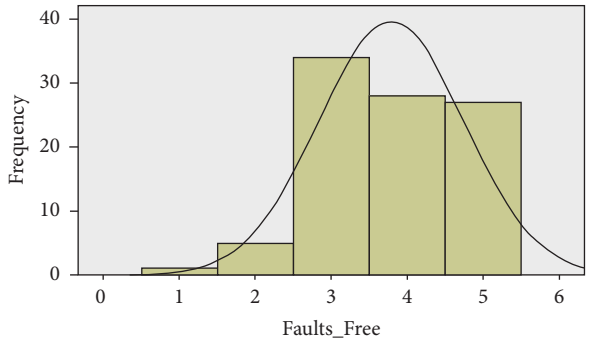

Figure 9: Histograms for the correctness quality factor from developers' "fault free" point of view.

(ii) Built the frequencies against the questions

(iii) To test the derived hypothesis on correctness, testability, changeability, and installability, we used mean and *T*-test (1-sample) against each quality element

(iv) Used the chi-square test for hypothesis verification on time and budget objectives

A full summary of the research methodology has been depicted in Figure 6.

4. Results and Discussion

The acronyms S, R, and A refer to sample, received, and accepted questionnaires. To collect data from software developers, QA leads, and software testers, 161 questionnaires were circulated. As a result, 139 responses (95%) were received, which are shown in Table 5.

The results in Table 5 show that the researcher received 147 questionnaires out of 161 questionnaires that were distributed. The response rate is about 91%. Eight respondents did not fulfill the defined criteria because eight questionnaires were ignored in the analysis. Therefore, 139 questionnaires were considered for analysis for this research, and their percentage is about 95%.

In Figure 7, we represent the diversification of 139 respondents based on their designation. It shows that 68.35% respondents are developers, 8.63% are quality assurance leads, and 23.02% are testers as per valid questionnaires.

4.1. Assessment of Quality Elements

4.1.1. Quick Debugging and Correctness. To access this quality characteristic, questions' responses (5–9) are evaluated using a 1-sample *T*-test of software developers, QA leads, and software testers, and associated with hypothesis-1. The results in Table 6 show that significant values are less than 0.05, so the null hypothesis is not accepted. It indicates that there does exist a difference in faster debugging and correction for the software developed using agile and waterfall. The visual representation is also presented in Figures 8 and 9 from the developers' point of view.

Table 7: *T*-test statistics for questions (10–12, 15).

Statement/question	*t*	df	Sig.2 tailed	Mean diff	95% confidence interval of difference	
					Lower	Upper
Developers						
Execution_Test_Script	29.675	94	0	3.463	3.23	3.69
Coding_Standards	29.08	94	0	3.505	3.27	3.74
No_Complex_Structure	33.053	94	0	3.758	3.53	3.98
Low_Interaction	35.977	94	0	3.758	3.55	3.97
QA Leads						
Execution_Test_Script	11.153	11	0	3.500	2.81	4.19
Coding_Standards	17.110	11	0	3.917	3.41	4.42
No_Complex_Structure	10.383	11	0	3.500	2.76	4.24
Low_Interaction	9.750	11	0	3.667	2.84	4.49
Testers						
Execution_Test_Script	16.566	31	0	3.563	3.12	4.00
Coding_Standards	20.256	31	0	3.750	3.37	4.13
No_Complex_Structure	17.723	31	0	3.625	3.21	4.04
Low_Interaction	16.091	31	0	3.656	3.19	4.12

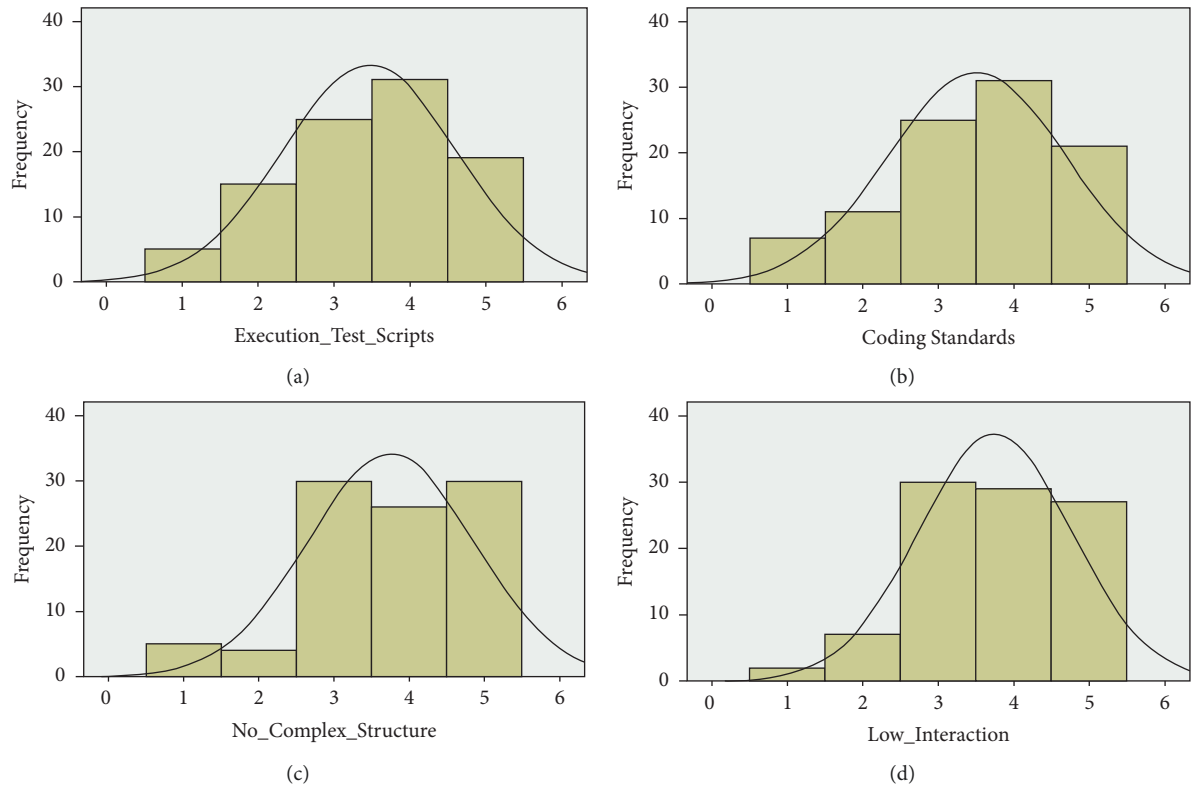

Figure 10: Histograms for the testability quality factor from developers' point of view.

4.1.2. Quick Testability. To access this quality characteristic, questions' responses (10–12,15) are also assessed on a 1-sample *T*-test of software developers, QA leads, and software testers, and associated with hypothesis-2. The results in Table 7 indicate that the null hypothesis is not accepted as significant values are less than 0.05. Therefore, it is concluded that a difference exists in easy testability quality characteristics for software developed with waterfall and agile. The visual representation of these results is also shown in Figure 10.

4.1.3. Changeability. To access this quality characteristic, questions' responses (13–14, 16–17) are evaluated using a 1-sample *T*-test of software developers, QA leads, and software testers, and associated with hypothesis-3. The results in Table 8 indicate that the null hypothesis is not accepted as significant values are less than 0.05. Therefore, it is concluded that there exists a difference in changeability for the software developed with waterfall and agile. The visual representation of these results is also shown in Figure 11.

Table 8: *T*-test statistics for questions (13-14, 16-17).

Statement/question	*t*	df	Sig.2-tailed	Mean diff	95% confidence interval of difference	
					Lower	Upper
Developers						
Easy_Modification	28.969	94	0	3.463	3.23	3.70
Easy_Minor_Changes	32.078	94	0	3.579	3.36	3.80
Low_Side_Effects	32.111	94	0	3.674	3.45	3.90
No_Functional_Issues	29.820	94	0	3.537	3.30	3.77
QA Leads						
Easy_Modification	7.097	11	0	3.083	2.13	4.04
Easy_Minor_Changes	8.864	11	0	3.333	2.51	4.16
Low_Side_Effects	10.66	11	0	3.583	2.84	4.32
No_Functional_Issues	9.466	11	0	3.583	2.75	4.42
Testers						
Easy_Modification	18.644	31	0	3.625	3.23	4.02
Easy_Minor_Changes	20.879	31	0	3.750	3.38	4.12
Low_Side_Effects	19.774	31	0	3.813	3.42	4.21
No_Functional_Issues	16.74	31	0	3.594	3.16	4.03

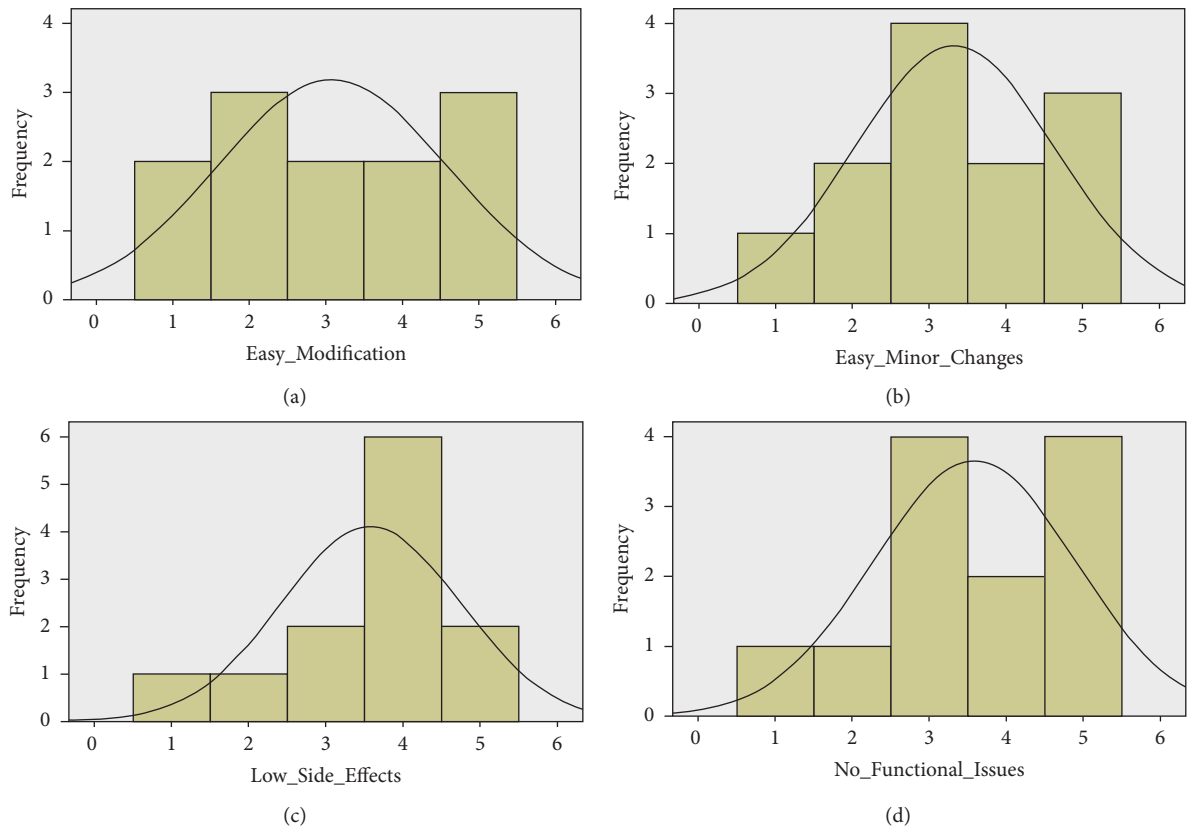

Figure 11: Histograms for the changeability quality factor from QA lead's point of view.

4.1.4. Quick Installability. To access this quality characteristic, questions' responses (18–20) are subjected to a 1-sample *T*-test of software developers, QA leads, and software testers, and associated with hypothesis-4. The results in Table 9 indicate that the null hypothesis cannot be accepted as significant values are less than 0.05. Therefore, it is concluded that there exists a difference in easy installability for software developed through waterfall and agile. The visual representation of these results is also shown in Figure 12.

4.1.5. Within Time Delivery. To access this quality characteristic, interviews are conducted and subjected to the chi-square test to verify hypothesis-5. The data are collected from 10 software project managers considering the projects

Table 9: *T*-test statistics for questions (18–20).

Statement/question	*t*	df	Sig. 2-tailed	Mean diff	95% confidence interval of difference	
					Lower	Upper
Developers						
Installation_Challenges	28.772	94	0	3.495	3.25	3.74
Installation_Modify	31.998	94	0	3.621	3.40	3.85
Compatibility	30.203	94	0	3.621	3.38	3.86
QA Leads						
Installation_Challenges	8.864	11	0	3.333	2.51	4.16
Installation_Modify	10.000	11	0	3.333	2.60	4.07
Compatibility	10.319	11	0	3.667	2.88	4.45
Testers						
Installation_Challenges	16.476	31	0	3.469	3.04	3.90
Installation_Modify	24.076	31	0	3.813	3.49	4.14
Compatibility	17.737	31	0	3.688	3.26	4.11

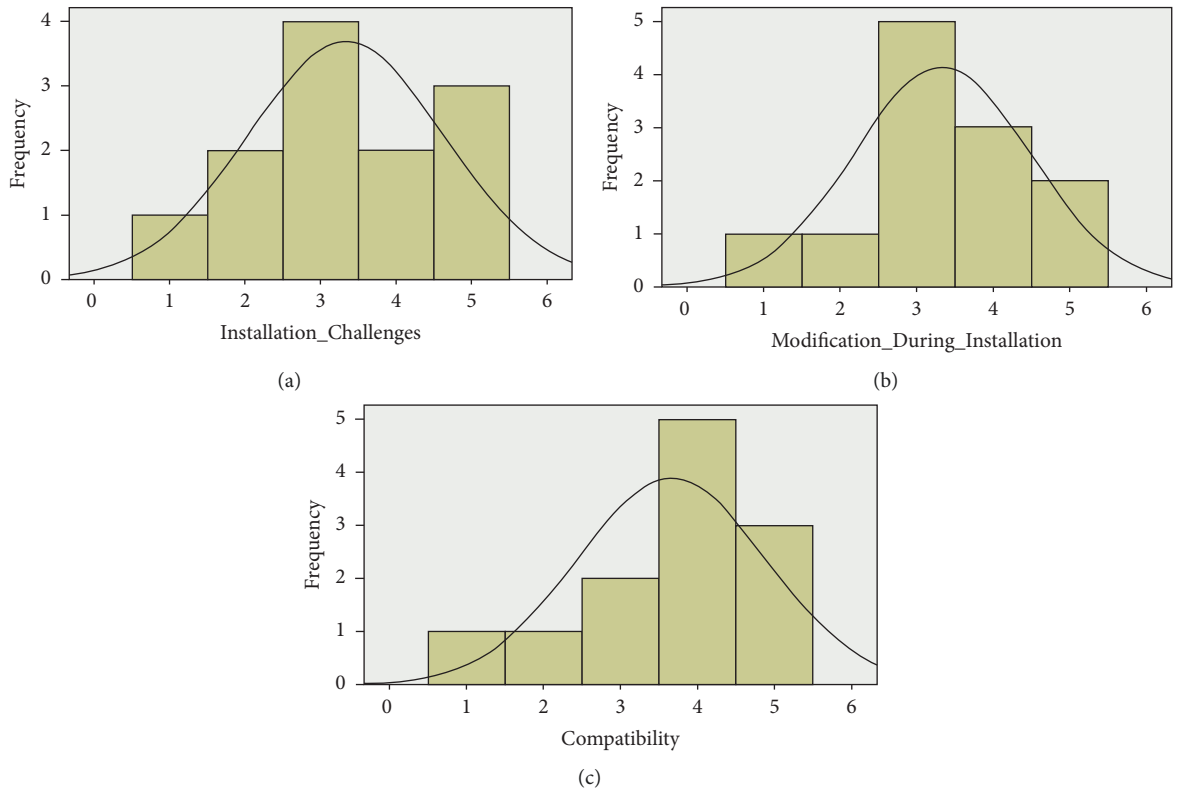

Figure 12: Histograms for the installability quality factor from QA lead's point of view.

completed in the last five years (Table 10). The project percentages completed on time through agile and waterfall are 69.23% and 52%, respectively.

The results in Tables 11 and 12 indicate that the chi-square test result is 2.7, which is lower than the critical value of 3.84. Therefore, it is concluded that the null hypothesis is rejected at a 5% significance level. It also highlights no evidence that there is a difference in the timely completion of software projects between waterfall and agile methods.

4.1.6. Within Budget Delivery. To access this quality characteristic, interviews are conducted and subjected to the chi-square test to verify hypothesis-6. The data are collected from 10 software project managers considering the projects completed in the last five years. The project percentages completed within budget with agile and waterfall are 58.97% and 48.00%, respectively.

The results in Tables 13 and 14 indicate that the chi-square value is 1.05, which is less than critical value as 3.84. Hence, the null hypothesis cannot be rejected at a 5%

Table 10: Data received against project managers' interviews for quality factor delivery on time.

Respondent	Agile projects	Waterfall projects	No. of agile projects Completed on time	No. of waterfall projects Completed on time
01.	03	02	02	02
02.	07	01	06	01
03.	08	02	06	01
04.	03	03	00	02
05.	02	04	01	02
06.	02	10	01	06
07.	03	08	02	03
08.	02	02	02	02
09.	05	07	04	03
10.	04	11	03	04
Total	39	50	27	26
	%Age		69.23%	52.00%

Table 11: Project manager interviews' observed and expected values for on-time completion of software projects.

Approach		Agile	Waterfall	Total
Successful	Observed	27	26	53
	Expected	23.22	29.78	53
Unsuccessful	Observed	12	24	36
	Expected	15.78	20.22	36
Total		39	50	89

Table 12: Chi-square test results for on-time completion of software projects.

O	E	$(O-E)$	$(O-E)^2$	$(O-E)^2/E$
27	23.22	3.78	14.2884	0.615349
26	29.78	-3.78	14.2884	0.479799
12	15.78	−3.78	14.2884	0.905475
24	20.22	3.78	14.2884	0.706647
		Statistics from the test		2.70727
		Critical value of chi-square		3.841

Table 13: Statistics of observed/expected values for within budget completion quality characteristics derived from project manager's responses.

Method		Agile	Waterfall	Total
Successful	Observed	23	24	47
	Expected	20.60	26.40	47
Unsuccessful	Observed	16	26	42
	Expected	18.40	23.60	42
Total		39	50	89

Table 14: Chi-square test results for within budget completion of software projects.

O	E	$(O-E)$	$(O-E)^2$	$(O-E)^2/E$
23	20.60	2.40	5.7600	0.279612
24	26.40	−2.40	5.7600	0.218182
16	18.40	−2.40	5.7600	0.313043
26	23.60	2.40	5.7600	0.244068
		Statistics from the test		1.054905
		Critical value of chi-square		3.841

Table 15: *T*-test statistics–agile vs traditional software development methods' questions.

Statement/question	*t*	df	Sig. 2-tailed	Mean diff	95% confidence interval of diff	
					Lower	Upper
Developers						
Flexibility	26.0	94	0	3.3	3.0	3.6
Robustness	28.0	94	0	3.4	3.2	3.7
Cost effectiveness	28.1	94	0	3.4	3.2	3.7
Reusability	26.2	94	0	3.3	3.1	3.6
Reduced risks in success	22.7	94	0	3.1	2.8	3.4
End user involvement	24.6	94	0	3.2	2.9	3.5
Delivery on time	24.0	94	0	3.1	2.8	3.3
Interaction between team members	25.2	94	0	3.0	2.8	3.3
QA leads						
Flexibility	8.6	11	0	3.1	2.3	3.9
Robustness	10.8	11	0	3.3	2.7	4.0
Cost effectiveness	9.0	11	0	3.4	2.6	4.3
Reusability	10.2	11	0	3.4	2.7	4.2
Reduced risks in success	10.0	11	0	3.6	2.8	4.4
End user involvement	9.8	11	0	3.2	2.5	3.9
Delivery on time	8.1	11	0	3.0	2.2	3.8
Interaction between team members	10.2	11	0	3.4	2.7	4.2
Testers						
Flexibility	17.5	30	0	3.8	3.3	4.2
Robustness	19.0	30	0	3.5	3.1	3.9
Cost effectiveness	16.8	30	0	3.5	3.1	4.0
Reusability	13.9	30	0	3.3	2.8	3.8
Reduced risks in success	14.3	30	0	3.4	2.9	3.8
End user involvement	16.5	30	0	3.2	2.8	3.6
Delivery on time	12.9	30	0	3.2	2.7	3.7
Interaction between team members	12.7	30	0	3.0	2.5	3.5

significance level. It is concluded that there is no evidence to say a difference in the completion of software projects within budget between waterfall and agile methods.

4.1.7. Traditional vs. Agile Methods. A questionnaire was designed with eight questions to compare the agile and traditional software development methodologies. For hypothesis-7, the results in Table 15 indicate that significant values are <0.05, so the null hypothesis is rejected at a 5% significance level. Therefore, it is concluded that there is a difference in software projects developed with agile and waterfall.

5. Threats to Validity

There are some limitations to this work. First, this case study focuses on the quality characteristics related to the software developers. Second, data are gathered to validate whether there is a contrast in completing the project within time and budget using software development process techniques, i.e., agile and waterfall. Third, we have not separated results based on the size of the project because of the data accumulation problem within the specified period. Last, the study was carried out in Pakistan and gathered data from public sector software development organizations. Hence, the results speak about the experiences of IT experts and professionals, particularly in public sector organizations in Pakistan; however, other software development organizations in Pakistan and around the globe were not considered.

6. Conclusion and Future Work

The software industry faces major challenges, e.g., high software quality, ever-increasing software complexity, low budgets, and tight schedules. To address these issues, agile software development approaches provide an effective solution. Adopting agile methods across the private sector has proven their usefulness and effectiveness. However, it remains questionable for the public sector. Our study intends to contribute to the already existing body of knowledge about the usefulness of agile methodologies for the public sector environment to improve the software quality. Our questionnaire-based survey results demonstrate a clear trend toward the effectiveness of agile-based development compared to traditional software methodologies. For quality factors, i.e., "quick debugging and correctness," "maintenance," "easy testing," and "installation," agile methods have distinct superiority over traditional software development for software development companies in a public sector environment. The results suggest that agile development techniques or methods must be used for faster delivery, easy debugging, easy testability, and easy installability, in a public software development organization. In the future, other

project quality components will be considered, such as time, cost, and the most suitable strategy between agile and waterfall.

Data Availability

Data are available on request.

Conflicts of Interest

The authors declare that there are no conflicts of interest.

Authors' Contributions

The authors contributed equally to the design of ideas, analysis of results, and writing of the article.

References

[1] D. Galin, *Software Quality: Concepts and Practice*, John Wiley & Sons, Hoboken, New Jersey, U.S, 2018.

[2] C. Y. Laporte and A. April, *Software Quality Assurance*, John Wiley & Sons, Hoboken, New Jersey, U.S, 2018.

[3] R. N. Charette, "Why software fails [software failure]," *IEEE spectrum*, vol. 42, no. 9, pp. 42–49, 2005.

[4] J. A. De Feo, *Juran's Quality Handbook: The Complete Guide to Performance Excellence*, McGraw-Hill Education, London, 2017.

[5] S. Ambler, "Quality in an agile world," *Software Quality Professional*, vol. 7, no. 4, p. 34, 2005.

[6] L. J. Mullins and J. E. McLean, *Organisational Behaviour in the Workplace*, Pearson Harlow, UK, 2019.

[7] C. Alves, X. Franch, J. P. Carvallo, and A. Finkelstein, "Using goals and quality models to support the matching analysis during cots selection," in *Proceedings of the International Conference on COTS-Based Software Systems*, pp. 146–156, Springer, Berlin, Heidelberg, 2005.

[8] J. Singh and N. B. Kassie, "User's perspective of software quality," in *2018 Second International Conference on Electronics, Communication and Aerospace Technology (ICECA)*, Piscataway, NJ, USA, March 2018.

[9] N. Z. H. Zakaria, A. R. Hamdan, J. Yahaya, and A. Deraman, "User centric software quality model for sustainability: a review," *Lecture Notes on Software Engineering*, vol. 4, no. 3, p. 199, 2016.

[10] M. W. Suman and M. Rohtak, "A comparative study of software quality models," *International Journal of Computer Science and Information Technologies*, vol. 5, no. 4, pp. 5634–5638, 2014.

[11] M. Kramer, "Best practices in systems development lifecycle: an analyses based on the waterfall model," *Review of Business & Finance Studies*, vol. 9, no. 1, pp. 77–84, 2018.

[12] L. Apke, *Understanding the Agile Manifesto*, Lulu. com, Morrisville, North Carolina, 2015.

[13] W. Zayat and O. Senvar, "Framework study for agile software development via scrum and kanban," *International Journal of Innovation and Technology Management*, vol. 17, no. 04, Article ID 2030002, 2020.

[14] F. Anwer, S. Aftab, S. S. M. Shah, and U. Waheed, "Comparative analysis of two popular agile process models: extreme programming and scrum," *International Journal of Computer Science and Telecommunications*, vol. 8, no. 2, pp. 1–7, 2017.

[15] A. F. Chowdhury and M. N. Huda, "Comparison between adaptive software development and feature driven development," in *Proceedings of the 2011 International Conference on Computer Science and Network Technology*, pp. 363–367, IEEE, Harbin, China, December 2011.

[16] S. Al-Saqqa, S. Sawalha, and H. AbdelNabi, "Agile software development: methodologies and trends," *International Journal of Interactive Mobile Technologies*, vol. 14, no. 11, p. 246, 2020.

[17] A. Nerurkar and I. Das, "Agile project management in large scale digital transformation projects in government and public sector: a case study of dilrmp project," in *Proceedings of the 10th International Conference on Theory and Practice of Electronic Governance*, pp. 580-581, New Delhi AA, India, March 2017.

[18] W. Bolhuis, *How Can (Large Scale) Agile Be Effectively Adopted and Scaled up in Dutch Public Sector Organisations*, Master's thesis, 2021.

[19] P. Mohagheghi and C. Lassenius, "Organizational implications of agile adoption: a case study from the public sector," in *Proceedings of the 29th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*, pp. 1444–1454, Athens, Greece, August 2021.

[20] H. Dwi Harfianto, T. Raharjo, B. Hardian, and A. Wahbi, "Agile transformation challenges and solutions in bureaucratic government: a systematic literature review," in *Proceedings of the 2022 5th International Conference on Computers in Management and Business (ICCMB)*, pp. 12–19, Singapore, January 2022.

[21] J. Klünder, P. Hohl, and K. Schneider, "Becoming agile while preserving software product lines: an agile transformation model for large companies," in *Proceedings of the 2018 International Conference on Software and System Process*, pp. 1–10, Gothenburg, Sweden, May 2018.

[22] J. Fangmann, H. Looks, J. Thomaschewski, and E.-M. Schön, "Agile transformation in e-government projects," in *Proceedings of the 2020 15th Iberian Conference on Information Systems and Technologies (CISTI)*, pp. 1–4, IEEE, Seville, Spain, June 2020.

[23] K. Wadood, M. K. Shahzad, and M. Iqbal, "Employability assessment of agile methods for software quality: an empirical case study," in *Proceedings of the European Conference on Software Process Improvement*, pp. 598–614, Springer, Cham, 2020.

[24] A. Bousdekis and D. Kardaras, "Digital transformation of local government: a case study from Greece," in *Proceedings of the 2020 IEEE 22nd Conference on Business Informatics (CBI)*, pp. 131–140, Antwerp, Belgium, June 2020.

[25] D. Russo, G. Taccogna, P. Ciancarini, A. Messina, and G. Succi, "Contracting agile developments for mission critical systems in the public sector," in *Proceedings of the 40th International Conference on Software Engineering: Software Engineering in Society*, pp. 47–56, Gothenburg, Sweden, May 2018.

[26] I. Vacari and R. Prikladnicki, "Adopting agile methods in the public sector: a systematic literature review Embrapa Informática Agropecuária-Artigo em anais de congresso (ALICE)," in *Proceedings of the International conference on software engineering and knowledge*, Pittsburgh, PA, USA, 2015.