International Journal of
INTELLIGENT SYSTEMS AND APPLICATIONS IN ENGINEERING
ISSN:2147-6799 www.ijisae.org Original Research Paper

Comparative Analysis of Testing Approach in Context of Agile Software Development

Garima Nahar[1], Dr. Sonal Bordia Jain[2]

Submitted: 04/02/2024 **Revised**: 12/03/2024 **Accepted**: 20/03/2024

Abstract: The Agile technique is becoming widely used by businesses worldwide to develop software products because it promises to provide high-quality products more quickly. Software testing is the most important way to evaluate a product's quality. Software testing in Agile development is still difficult and highly complex. This has mostly occurred as a result of Agile development's lack of emphasis on software testing activities. It emphasises frequent delivery, brief iterations, and client participation. This paper delves into a comparative analysis of testing approaches within the context of ASD. We explore traditional testing methodologies used in waterfall models and contrast them with techniques tailored for the iterative nature of agile projects. Key aspects like test automation, exploratory testing, and continuous integration will be discussed alongside their advantages and limitations in the agile environment. Finally, the paper concludes by outlining a potential framework for agile testing, integrating various techniques for optimal results.

Keywords: *ASD, SDLC, Testing Methods, Software Development*

1. Introduction

The software development landscape has witnessed a significant shift towards agile methodologies. Agile prioritizes continuous delivery of value, customer involvement, and adaptation to evolving requirements. This approach fosters innovation and responsiveness but poses challenges for testing practices traditionally built for a more linear, phased development process.

The Agile Software Development has become the most successful from various effective approaches that have been developed to reach the final software product. Agile Software Development is a lightweight methodology that intends to overcome the drawbacks and limitations of traditional waterfall software development method. It reduces overhead and other operating costs along with providing flexibility to adapt to alterations in requirements at any stage. This is attained via a set of values and principles that govern task management and coordination [1].

While many software development techniques assume that project requirements can be accurately gathered at the initiation of the software project, this is often not possible due to the inherent complexity and unpredictability of software projects. As a result, iterative software development designs are necessary to cope with the multitude of unknown effecting variables. The success of the lean development method in the 1980s led to the emergence of a variety of "iterative" software methods, such as the Unified Process, Evo, Spiral, and Agile methods. These approaches recognize the need for flexibility and adaptability throughout the software development process [2].

The Agile Software Development Methodology has gained popularity in recent years due to its ability to become acquainted to changing requirements and user's needs. In contrast to traditional software development approaches/models that follow a linear and sequential process, the Agile Software Methodology is iterative and incremental. It emphasizes collaboration, flexibility, and adaptableness to change [3].

In the Agile Development Methodology, the software development process is broken down into smaller iterations sometimes called sprints. Each iteration involves a small set of requirements or user stories, which are developed and tested within a short period of time. At the end of each iteration, the development team demonstrates the working software to the customer and gathers feedback. This feedback is used to refine and prioritize the requirements for the next iteration or we can say

[1]*Research Scholar, RTU, Kota and Asst. Professor, Dept. of Computer Science, S. S. Jain Subodh P.G. Mahila Mahavidyalaya, Rambagh Circle, Jaipur*
[2]*Associate Professor, Dept. of Computer Science, S. S. Jain Subodh P.G. College, Jaipur*
Email:dagagarima@gmail.com1, Sonalbordiajain@gmail.com2

each iteration involves a full software devilment life cycle including all phases [4].

Agile Software Development Methodologies are widely used for highly collaborative software development. The Agile development approach is commonly associated with "lean" engineering and emphasizes activities that contribute directly to the project's end goal of delivering high-quality software that meets business needs. The Agile Manifesto, published in 2001, defines the approach known as Agile Software Development and serves as an influencing and guiding force for Agile professionals. The manifesto was created by 17 influential figures, some of whom formed the Agile Alliance [5]. The manifesto established a common set of overarching values and principles for all individual Agile Methodologies at the time.

This paper aims to analyze various testing approaches in the context of agile software development. We will explore the limitations of traditional methodologies and delve into agile-specific testing techniques. Through a comparative analysis, we will identify the strengths and weaknesses of each approach, paving the way for a more comprehensive testing strategy within agile projects.

1.1 Traditional Testing Methodologies

Traditional software development methodologies, like the waterfall model, follow a well-defined, sequential approach. Requirements are meticulously documented before the development phase. Testing occurs towards the end of the development cycle, often as a separate and distinct phase following development completion. This approach utilizes the following testing methods:

- **Unit Testing:** Focuses on validating the functionality of individual software units (modules, functions). Unit testing is typically performed by developers themselves.
- **Integration Testing:** Verifies how integrated software modules interact with each other.
- **System Testing:** Evaluates the entire software system against the defined requirements.
- **Acceptance Testing:** Confirms whether the software meets the user's acceptance criteria.

While these methods serve a purpose, they are not readily adaptable to the dynamic nature of agile development. Rigorously defined requirements at the project's outset are often unrealistic in agile projects, and testing at the end of each iteration may be time-consuming and inefficient.

1.2 Agile Testing Techniques

Agile testing methodologies aim to integrate seamlessly with the iterative development cycles of agile projects. Here are some key approaches:

- **Test-Driven Development (TDD):** Involves writing unit tests before writing the actual code. This ensures the code meets functional requirements from the outset and aids in code refactoring.
- **Behavior-Driven Development (BDD):** Focuses on defining user stories and acceptance criteria in a collaborative manner. BDD facilitates the creation of automated acceptance tests based on user stories.
- **Exploratory Testing (ET):** Involves an informal, session-based approach to testing. Testers actively explore the functionality of the software, using their experience and judgment to identify bugs.
- **Continuous Integration (CI):** Automates the process of integrating code changes from multiple developers into a central repository. CI typically involves running automated tests after each commit, providing immediate feedback on potential regressions.
- **Continuous Delivery/Continuous Deployment (CD):** Automates the process of deploying new software versions to production environments. CD allows for frequent releases with minimal manual intervention.

2. Agile Methodology

Agile methodologies have gained popularity in recent years due to their ability to enhance collaboration, improve productivity, and deliver software that meets business and user needs. Research has shown that Agile Development Methodologies are very effective in managing software development projects by allowing for flexibility and adaptability, achieving higher customer satisfaction, speedy time-to-market, and also increased return on investment [6, 7].

Agile methodologies provide a framework for highly collaborative software development that firmly follows the flow of business value, with a focus on activities that directly contribute to the

software project final goal i.e. quality software product. The Agile Manifesto, with its core values and principles, provides guidance for Agile developers, and has proven to be an effective approach to software development [8].

User-Centered Design (UCD) and Human-Computer Interaction (HCI) are also important components of Agile development that facilitate the development of efficient and user-friendly software products [9].

Usability Engineering is an important aspect of software design that has gained significant attention in recent years. According to O. Sohaib and K. Khan (1988), Usability Engineering is the process of designing and evaluating products, systems, or services to ensure that they meet the needs and expectations of their intended users. This process involves conducting user research, gathering feedback, and also testing the product with real users to identify areas that need improvement [10].

Usability Engineering is commonly used in the design and development of digital products, including software applications and websites. However, it can also be applied to physical products and services to enhance their usability and accessibility. As noted by O. Sohaib and K. Khan (2010), Usability Engineering plays a significant role in improving user experience, which can lead to increased user satisfaction and adoption [11]. Incorporating Usability Engineering into the design process can help organizations create products that are more effective, efficient, and enjoyable to use. By focusing on the needs and expectations of their intended audience, organizations can develop products that are user-friendly and accessible, thereby enhancing user interaction, experience and satisfaction.

Usability Testing has been recognized as a fundamental part of various software development methodologies, including Agile Software Development, and Usability Engineering [12]. The main objective of Usability Testing is to evaluate whether the software is user-friendly and meets the needs of its users [13].

In Agile Software Development, Usability Testing is an important phase of the development process, and it is conducted in each sprint to identify any usability issues early on [14]. This approach allows for quick feedback and iteration to ensure that the final product meets the user's needs and is easy to use. The testing is conducted to identify any usability issues and to evaluate the overall performance of the software.

In Usability Engineering, Usability Testing is used to ensure that the system is easy to use and meets the needs of its users [15]. The testing is conducted at various stages of the development process, from design to post-launch, to ensure that the system is user-friendly and meets the user's needs. Usability Testing is an important aspect of software development, and it has been shown to improve user satisfaction and adoption rates [16]. Usability Testing can contribute to the success of software projects.

2.1 Agile Software Development Life Cycle

Agile Software Development Life Cycle is an incremental and iterative approach to software development that emphasizes flexibility, collaboration, and continuous improvement. It is a departure from traditional waterfall development methodologies and is well-suited for software projects with evolving requirements and dynamic environments. The Agile project team continuously refines and reprioritizes the backlog based on feedback and evolving requirements. Agile SDLC promotes collaboration, adaptability, and transparency throughout the software development process. By embracing change and focusing on delivering value incrementally, Agile SDLC enables teams to respond quickly to user needs and deliver high-quality software products [17]. The Agile Development Life Cycle encompasses a sequence of six distinct stages that aptly capture the iterative and collaborative essence of Agile Development methodologies. These stages can be broadly explained as follows [17] [18]:

- Project Discovery and Prioritization
- Planning and Requirement Refinement
- Development and Iteration
- Release and Deployment
- Production and Maintenance
- Retirement or Transition

The Agile Development Life Cycle represents a dynamic journey characterized by constant adaptation, collaboration, and incremental progress. It serves as a robust framework for magnificently delivering software solutions that associate with evolving requirements and user expectations. These

stages in the Agile Development Life Cycle reflect the dynamic and iterative nature of Agile methodologies, allowing for continuous improvement, customer collaboration, and the delivery of valuable software increments.

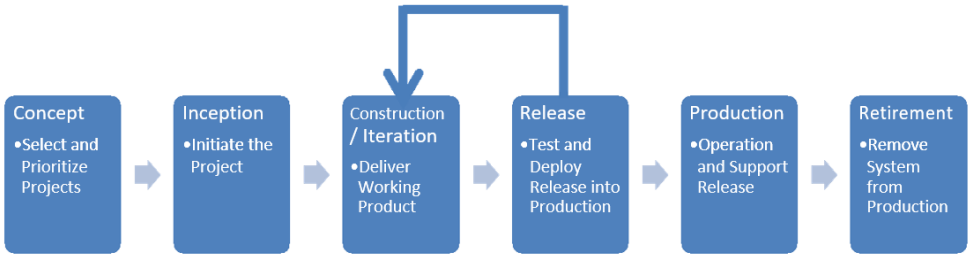

Fig 1: Agile Process

3. Related Work

The concept of the "Paradox of the Active User" was invented by John M. Carroll and Mary Beth Rosson, at IBM. This concept emerged as a way to explain a recurring observation noted in numerous user studies conducted at the IBM User Interface Institute in the early 1980s. According to authors, users tend to skip manuals and start using software right away. This behavior pattern has been validated by research and other studies. Their motivation lies in promptly commencing their tasks and achieving immediate objectives. Their focus is on the task at hand, rather than on the intricacies of the system itself [22].

Najmeh Ghasemifard et.al (2015) proposed Curative Usability Test Methods for Usability Testing. The evaluation criteria encompass a range of factors that contribute to effective Usability Testing. To conduct successful tests, it's important to consider several key factors. To ensure a smooth user experience, completing tasks quickly and efficiently is essential. Cost-effectiveness is also important to keep expenses low. Utilizing flexible tools and frameworks that can adapt to different situations is important. Adequate resources must be available to conduct tests. It is necessary to determine the appropriate number of tests for a comprehensive evaluation. When deciding between experimental and analytical test types, the specific situation should be taken into consideration. The evaluator's experience and expertise can impact test outcomes. Finally, categorizing identified issues based on severity, distinguishing between major and minor problems, and defining the purpose parameters of the chosen testing method are all essential steps. [23]

Geisen, E. et. al. (2017) discussed that in later rounds of testing, a common technique involves a combination of cognitive and usability testing to maximize efficiency. This approach leverages methods like "Think Aloud" and verbal probing to enhance error detection. However, the depth of verbal probing can impact usability metrics. To achieve a balance between priorities and optimize usability metrics, retrospective cognitive probes may be employed, whereas concurrent probing is more apt for a cognitive interviewing focus. It is crucial to carefully manage the balance between cognitive insights and usability metrics while taking into account the timing of probes when using techniques like "Think Aloud" and verbal probing [24].

M.J. Van den Haak et. al. (2004) study presented a comparative analysis of three usability test approaches: concurrent Think-Aloud protocols, retrospective Think-Aloud protocols, and constructive interaction. The assessment was conducted on an online library catalogue, considering four aspects: the number and type of usability issues detected, the importance of the problems found, overall task performance, and participant feedback [25].

Nichols, Elizabeth, et al. (2020) shared their current practices in Usability Testing online surveys at the Census Bureau. These practices have evolved due to technological shifts, sponsor requirements, insights from UX literature, and practical experience. The authors explored the connection between research on Think-Aloud methodology and participant numbers, and how experience influences it. However, there are challenges when incorporating usability concepts in social survey tools. Developing a comprehensive satisfaction score that aligns with participant behavior can enhance evaluation and comparisons. Utilizing templates for eye-tracking data and theory development may aid in measuring effectiveness, and incorporating paradata systematically can sharpen the focus on Usability

Testing. The evolution of usability practices offers valuable insights for survey designers to enhance design features and minimize errors. By carefully observing user actions and feedback, designers can gain a deeper understanding of how to improve surveys [26].

Franz, Rachel et. al. (2019) thoroughly explored the significance of Usability Testing in evaluating technologies for individuals aged 65 and above. The authors share their practical experience and knowledge from literature to offer a range of effective strategies for designing, conducting, and analyzing usability tests. These strategies encompassed selecting appropriate test locations for the Co-discovery approach, devising questions, choosing relevant testing techniques, and using mixed-methods approaches. The authors also addressed challenges like recruitment and participant impressions, utilizing insights from Human-Computer Interaction and sociology. Usability Testing, essential for refining technology and ensuring positive user experiences, provided insights into how older adults integrate technology into their lives. While acknowledging limitations in sample sizes common in research involving frail older adults, the findings contribute valuable insights for improving usability methods and best practices. These insights could guide the study of emerging technologies, with the chapter urging researchers to continually enhance User-Centered Design methods tailored to older adults [27].

Banker, Andria et. al. (2022) structured literature review on Usability Testing for children, the authors explored the evolution of practices over time and the distinct variations between children's interactions with prototypes compared to those of adults. The study identifies potential avenues for future research, including longitudinal vs. cross-sectional testing, distinguishing between physical and digital product testing, and determining suitable age ranges for child participants. It underscores the significance of transparency and possible bias in testing, while emphasizing the imperative of valuing children's instincts and opinions to enhance product design [28].

Kirkscey, Russell (2022) utilized a mixed-methods approach to examine the creation, Usability Testing, and user experience assessment of an mHealth app that aims to educate older women about diagnosis, treatment, and prevention of osteoporosis. To cater to the needs of older users, the app was designed with Universal DesignUsability Tests were conducted to evaluate functional, informational, and navigational tasks. The data collected included audio transcript records, video observer notes, task completion times, and a post-test survey evaluating user experience. Users interacted effectively and comfortably with the app. Results showed that users found the app effective and comfortable to use, but there were certain challenges that could be addressed in future iterations. The framework of the study, which incorporated both qualitative and quantitative elements, can provide valuable guidance for other researchers who will be developing similar mHealth products [29].

Rahmawati, A.F. et. al. (2022) explained the evaluation of the SiNovi website's User Experience (UX) after conducting Moderated Remote Usability Testing and a User Experience Questionnaire (UEQ) revealed several important findings. The results showed that while the website received positive evaluations in categories like "Attractiveness," "Efficiency," "Dependability," and more, indicating a good overall user experience. However, there are areas that require improvement to reduce the number of issues and enhance user satisfaction [30].

Khalid, Md Saifuddin et. al. (2023) examined the complexities of designing and evaluating adaptive learning systems from a usability perspective, considering adaptability and diverse stakeholder requirements. Students and educators were increasingly involved in educational quality assessments, often without perceiving value. Few case studied report on usability evaluations for such systems [31].

4. Comparative Analysis

Here's a breakdown of the methodology for a comparative analysis of testing approaches in Agile Software Development:

1. Define Scope and Objectives:

- **Project Type:** Specify the type of Agile methodology used (Scrum, Kanban, etc.)
- **Testing Techniques:** Identify the testing approaches you'll compare (e.g., exploratory testing, BDD, etc.)
- **Evaluation Criteria:** Determine the factors for comparison (e.g., defect detection rate, test automation coverage, etc.)

2. Research and Data Collection:

- **Agile Testing Practices:** Research best practices for testing in your chosen Agile methodology.
- **Testing Techniques:** Gather information on the strengths and weaknesses of each testing approach you'll compare.
- **Case Studies:** Look for case studies where different testing approaches were used in Agile projects.

3. Develop Evaluation Framework:

- **Metrics:** Define clear metrics for each evaluation criterion (e.g., number of defects found, percentage of automated tests).
- **Weighting:** Assign weights to each criterion based on their importance to your specific project.

By following this methodology, conduct a comprehensive comparison of testing approaches and select the one that best aligns with Agile software development project.

Table 1: Comparative Analysis of Testing Approaches

Feature	Traditional Testing	Agile Testing
Focus	Requirements verification	Continuous feedback and improvement
Timing of Testing	Separate phase	Integrated throughout development cycle
Level of Automation	Limited automation	Emphasis on automation
Adaptability to Changing Requirements	Difficult	Highly adaptable
Efficiency in Agile Environment	Low	High

Advantages of Agile Testing:

- **Early Feedback:** Agile testing methods provide feedback throughout the development cycle, leading to faster bug detection and resolution.
- **Improved Quality:** Continuous testing ensures a higher quality product by identifying issues early on.
- **Reduced Risk:** Frequent deployments and automated testing minimize the risk of introducing regressions.
- **Increased Collaboration:** Testing becomes an integral part of the development process, fostering better communication between developers and testers.

Limitations of Agile Testing:

- **Test Automation Overhead:** Creating and maintaining a robust suite of automated tests can be time-consuming.
- **Technical Expertise:** Agile testing techniques may require specialized skills and knowledge from testers.
- **Over-reliance on Automation:** Over-dependence on automation may neglect the benefits of exploratory testing.

Table 2: Comparative Analysis of Testing Approach in Different Software Development Models

Feature	Waterfall Model	Agile Model	Spiral Model
Testing Phases	Sequential (Unit -> Integration -> System -> Acceptance)	Iterative and Incremental (Testing throughout development)	Risk-driven, integrates testing throughout the project lifecycle
Testing Techniques	Primarily black-box testing (focus on requirements)	Mix of black-box and white-box testing (adapts based on iteration)	Combination of black-box, white-box, and other techniques based on risk assessment
Documentation	Formal test plans and test cases created upfront	Test plans and cases evolve with each iteration	Test plans and cases created and updated based on risk assessment
Defect Management	Defects found later in the process can be expensive to fix	Early defect detection and correction due to continuous testing	Focus on mitigating high-risk defects early
Change Management	Difficult to accommodate changes due to sequential nature	Easier to incorporate changes due to iterative approach	Adaptable to changes based on risk evaluation
Advantages	Well-defined process, easy to manage for simple projects	Flexible, faster feedback loop, good for complex or evolving projects	Mitigates risk early, good for large, high-risk projects
Disadvantages	Inflexible, difficult to adapt to changes, late defect detection	Requires strong discipline and communication, can be resource-intensive	More complex to manage, requires good risk assessment skills

5. Conclusion

This comparative analysis highlights the strengths of agile testing approaches in the context of agile software development. Agile testing prioritizes:

- **Continuous Integration and Delivery (CI/CD):** Frequent testing throughout development cycles ensures early defect detection and faster fixes.
- **Collaboration:** Close interaction between developers, testers, and stakeholders fosters a shared understanding of evolving requirements and leads to more effective testing strategies.
- **Adaptability:** Agile testing readily adjusts to changing priorities and accommodates new features introduced during sprints.

While traditional, plan-driven testing methodologies might offer more structure in specific contexts, agile testing fosters a more responsive and efficient approach within the iterative nature of agile development.

References

[1] V. Szalvay, “An Introduction to Agile Software Development,” 2004.

[2] A. Choday and C. Dwivedula, “A Systematic Literature Review and Industrial Evaluation of Incorporating Lean Methodologies in Software Engineering”.

[3] K. Beck and C. Andres, *Extreme programming explained: embrace change*, 2nd ed. Boston, MA: Addison-Wesley, 2005.

[4] M. Cohn, *Succeeding with agile: software development using Scrum*. Pearson Education, 2010.

[5] A. Cockburn, *Agile software development: the cooperative game*. Pearson Education, 2006.

[6] R. C. Martin, *Agile software development, principles, patterns, and practices*, First edition, Pearson new international edition. Harlow: Pearson, 2014.

[7] Beck, K., Beedle, M., van Bennekum, A., Cockburn, A., Cunningham, W., Fowler,

M., Grenning, J., Highsmith, J., Hunt, A., Jeffries, R., "Manifesto for Agile Software Development.," 2001, [Online]. Available: https://agilemanifesto.org/

[8] S. W. Ambler, *Agile modeling: effective practices for eXtreme programming and the unified process*. New York: J. Wiley, 2002.

[9] J. D. Gould and C. Lewis, "Designing for usability: key principles and what designers think," vol. 28, no. 3, 1985.

[10] J. Whiteside, J. Bennett, and K. Holtzblatt, "Usability Engineering: Our Experience and Evolution," in *Handbook of Human-Computer Interaction*, Elsevier, 1988, pp. 791–817. doi: 10.1016/B978-0-444-70536-5.50041-5.

[11] O. Sohaib and K. Khan, "Integrating usability engineering and agile software development: A literature review," in *2010 International Conference On Computer Design and Applications*, Qinhuangdao, China: IEEE, Jun. 2010, pp. V2-32-V2-38. doi: 10.1109/ICCDA.2010.5540916.

[12] K. Curcio, R. Santana, S. Reinehr, and A. Malucelli, "Usability in agile software development: A tertiary study," *Comput. Stand. Interfaces*, vol. 64, pp. 61–77, May 2019, doi: 10.1016/j.csi.2018.12.003.

[13] S. Roy and P. K. Pattnaik, "Some Popular Usability Evaluation Techniques for Websites," in *Proceedings of the International Conference on Frontiers of Intelligent Computing: Theory and Applications (FICTA) 2013*, S. C. Satapathy, S. K. Udgata, and B. N. Biswal, Eds., in Advances in Intelligent Systems and Computing, vol. 247. Cham: Springer International Publishing, 2014, pp. 535–543. doi: 10.1007/978-3-319-02931-3_61.

[14] M. Düchting, D. Zimmermann, and K. Nebe, "Incorporating User Centered Requirement Engineering into Agile Software Development," in *Human-Computer Interaction. Interaction Design and Usability*, J. A. Jacko, Ed., in Lecture Notes in Computer Science, vol. 4550. Berlin, Heidelberg: Springer Berlin Heidelberg, 2007, pp. 58–67. doi: 10.1007/978-3-540-73105-4_7.

[15] J. Nielsen, "The usability engineering life cycle," *Computer*, vol. 25, no. 3, pp. 12– 22, Mar. 1992, doi: 10.1109/2.121503.

[16] M.-L. Sánchez-Gordón and L. Moreno, "Toward an Integration of Web Accessibility into Testing Processes," *Procedia Comput. Sci.*, vol. 27, pp. 281–291, 2014, doi: 10.1016/j.procs.2014.02.031.

[17] Y. B. Leau, W. K. Loo, W. Y. Tham, and S. F. Tan, "Software Development Life Cycle AGILE vs Traditional Approaches".

[18] De Vicente Mohino, Bermejo Higuera, Bermejo Higuera, and Sicilia Montalvo, "The Application of a New Secure Software Development Life Cycle (S-SDLC) with Agile Methodologies," *Electronics*, vol. 8, no. 11, p. 1218, Oct. 2019, doi: 10.3390/electronics8111218.

[19] "The Stages of the Agile Software Development Life Cycle," *The Stages of the Agile Software Development Life Cycle*. https://www.lucidchart.com/blog/agile-software-development-life-cycle

[20] "Types of Usability Testing," *Types of Usability Testing*. https://www.uxtweak.com/usability-testing/types/

[21] G. Nahar and S. Bordia Jain, "Uncovering the Usability Test Methods for Human–Computer Interaction," vol. 681, no. 1, p. 57, doi: https://link.springer.com/chapter/10.1007/978-981-99-1909-3_6.

[22] J. M. Carroll and M. B. Rosson, "Paradox of the active user.," in *Interfacing thought: Cognitive aspects of human-computer interaction.*, Cambridge, MA, US: The MIT Press, 1987, pp. 80–111.

[23] N. Ghasemifard, M. Shamsi, and A. R. R. Kenari, "A New View at Usability Test Methods of Interfaces for Human Computer Interaction," 2015.

[24] E. Geisen and J. Romano Bergstrom, "Think Aloud and Verbal-Probing Techniques," in

Usability Testing for Survey Research, Elsevier, 2017, pp. 131–161. doi: 10.1016/B978-0-12-803656-3.00006-3.

[25] M. J. Van den Haak, M. D. T. de Jong, and P. J. Schellens, "Employing think-aloud protocols and constructive interaction to test the usability of online library catalogues: a methodological comparison," *Interact. Comput.*, vol. 16, no. 6, pp. 1153–1170, Dec. 2004, doi: 10.1016/j.intcom.2004.07.007.

[26] E. Nichols, E. Olmsted-Hawala, T. Holland, and A. A. Riemer, "Usability Testing Online Questionnaires: Experiences at the U.S. Census Bureau," in *Advances in Questionnaire Design, Development, Evaluation and Testing*, P. Beatty, D. Collins, L. Kaye, J. L. Padilla, G. Willis, and A. Wilmot, Eds., Hoboken, NJ, USA: John Wiley & Sons, Inc., 2019, pp. 315–348. doi: 10.1002/9781119263685.ch13.

[27] R. Franz and B. B. Neves, "Usability Is Ageless: Conducting Usability Tests with Older Adults," in *Ageing and Digital Technology*, B. B. Neves and F. Vetere, Eds., Singapore: Springer Singapore, 2019, pp. 99–114. doi: 10.1007/978-981-13-3693-5_7.

[28] University of Minnesota-Twin Cities, United States of America and A. Banker, "Usability testing with children: History of best practices, comparison of methods and gaps in literature," presented at the DRS2022: Bilbao, Jun. 2022. doi: 10.21606/drs.2022.646.

[29] R. Kirkscey, "Development and Patient User Experience Evaluation of an mHealth Informational App for Osteoporosis," 2021.

[30] A. F. Rahmawati, T. Wahyuningrum, A. C. Wardhana, A. Septiari, and L. Afuan, "User Experience Evaluation Using Integration of Remote Usability Testing and Usability Evaluation Questionnaire Method," presented at the 2022 IEEE International Conference on Cybernetics and Computational Intelligence (CyberneticsCom), IEEE, 2022, pp. 40–45.

[31] M. S. Khalid, T. A. B. Tretow-Fish, and A. Roark, "Usability Evaluation of Adaptive Learning System RhapsodeTM Learner," presented at the Proceedings of International Conference on Information and Communication Technology for Development: ICICTD 2022, Springer, 2023, pp. 71–82.

[32] Bhawana Verma, S.K.A.. (2019). Design & Analysis of Cost Estimation for New Mobile-COCOMO Tool for Mobile Application. *International Journal on Recent and Innovation Trends in Computing and Communication*, *7*(1), 27–34. https://doi.org/10.17762/ijritcc.v7i1.5222