How SE Models Can Be Used to Determine the Quality of Software Being Developed

Jacky F Wong
Clemson University
Clemson, South Carolina, USA
jackyw@clemson.edu

Mary Damilola Aiyetigbo
Clemson University
Clemson, South Carolina, USA
maiyeti@clemson.edu

Venkata Revanth Naidu Danala
Clemson University
Clemson, South Carolina, USA
vdanala@clemson.edu

Hruday Charan Reddy Santimalla
Clemson University
Clemson, South Carolina, USA
hsantim@clemson.edu

ABSTRACT

The software engineering process is highly complicated and involves many steps, individuals, and tools to produce good software. A constantly evolving field demands an equally evolving process; organizations have created various software engineering models to standardize general guidelines to streamline this process best. In order to better understand the differences between each of the most popular models and how they contribute to good quality software, we asked, via questionnaire, software engineering professionals to rank models they have experience within various software quality attributes. Analyzing over forty participants' opinions provides insight into how software engineering models differ. Each model has its advantages and disadvantages for creating quality software, but our study suggests the Agile and DevOps models as creating the highest quality code.

1 INTRODUCTION

In order to better facilitate the predictable production of well documented, maintainable, and otherwise effective software, software engineering emerged with sets of models and methods that describe specific development life cycles [21]. Many types of software engineering models have been available for decades, and yet still more progress is being made in software engineering models. Traditional software development approaches can be ineffective in dealing with rapidly changing needs, especially in modern technology services such as social media [25]. Predictive rather than adaptive strategies are used in conventional software development life cycle (SDLC) methods [21].

One of the many tasks during the development of a project in the IT industry is the implementation of software engineering models. Companies have at least one vertical in their business section dedicated to this, which requires entire emphasis and the development of creative processes. A model is an abstract depiction of the entire software engineering process, created to share and utilize sets of techniques in order to develop better software [25]. Effective utilization of the correct software engineering model for a project can significantly increase the efficacy of the development team [25].

There are varieties of software development models. Bakota et al. defined six high-level product quality characteristics that industry experts and academic researchers alike have universally accepted: functionality, dependability, usability, efficiency, maintainability, and portability. Low-level quality features, which might be internal or external, have an impact on the characteristics [3].

There are many models that modern companies use in their software engineering. For example, this includes but is not limited to five primary techniques: Waterfall, Iteration, V-shaped, Spiral, and Extreme [25]. Of particular interest is the increasingly popular Agile framework, which would fall under the umbrella of the iteration technique previously mentioned, compared to the more traditional waterfall model.

Agile has become ever more popular due to the prevalence of mobile applications (apps) used on a smartphone. Due to the high demand by users for new features, the rapid pace at which Agile allows a team of developers to work at an appropriate pace [17]. This model has pros and cons a software engineering team must consider, along with the specific use case, in deciding whether to pursue its usage. If a team were to adopt one or more of the agile methodologies, then said the team could be rewarded with improved cost, time, quality, and productivity [20]. Additionally, the waterfall model remains very widely used by many companies in various forms for various reasons [25]. Although there are quite a few disadvantages with this model(like, need to start code from scratch if there was a requirement in the middle of this cycle), This model has not lost its touch. The less common but still relevant techniques are studied as well.

An area of concern for adopting any given software engineering model is the severity of risk at various phases of software development [5]. Many software engineering models have different risks during their stages of development, which can contribute towards differing quality of software [23]. For example, depending on the specific model, a change during development could increase the number of errors in the final code. [7, 10]. There is a method to avoid errors from creating large disasters during critical stages of development called proper requirements engineering, which can help properly establish the exact "needs and wishes" of the end-users of the software [9]. Multiple methods exist for this process and can be adapted to the different types of models that exist for software engineering. For example, a software development team can take an agile approach for software engineering and apply it to requirements engineering with varying success, and drawbacks [6].

An additional area for consideration would include open-source software development. This model contrasts heavily with the relatively traditional closed models previously mentioned in nearly every aspect. The quality of software developed by open source initiatives can differ wildly due to its nature but should have some shared characteristics compared to closed source [32]. Open source allows a community of users to collaborate in building software, be inside of the larger company (e.g., Microsoft - Azure, .NET, Visual Studio Code, Windows terminal, PowerShell, and so on. Google - Android, Angular, Chromium, Firebase, Flutter, and so on.) or smaller, volunteer-driven groups (e.g., the Spacemacs text editor).

There is a lack of a clear answer as to which of the many software engineering models produce the best end result of their processes: good software. There have been studies about how individual models affect software quality, but not necessarily comparing them in a qualitative method against each other. To this end, we have conducted a study comparing the various software engineering models and their effect on software quality from the software engineers' point of view.

2 PROBLEM STATEMENT

Many studies [15, 18, 27] have looked into what techniques in each software model have impacted the quality of software. However, we want to evaluate which of these models have more significance in affecting software quality based on the responses we get from the survey, which will be conducted with industry experts in software engineering. Most organizations adopt agile and DevOps because it is fast and easy. We want to know how these models impact the overall quality of the product. In this study, we will use a set of adapted quality attributes as defined by [3], [8] (Fig. 1), [13], and we will focus on three software models: Waterfall,
Agile and DevOps.

In this study, we want to answer the following research questions:

- **RQ1:** What is the most-used software model in organizations?
- **RQ2:** Is there an effect that the software engineering models waterfall, agile, and DevOps have on the quality of the software they produce? If so, what is their effect on quality?
- **RQ3:** What is the difference between waterfall, agile, and DevOps in terms of the quality of the software they produce?
- **RQ4:** What is the top choice of software model among developers?

3 BACKGROUND

Most organizations depend heavily on software for the companies' day-to-day activities and to help perform complex tasks in a faster and efficient way. These software applications are developed using different software development methods depending on the type and functionalities of the software being developed; hence the choice of software development model plays a significant role in the overall quality of software application and the development process. However, many organizations are faced with the challenge of choosing a suitable software development model.

Software development life cycle (SDLC) applies standard business practices to build software applications. The development of every software application is broken down into various stages, and SDLC is used to outline the tasks to be performed at each stage in the software development. The life cycle is used to define the methodology for improving the quality of the software and the overall development process to produce applications that are cost-efficient and meet users' needs [19].

3.1 Software Development Phases

SDLC comprises several stages from the planning phase, requirement gathering phase, design phase, implementation phase, testing phase, and maintenance phase. The different software development models are based on these phases of software development. Different software development models have advantages and disadvantages, and choosing the suitable model is very crucial [26]. Therefore, it is necessary to measure how the choice of a software development model affects the overall quality of the application being developed.

The SDLC provides a series of tasks for those involved in the application development process to follow at each stage of development.

Planning Phase is where a feasibility study of the intended software is being carried out, and project stakeholders evaluate the terms of the project. The outcome of the planning phase is a clearly defined scope and purpose of the application.

Requirement Gathering Phase involves defining the functionalities of the application, and it also consists in determining the resources needed to build the project.

Design and Prototyping Phase is where features of the application are defined in detail. It explains how users will interact with the software and how the system will respond. It also explains in details platforms where the software will run, how the system will interact with other assets as well the security features of the application.

Implementation Phase is the actual coding of the application. This phase is implemented mainly by the software developers, and at this phase, developers follow the agreed blueprint created during the phase to create a working software application.

Testing Phase involves validation and verification process carried out on the software to ensure that applications developed are built according to specified requirements. Application is deployed to production at the *Deployment Phase* for access by end-users.

Maintenance Phase is performed to keep the system capable of operating correctly without interruption. This is the process of modifying a software solution after delivery to fix defects and improve performance. It also includes adapting software to its environment and accommodating new user requirements [33].

3.2 Software Development Models

Some of the most popular software development life cycles are Waterfall, V-Shaped, Incremental, Spiral, and Agile Model [2]. The choice of these models depends on the nature of the software, skills, and strength of team members, product time to market, and management's criteria.

Waterfall Model: is commonly referred to as the traditional method. It is a relatively simple model to use, and its processes are in a sequential downward direction through the list of phases that must be executed. In the waterfall model, each phase must

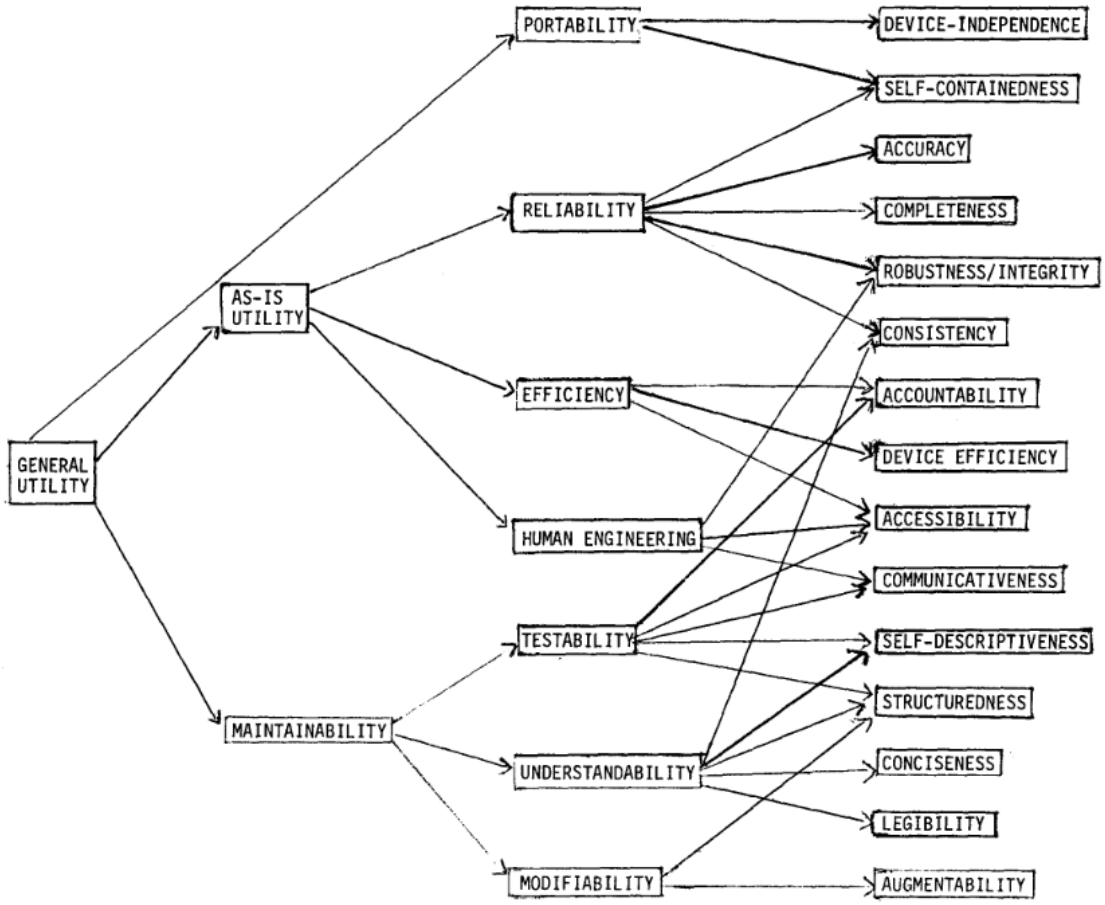

Figure 1: Software Quality Characteristics Tree [8]

be completed entirely before the start of the next phase [31]. The waterfall model requires that all the application's requirements are predetermined and documented. This model is usually not flexible as any change in the requirements during the development process may result in the project starting from scratch. However, it is easy to estimate costs and allocate resources. The waterfall model comprises five phases: Analysis, design, implementation, testing, and maintenance [4].

Incremental Model: In this model, projects are divided into small iterations, and each iteration is a mini-waterfall process. The output of one iteration is used as input for the next build. This process allows further improvement even before the whole application is completely developed. All the processes are done iteratively until the product is entirely developed.

Spiral Model: This model is a combination of incremental and waterfall models. In this phase, the development process is divided into parts, and the risky parts are developed early [30]. Spiral Model has four phases: Analysis, Evaluation, Development, and Planning phase. Software product goes through each phase iteratively, and the outcome of each iteration is used for evaluation to ensure that necessary adjustments are made at the early stage.

Agile Model: The primary motivation for this model is that it puts users' needs first and ensures working products are released to customers on time to respond to a changing market while improving on the features of the product on demand. Agile methodology focuses strongly on users experience, and it relies strongly on excellent communication among team members [19, 22].

DevOps: This model allows for collaboration between developers and the operations team to deliver software and services rapidly, reliably, and with high quality. DevOps encourages sharing tasks and responsibilities among team members from development to deployment and support. DevOps extends the goal of agile from continuous development to continuous integration and release [27].

3.3 Software Quality

Quality is the most crucial factor of software development as the success of a software project is related to meeting users' satisfaction. Quality describes the customer satisfaction as well as development organization [1]. The main aim of the Software development life cycle is to ensure delivery of quality software that meets or exceeds users' expectations, is cost-effective, has early time to market, and secure. Therefore, it is the role of organizations to adopt a software development model that ensures high-quality software products.

There are some misconceptions by many project teams about the word quality. Many believe that quality is the duty of a Quality Assurance. However, this belief is untrue because quality is everybody's business, and quality cannot be ensured by only a fraction of the product team [29]. Quality must be considered at every stage of the development, and this requires the involvement of almost all the groups involved in the development process.

Several attributes define the quality of a software product, and these attributes can be divided into two quality criteria: External and Internal. External qualities are related to the user's satisfaction with the product when using it, while Internal qualities are code-related, meaningful to developers [15]. Software quality attributes are defined by fixed quality models such as ISO/IEC 25010:2011 standard. This standard defined the product quality model to compose of eight characteristics, and these are Functional suitability, Performance Efficiency, Compatibility, Usability, Reliability, Security, Maintainability, and Portability [12, 13, 16].

Functional suitability is a view of the product's effectiveness, and it defines the completeness, correctness, and appropriateness of the software product.

Performance efficiency explains the time behavior, resource use, and capacity of the product. Performance of delivered product may be difficult to determine during development as production environment differs.

Compatibility describes the coexistence and interoperability of a product between other applications.

Usability is measured by relevant product sub characteristics such as appropriateness, recognition, learnability, operability, user-error protection, user interface aesthetics, and accessibility.

Reliability explains how mature, available, recoverable, and fault-tolerant a system can be when deployed.

Security attribute describes users' confidence in how secure a software product is in terms of confidentiality, integrity, accountability, and authenticity.

Maintainability measures the degree of effectiveness and efficiency with which a product can be modified, and it comprises of testability, reusability, and modifiability of a software product.

Portability is the degree of effectiveness and efficiency with which a system or product can be transferred from one platform or usage environment to another. It describes the adaptability, installability, and replaceability of a software system.

Hossain [15] focused on how Agile methodology enhances software quality by evaluating how each technique in the agile model improves quality factors of a software product. This research grouped quality factors into three: Quality of Design, Quality of Performance, and Quality of Adaptation, and fourteen quality characteristics in total under these groups were observed. The result showed that the continuous integration technique of the agile model contributed to about 50% of the quality attributes considered in the evaluation. Perera [27] conducted a study on the impact of DevOps practices on software quality, and the research highlighted that quality of the software could be improved by using CAMS (Culture, Automation, Measurement, Sharing) framework in DevOps. Using Pearson Correlation, this study tried to find the relationship between each of the DevOps frameworks and software quality indicators standards defined by [11]. The result obtained showed a strong relationship between each technique of the DevOps framework and the quality of software. The outcome of regression analysis also showed that quality would increase if automation, measurement, culture, and sharing were increased simultaneously in this order.

Another study by [18] mapped the agile software development process with various quality attributes to determine the impact of the agile model on the quality of software. [24] identified possible practices that can be done to increase the quality of agile methodology. [34] also concluded that the use of the agile model in software model increases some software quality factors such as correctness, reliability, portability, testability, efficiency, and extensibility. In general, these studies focused on only one software model in their research, and they evaluated the impact of software quality based on some model techniques. This work would like to consider which software model impacts software quality the most, mainly Waterfall, Agile, and DevOps.

4 STUDY DESIGN

4.1 Participants

Participants consisted of volunteers recruited online through the social media platform LinkedIn. This platform was chosen based on the ability to quickly reach our target population via each researcher's network of connections. Each researcher created a public post on their own LinkedIn profile, asking volunteers to take this study's survey. If the reader wanted additional details, they clicked the provided hyperlink, which brought them to the survey. The post also asked participants to share the post themselves to gain more exposure.

This study asked for a population of users who have had previous experience or were currently working in software engineering or software development. Age was restricted to adults eighteen (18) years or older. There were no other inclusion or exclusion criteria. No incentives were provided to participants.

4.2 Data Collection

Using Google Forms, we created survey questions. The Google Forms platform was used to create the survey questionnaires, asking questions based on software quality that consisted of 3 main sections: Organization and Team, Experience, and Quality Attributes. The participant's details, such as location, current working industry, and organization details, were included in the first section of the survey. The majority of the questions were based on a rating system that ranged from a scale of 1 to 7. Apart from software quality questions, we asked participants about their experience in the software industry, expertise in software methodologies, and implementations in the software quality factors.

We gathered information from 43 industry experts. The questions took about 15 minutes to complete on average. This information was compiled automatically by the Google Forms software into a spreadsheet.

4.3 Methods

The data collected from the survey was analyzed via descriptive statistics; we calculated the mean rating of each mentioned attribute for the testing models. Using these mean values, represented in histograms from respective tables, we found that each model has its strength and weakness based on the attributes calculated. We

also assigned a value to the model calculated from the mean of all its attributes.

4.4 Threats to Validity

Due to time constraints and the relatively low number of responses from our target respondents, we could only analyze data collected from 43 survey responses. Considering that the study tries to analyze the quality impact in more than one software model, a higher number of responses will give a generalized view of models used in different organizations to make accurate conclusions about our findings. Another limitation to this study is that most of our respondents are from the United States. Considering the widespread IT application across different regions and countries, conclusions drawn from data collected in the United States might not apply to other countries. We want to collect responses from software developers in different countries in future work. This additional data will also give us insights into what software model is mainly used in other countries and how these models impact the quality of applications deployed in each country.

5 RESULTS

5.1 Demographics of Participants

The first section of the survey deals with the demographics of the respondents with questions including location, industry, and organization size. Forty-three participants responded to our survey. Of this, most of the participants are located in the United States, 74.4%. As shown in Fig 2 below, respondents also responded from Nigeria, Canada, The Netherlands, and Egypt. 61% of participants work in the information technology industry, followed by education at 19.5%; healthcare, automotive, government, and financial sectors had participants as well. Most (54.8%) of participants work in large organizations of 1000+ persons, while team sizes were small to medium (76.2% of participants in teams of 2-20 people).

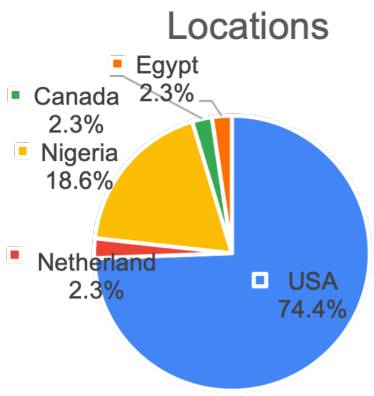

Figure 2: Participants locations

Industry : As expected, more than 61% of participants are working in Information Technology, 19.5% in education, 4.9% in Healthcare, Automotive, and Government, and 2.5% work in the financial industry.

Experience: As shown in Fig 3, we also collected information about the years of experience of the respondents to know their exposure to the industry. 52.4% participants have experience from 1-5 years, 23.8% of them experience from 6-10 years, 19% of them have less than one year experience, and 4.8% of the participants have between 11 and 20 years of experience.

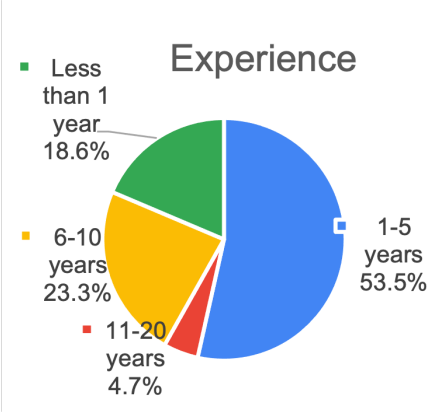

Figure 3: Participant Experience

Size of Organization: We also asked participants about the size of their organizations in terms of staff capacity to understand the type of software model organizations with different staff strengths adopt and how large projects each participant has experienced. 54.8% of participants work in an organization with more than 1000 people, 7.1% work with about 501-1000 people, 16.7% of the participants' organizations have 101-500 staff. 9.5% work with the range 51 to 100, 2.4 % in the range from 11-50, and 9.5% have below ten people in the organization. Our results show that more than 50% of the participants work with large organizations with more than 500 staff and 80% have more than 100 people.

Size of the team: 40.5% of the respondents work within a team size of 6-20 people, 35.7% work less than five team members. 11.9% of respondents work with a team size of 21-30 people. 7.1% in the range of 31-40 team members and 4.8% have more than 40 people in their team. 76.2% of the participants have team members of 20 people or lower.

5.1.1 RQ1: What is most-used software model in organizations?

To know the widely adopted software model among software engineers, we asked the participants about the current models in their various organizations. We also asked if each participant had prior experience with another software model in the past. The majority of the participant, about 72.09% (31 out of 43 respondents), said they are using the Agile model, followed by DevOps at a far distance of 13.95%. 6.98% use the incremental model, while 4.65% of the participants use the waterfall model. When we analyzed the software model previously used, Waterfall and agile models were mainly used by participants in the past, with both having 37.21%. At the same time, DevOps was about 11.63%, and no participant indicated using the incremental model in the past. When we compared the currently used model with the past models used by participants in Fig 4, we could conclude that the waterfall model is rarely used

SE Model	General	Functional Suitability	Compatibility	Usability	Reliability	Security	Maintainability	Portability
Waterfall	5.6	5.2	5.4	5.53	5.2	6.33	5	5.47
Agile	5.63	5.25	5.44	5.31	5.63	5.81	5.07	4.75
DevOps	6	4	6.4	6.4	5.4	5.8	5.6	5.4

Table 1: Average Rating for Past Experience

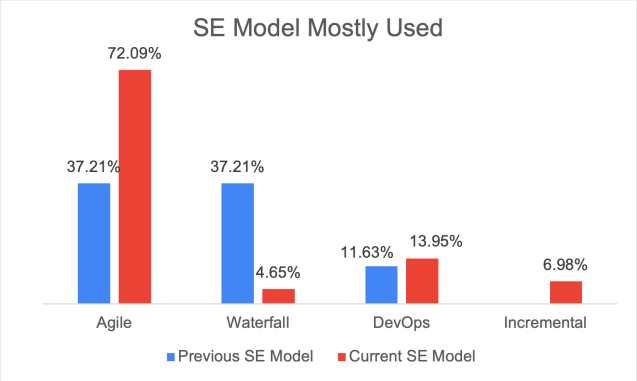

Figure 4: Past and Current Software Engineering Model Participants Used

by organizations today as it has only about one-eighth of its previous impact. However, we can deduce that the adoption of Agile methodology has increased from 37.21% to 72.09%. There is a slight increase in DevOps use while the incremental model is gaining its importance as it has a user percentage of 7.1% compared to none in the past. Most companies are currently choosing the Agile model over any other model.

5.2 Previous Experience

A survey section asked participants whether they had had previous software engineering experience. If they answered yes, they were directed to a few questions about this experience with previously used software models and how it impacts software quality. A developer with knowledge of two or more models has a better frame of reference as to how said models compare to each other. About 85% of our respondents have had previous experience in at least one software model, with 81% of them having a minimum of one year of experience. As expected, most of them have previous experience in Agile (37.21%) and Waterfall (37.21%) models, while 13.95% had experiences in DevOps.

The participants were asked questions about larger umbrellas of software quality attributes. Answers were in a 7-point rating, where 1 represents a poor score, and 7 represents the highest score. There were no participants with experience in the Incremental, Spiral, or other software engineering models. As shown in Table 10, DevOps had the highest mean score of 6.0 in Functional stability, DevOps had a mean score of 6.4 in Compatibility followed by 5.44. DevOps also had the highest score in usability and Maintainability, with a mean score of 6.4 and 5.6. Agile had the best score in Reliability with a mean score of 5.63, followed by waterfall with 5.2 and DevOps with a 4.0 mean score. Waterfall had the highest score in both Security and Portability quality attributes. From the result shown in the table, developers rated DevOps as the average best in terms of software quality.

5.3 Comparison of Currently Used Models

All participants were asked about the software engineering model they currently utilize at their work. As shown in Fig 4, the vast majority of the participants utilize Agile in their work with 72.09%, followed by DevOps at 13.95%. No respondents currently use Spiral or "Other" models. So we asked them to rate the quality of the software model used in terms of the quality attributes defined by [16] standard, which are Functional suitability, Compatibility, Usability, Reliability, Security, Maintainability, and Portability. Apart from these standard attributes, we also included three additional quality attributes that we think are standard terms among developers. We categorized these as General Attributes; Ease of use for the developers, Developer collaboration, and software delivery speed. The questions in the survey for participants to rate the currently used models in terms of quality attributes were more detailed than that of the questions we asked in previous software model experience. For each attribute, we also asked the participants to rate the sub-attributes of these standard quality factors; the result of the survey are discussed below.

General Attributes: We asked participants to rate their choice of software model in terms of ease of use for the developer: How easy is the model to work with?; Collaboration: Does the model allow for collaborations among developers and the entire project team?; and speed of product delivery: How quickly does the model get a product to the customer? The average results of the participants' answers are shown below. We took an average of the scaling results given by the participant for each of the software models as shown in Table 7 below. For ease of use, Waterfall had an average of 6.0, agile with 5.2, DevOps had 6.0, which is the same score as Waterfall, and Incremental had the highest with 6.67. Incremental had the highest score because it had the least number in the choice software model. For collaboration, DevOps had the highest average score with 6.17, while Agile, WaterFall, and Incremental models had average scores of 5.87, 5.0, and 5.0, respectively. DevOps also had the highest mean score of 6.0 in terms of Speed of Delivery followed by Agile with 5.77.

Functional suitability: We asked participants to rate three sub-attributes of the Functional Suitability quality factors in terms of: Defects in production (To what degree does the model reduce defects once in production?); Completeness (To what degree does the software meet its specified requirements); and Correctness - How does the software meet user expectations?. The results of the participants' responses are shown in table 3. For reduction in

SE Model	Ease of use	Collaboration	Delivery speed
Waterfall	6	5	5
Agile	5.2	5.87	5.77
DevOps	6	6.17	6
Incremental	6.67	5	3

Table 2: Average score of General Quality Attributes

production defects, Agile had the highest mean score of 5.74, followed by the Incremental model with 5.33. In contrast, DevOps and Waterfall models had mean score values of 4.8 and 4.5, respectively. For the Completeness of software, developers think of DevOps with a mean score of 5.83. Agile had 5.67, which is the second-highest, Incremental model was 5.33, while the Waterfall model was scored at 5.0. Finally, DevOps also had the best score for the Correctness sub-attribute, followed by Agile, Incremental, and Waterfall.

SE Model	Production Defects	Completeness	Correctness
Waterfall	4.5	5	5
Agile	5.74	5.67	5.74
DevOps	4.8	5.83	5.83
Incremental	5.33	5.33	5.33

Table 3: Average score of Functional Suitability Attributes

Compatibility Participants were asked how compatible they rated software created with their currently used software engineering model. To be precise, we asked how easily the software integrates with other applications. Average scores for compatibility were highest for the DevOps model at 5.67, followed closely by the Agile model at 5.64.

SE Model	Compatibility
Waterfall	5.5
Agile	5.64
DevOps	5.67
Incremental	5

Table 4: Average score of Compatibility Attributes

Usability Participants were asked to rate usability on three attributes. We asked about the initial learnability by the end-users, or how easy is the software to pick up and use; day-to-day usability by the end-users, or how easy is the software to use regularly after learning how to use it; and accessibility, or (generally) how easily do people with disability use the software. The Waterfall model had the highest average scores across all three attributes, while the DevOps model tied the Waterfall model to the day-to-day usability score. For initial learnability, Waterfall scored an average of 6.5, with DevOps having the second-highest score of 6.0. For day-to-day usability, Waterfall and DevOps both scored a 6.0, while Agile followed with a 5.77 score. Finally, for accessibility, Waterfall scored a 5.5, followed by DevOps with a 5.17 score.

SE Model	Learnability	Usability	Accessibility
Waterfall	6.5	6	5.5
Agile	5.16	5.77	4.52
DevOps	6	6	5.17
Incremental	5	5.33	3.33

Table 5: Average score of Usability Attributes

Reliability We also asked participants to rate two sub-attributes of Reliability in factors of Recoverability(How is the software response to failure or other downtimes?) and Software Availability(To what extent does the software provide timely and uninterrupted access of the application?). The average results of the participants' answers are shown in Table 7. In the case of Recoverability, the Waterfall model has the least average score of 4, Agile has 5.42, DevOps has the maximum mean score of all with 5.67, and Incremental with a mean score of 4.33. When Availability is concerned, Waterfall has a least average score of 4.5, Agile with an average score of 5.87, DevOps has the highest mean score of 6.17, and Incremental has an average score of 5.67. On the whole, DevOps has the maximum Reliability, and Waterfall has the least Reliability.

SE Model	Recoverability	Availability
Waterfall	4	4.5
Agile	5.42	5.87
DevOps	5.67	6.17
Incremental	4.33	5.67

Table 6: Average score of Reliability Attributes

Security We asked participants to rate Security in terms of two attributes, Confidentiality(How secure is the software from unauthorized access?) and Integrity(How secure is the software from unauthorized changes?). The average results of the participants' answers are shown in table 7. If we look into Confidentiality, the Waterfall model has the highest mean score of 6.5, Agile has an average score of 5.1, DevOps has a mean score of 5.5, which is the least, and Incremental has an average mean score of 6.33. For Integrity, the Waterfall model and DevOps both have the least mean scores with 5.5, Agile has an average score of 5.9, and Incremental has the highest mean score of 6.33. Waterfall has the highest confidentiality mean score, and Incremental has the highest Integrity mean score. However, when we see Security as a whole attribute, Incremental has the highest Security and DevOps the least.

SE Model	Confidentiality	Integrity
Waterfall	6.5	5.5
Agile	6.1	5.9
DevOps	5.5	5.5
Incremental	6.33	6.33

Table 7: Average score of Security Attributes

Maintainability In Maintainability, participants were given their average scores on testability and Maintainability. The amount to which software responds to software artifacts, such as modules, systems, and designs, enables testing in a specific software test scenario is known as testability. If the artifact is high, it implies that testing is a better way to find vulnerabilities. As of results shown from table 8, DevOps has the highest average score with 5.83, which highest artifact, Agile has 5.6 of the testing score, waterfall and Incremental has very less testing scores with 4.5 and 4.33. This result says that DevOps is best in finding and solving faults in software.

When it comes to Maintainability, we ask: How easily should software systems be adjusted to fix flaws and increase performance with changes in the updated environment?. As if we refer to table 8 maintainability attribute shows that Agile has the highest maintainability average score with 5.77, following agile, DevOps has a 5.5 average score. This implies that both agile and DevOps are good in the Maintainability of software systems. When it comes to both Testability and Maintainability, Agile and DevOps are in the highest position; this says that both models are good in Maintainability quality attribute.

SE Model	Testability	Maintainability
Waterfall	4.5	3.5
Agile	5.6	5.77
DevOps	5.83	5.5
Incremental	4.33	4.33

Table 8: Average score of Maintainability Attributes

Portability Finally, participants will be asked to score the software's reusability and Portability. Reusability is a feature that makes it simple to create new software using the same facilities and components. "Portability" refers to how easily a product may be executed on a different operating or hardware system. According to the average statistics from table 9, DevOps has the highest reusability rate of 5.83, followed by Agile with a 5.42 average score. Every model has a solid average score in Portability, but DevOps gets the highest average score with 5.83. DevOps gets the highest average score in both reusability and Portability, indicating that DevOps is an excellent software paradigm in general.

SE Model	Reusability	Portability
Waterfall	4	5
Agile	5.42	5.52
DevOps	5.83	5.83
Incremental	4.33	5.67

Table 9: Average score of Portability Attributes

5.4 SE Model Evaluation

This section answers the remaining research questions after evaluation of the data from sections 5.2 and 5.3.

5.4.1 RQ2: Is there an effect that the software engineering models Waterfall, Agile, and DevOps have on the quality of the software they produce? If so, what is their effect on quality?

A measure to compare each software engineering model is the average overall quality rating for each model. Tables 10 and 11 show this rating for the previous and current SE model experience. Each model has a distinct score, with the largest difference, 0.7, in rating occurring between the Incremental and DevOps everyday experience. This difference suggests that, yes, the choice of SE model for developing software does affect software quality. Each SE model's effect will be discussed in RQ3 and section 6.

SE Model	Average Rating
Waterfall	5.47
Agile	5.36
DevOps	5.63

Table 10: Overall Average Rating for Previous Experience

SE Model	Average Rating
Waterfall	5.08
Agile	5.50
DevOps	5.74
Incremental	5.04

Table 11: Overall Average Rating for Current Experience

5.4.2 RQ3: What is the difference between waterfall, agile, and DevOps in terms of the quality of software they produce?

Two models represent the difference between the SE models regarding the quality of the software they create.

The first model, represented by Figure 5, considers each model's total score as a percentage of all model's scores for each category. For example, participants' scores for the Agile model's Functionality under current experience constitute 68.9% of the total scores across all models for Functionality under current experience. A combination of many participants having experience in Agile plus high per-participant scores for Agile software attributes result in Agile ranking near 70% across all software quality attributes. DevOps ranks second with scores of approximately 15% across all attributes. The remainder of SE models under current experience falls into single-digit percentages. This first model suggests that Agile produces the highest quality code, considering its popularity, of our measured SE models.

The second model, represented by Tables 10 and 12, considers the average score for each model for each software quality attribute category. This model eliminates the popularity biases found in the first model. For example, the DevOps model resulted in a mean score of 6.06 under Functionality under current experience. On average, each participant who selected DevOps as their currently used SE model gave it a 6.06/7 score. This model states five of the eight quality attribute categories are highest for DevOps under current experience and four under experience. This superiority

suggests that DevOps creates software of higher quality across more attributes when compared to other measured models. However, this result does not consider the significantly wider adoption of the Agile model when compared to all other models as measured.

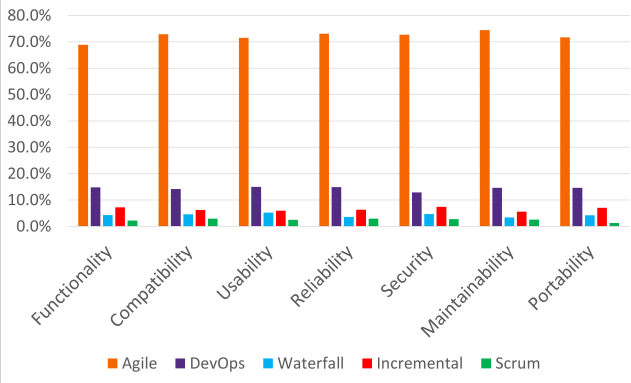

Figure 5: Percentages of Total Score for Current Experience

5.4.3 RQ4: What is the top choice of software model among developers?

We asked the participants to rate different software models regarding developers' preferences to answer this research question. Agile methodology took the lead as the top-choice model with 61%. However, this value is surprisingly lower than the percentages of participants currently using the Agile model in their organizations. DevOps follow this model preference with 22%. However, the percentage value increased when compared with the currently used model. The waterfall model had 7% preference among developers, which further proves that the adoption and preference of the waterfall model have drastically reduced. Although some organizations whose software products are less prone to changes and have well-developed documentation and strategy still prefer to use the waterfall model [28]. The incremental model had 3%, and about 7% of our respondents did not have any software model as their personal preference.

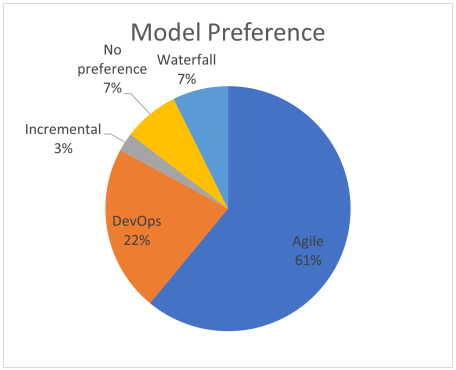

Figure 6: SE Model Preferences

6 DISCUSSION

Our research tried to find the impacts of software development models on software quality. We used the quality model defined by the ISO/IEC 25010:201 standard to rate the impact of these models. First, we asked participants about the software model they have used in the past, and we asked them to rate these software models used previously on their impact on software quality attributes. We saw that Agile and Waterfall models were mainly used by more than half of the participants in the past, followed by DevOps. However, the waterfall model was rated very low in almost all the software quality attributes considered, such as functional stability, compatibility, usability, and maintainability. This is not surprising because, in the Waterfall model, one phase has to be completed before starting another phase. This does not give room for flexibility, and requirement change in the requirement can require that the whole development process is started from the beginning all over again [14]. We also saw from the results that DevOps had the highest average ratings for functional stability, compatibility, usability, and maintainability.

Secondly, we also asked participants about the software development they are currently using in their various organizations, and they also rated these models on their impacts on software quality. 72.09% of the participants responded that they are currently using Agile methodology, which is about twice the increase from the past experience. In contrast, the use of the waterfall model had dropped from 37.21% to 4.65%. From the rating result, which was summarized in Table 11 for all the quality attributes, DevOps also had the highest average rating in compatibility and usability, similar to the rating from past experience. In addition, DevOps was also rated the highest in the General attributes (ease of use, collaboration, and speed of delivery), reliability, and portability. Agile methodology was also rated high for functional suitability and maintainability, while the incremental model was rated high for security. Even though most of our respondents currently use Agile methodology, it was surprising that DevOps had the highest ratings overall. We think this is because the few who chose DevOps gave it excellent scores in all the quality attributes. However, when we considered using the percentage value of the ratings, we saw that Agile methodology had the highest percentages overall for all the quality attributes. The waterfall model was rated the lowest overall, with as low as 4.0 in maintainability.

Finally, we asked the participants about their preferred choice of software development model, and about 61% said they preferred the Agile model the most, while waterfall and incremental models were the least preferred software models among our participants

7 CONCLUSION

From our results, we can say that the DevOps model has the maximum average score in the case of both previous software model experience and current model ratings. Although Waterfall was closely rated with Agile in the past experience, it has the minimum ratings on software quality. The challenge here is that there are not many participants, and because of that, we cannot accurately rate the models based on these ratings. Also, from our results, we cannot conclude why DevOps did not have many organizations using it despite having the highest software quality ratings than Agile.

SE Model	General	Functional Suitability	Compatibility	Usability	Reliability	Security	Maintainability	Portability
Waterfall	5.33	4.83	5.5	6	4.25	6	4	4.5
Agile	5.61	5.72	5.64	5.04	5.65	6	5.69	5.47
DevOps	6.06	5.49	5.67	5.72	5.92	5.5	5.67	5.83
Incremental	4.89	5.33	5	4.55	5	6.33	4.33	5

Table 12: Average Rating for Current Experience

There can be many improvements to the results found in our study with future work. The most significant improvement to our study would be to have more participants overall and include participants with experience in many models and models aside from Agile. This would provide a more well-rounded set of opinions and a better representation of software quality attributes while eliminating the effects of having a small sample size. Each model could be better compared to the other.

A differently formatted questionnaire or overall survey focus can also provide a better comparison between each software engineering model. For example, each participant with experience in multiple software engineering models can rank each model against each other for each quality attribute; this is opposed to our study asking each participant to rank one model. Allowing each participant to compare models directly would allow the personal experiences of each participant to be better represented.

Finally, our study did not collect more personal demographic data. Future work can expand our findings by including age, gender, more work experience, and other biographic information from participants. This would allow better insight into which software engineering model works better for a specific team.

Each software engineering model has its differences and benefits. Each company, team, and project are different and have unique requirements that must be fulfilled. Careful consideration of the choice of software engineering model can ensure the best possible software is produced. Our study suggests that Agile and DevOps can produce high-quality software and work across many different situations. This can be seen by the high adoption rate of the Agile model and high mean ratings for the DevOps model. Agile and DevOps may work for many and should be considered for developing softwares.

REFERENCES

[1] Muhammad Azeem Akbar, Jun Sang, Arif Ali Khan, Muhammad Shafiq, Shahid Hussain, Haibo Hu, Manzoor Elahi, Hong Xiang, et al. 2017. Improving the quality of software development process by introducing a new methodology–AZ-model. *IEEE Access* 6 (2017), 4811–4823.

[2] Alexandra Altvater. 2020. *What Is SDLC? Understand the Software Development Life Cycle.* https://stackify.com/what-is-sdlc/

[3] Tibor Bakota, Péter Hegedűs, Péter Körtvélyesi, Rudolf Ferenc, and Tibor Gyimóthy. 2011. A Probabilistic Software Quality Model. *2011 27th IEEE International Conference on Software Maintenance (ICSM)* (2011), 243–252.

[4] Youssef Bassil. 2012. A simulation model for the waterfall software development life cycle. *arXiv preprint arXiv:1205.6904* (2012).

[5] Raghavi K Bhujang and V Suma. 2017. Analysis of Risk In Software Process Models. *2017 International Conference on Intelligent Sustainable Systems (ICISS)* (2017), 199–204. https://doi.org/10.1109/ISS1.2017.8389397

[6] Elizabeth Bjarnason, Krzysztof Wnuk, and Björn Regnell. 2011. A case study on benefits and side-effects of agile practices in large-scale requirements engineering. In *proceedings of the 1st workshop on agile requirements engineering*. 1–5.

[7] J. Boegh, S. Depanfilis, B. Kitchenham, and A. Pasquini. 1999. A Method For Software Quality Planning, Control, and Evaluation. *IEEE Software* 16, 2 (1999), 69–77. https://doi.org/10.1109/52.754056

[8] Barry W Boehm, John R Brown, and Mlity Lipow. 1976. Quantitative evaluation of software quality. In *Proceedings of the 2nd international conference on Software engineering*. 592–605.

[9] Abhijit Chakraborty, Mrinal Kanti Baowaly, Ashraful Arefin, and Ali Newaz Bahar. 2012. The Role of Requirement Engineering in Software Development Life Cycle. *Journal of emerging trends in computing and information sciences* 3, 5 (2012), 723–729.

[10] K. Chari and M. Agrawal. 2018. Impact of Incorrect and New Requirements on Waterfall Software Project Outcomes. *Empir Software Eng* 23 (2018), 165–185. https://doi.org/10.1007/s10664-017-9506-4

[11] ISO DSTU. [n.d.]. IEC 9126–1: 2013. Prohramna inzheneriya. Yakist produktu. Chastyna 1. Model yakosti (ISO/IEC 9126–1: 2001, IDT).[ISO/IEC 9126–1: 2001. Software engineering. Product quality. Part 1: Quality model]. Kyiv, 2014. 20 p.

[12] John Estdale and Elli Georgiadou. 2018. Applying the ISO/IEC 25010 quality models to software product. In *European Conference on Software Process Improvement*. Springer, 492–503.

[13] American Society for Quality. 2021. *What is Software Quality?* https://asq.org/quality-resources/software-quality

[14] year = 2019 url = https://www.geeksforgeeks.org/software-engineering-failure-of-waterfall-model/ GeeksforGeeks, title = Software Engineering | Failure of Waterfall model. [n.d.]. .

[15] Amran Hossain, Md Abul Kashem, and Sahelee Sultana. 2013. Enhancing software quality using agile techniques. *IOSR Journal of Computer Engineering* 10, 2 (2013), 87–93.

[16] ISO. 2021. *ISO/IEC 25010:2011.* https://www.iso.org/standard/35733.html

[17] Ronald Jabangwe, Henry Edison, and Anh Nguyen Duc. 2018. Software Engineering Process Models for Mobile App Development: A Systematic Literature Review. *Journal of Systems and Software* 145 (2018), 98–111. https://doi.org/10.1016/j.jss.2018.08.028

[18] Parita Jain, Arun Sharma, and Laxmi Ahuja. 2018. The Impact of Agile Software Development Process on the Quality of Software Product. In *2018 7th International Conference on Reliability, Infocom Technologies and Optimization (Trends and Future Directions)(ICRITO)*. IEEE, 812–815.

[19] Goran Jevtic. 2019. *What is SDLC? Phases of Software Development, Models, & Best Practices.* https://phoenixnap.com/blog/software-development-life-cycle

[20] Kamaljeet Kaur, Anuj Jajoo, and Manisha. 2015. Applying Agile Methodologies in Industry Projects: Benefits and Challenges. In *2015 International Conference on Computing Communication Control and Automation*. 832–836. https://doi.org/10.1109/ICCUBEA.2015.166

[21] Gaurav Kumar and Pradeep Kumar Bhatia. 2014. Comparative Analysis of Software Engineering Models from Traditional to Modern Methodologies. In *2014 Fourth International Conference on Advanced Computing Communication Technologies*. IEEE, 189–196. https://doi.org/10.1109/ACCT.2014.73

[22] Yu Beng Leau, Wooi Khong Loo, Wai Yip Tham, and Soo Fun Tan. 2012. Software development life cycle AGILE vs traditional approaches. In *International Conference on Information and Network Technology*, Vol. 37. 162–167.

[23] A. Memon, A. Porter, C. Yilmaz, A. Nagarajan, D. Schmidt, and B. Natarajan. 2004. Skoll: Distributed Continuous Quality Assurance. *Proceedings. 26th International Conference on Software Engineering* (2004), 459–468. https://doi.org/10.1109/ICSE.2004.1317468

[24] Ernest Mnkandla and Barry Dwolatzky. 2006. Defining agile software quality assurance. In *2006 International Conference on Software Engineering Advances (ICSEA'06)*. IEEE, 36–36.

[25] Munassar Nabil, Mohammed Ali, and A Govardhan. 2010. A Comparison Between Five Models Of Software Engineering. *International Journal of Computer Science Issues (IJCSI)* 7, 5 (2010), 94–101.

[26] Suryanto Nugroho, Sigit Hadi Waluyo, and Luqman Hakim. 2017. Comparative analysis of software development methods between Parallel, V-Shaped and Iterative. *arXiv preprint arXiv:1710.07014* (2017).

[27] Pulasthi Perera, Roshali Silva, and Indika Perera. 2017. Improve software quality through practicing DevOps. IEEE, 1–6.

[28] Leo Prada. 2021. *Why waterfall development may still suit your organization.* https://www.itproportal.com/features/why-waterfall-development-may-still-suit-your-organization/

[29] G Gordon Schulmeyer. 2007. *Handbook of software quality assurance.* Artech House, Inc.

[30] S Shylesh. 2017. A study of software development life cycle process models. In *National Conference on Reinventing Opportunities in Management, IT, and Social Sciences*. 534–541.

[31] Amninder Singh and Puneet Jai Kaur. 2019. Analysis of software development life cycle models. In *Proceeding of the Second International Conference on Microelectronics, Computing & Communication Systems (MCCS 2017)*. Springer, 689–699.

[32] Ioannis Stamelos, Lefteris Angelis, Apostolos Oikonomou, and Georgios L Bleris. 2002. Code Quality Analysis in Open Source Software Development. *Information systems journal* 12, 1 (2002), 43–60.

[33] Andrew Stellman and Jennifer Greene. 2005. *Applied software project management.* " O'Reilly Media, Inc.".

[34] Muhammad Asaad Subih, Babar Hayat Malik, Imran Mazhar, Amina Yousaf, Muhammad Usman Sabir, Tamoor Wakeel, Wajid Ali Izazul Hassan, Muhammad Suleman10 Bilal-bin Ijaz, and Hadiqa Nawaz11. 2019. Comparison of agile method and scrum method with software quality affecting factors. *Int. J. Adv. Comput. Sci. Appl* 10, 5 (2019), 531–535.