Continuous Software Engineering for Augmented Reality

Suzanna[1], Sasmoko[2], Ford Lumban Gaol[3], Tanty Oktavia[4]
Computer Science Department-BINUS Graduate Program-Doctor of Computer Science,
Bina Nusantara University, Jakarta, Indonesia, 11480[1, 2, 3]
Information Systems Department-Binus Online Learning, Bina Nusantara University, Jakarta, Indonesia, 11480[1]
Information System Management Department-BINUS Graduate Program-Master of Information System Management,
Bina Nusantara University, Jakarta, Indonesia 11480[4]

***Abstract*—Continuous software engineering is a new trend that has attracted increasing attention from the research community in recent years. In software engineering there are "continuous" stages that are used depending on the number of artifact repositories such as databases, meta data, virtual machines, networks and servers, various logs, and reports. Augmented Reality (AR) technology is currently growing rapidly. We can find this technology in various fields of life, but unfortunately sustainable software engineering for Augmented Reality is not found. The method shown in previous research is a general method in software engineering so that a theory is needed for sustainable software engineering for AR considering that AR is not just an ordinary application but there are 3D elements and specific components that must be met so that it can be called AR. The main idea behind this research is to find a continuous pattern from the stages of the existing method so far. For example, in general the stages of system development are planning, analysis, design, implementation and maintenance. Then after the application has been built, does it finish there? As we know software always grows and develops according to human needs. Therefore, there are continuous stages that must be patterned so that the life cycle process can be maintained. In this paper we present our initial findings about the continuous stages of continuous software engineering namely continuous planning, continuous analysis, continuous design, continuous programming, continuous integration, and continuous maintenance.**

Keywords*—Continuous software engineering; augmented reality; method in software engineering; continuous planning; continuous analysis; continuous design; continuous programming; continuous integration; continuous maintenance

I. Introduction

Continuous software engineering is performed to avoid discontinuities between development and deployment to continuously perform continuous development [1]. Continuously means without interruption, so the application delivery processing time can be maintained [2]. In continuous software engineering, software developers can flexibly implement several changes from coding, testing and deployment so that if there are problems such as detecting bugs, requesting changes and other unexpected actions will be resolved more quickly. In addition, developers can also test more often and implement some changes. Instead of creating new code, developers can leverage code from an already developed version of the software. All changes or additions to the code will be integrated and then tested for functionality. If it passes, the deployment process produces output in the form of the latest version of the application. In this case, all parties are involved in collaborative development until the software is ready for release [3].

A method is a systematic way or technique used to do something. Methods can be applied to one or more of the existing processes in the methodology. Meanwhile, methodology is a unity of methods, rules and stages used by science, art, and other scientific disciplines [4]. From this understanding, systematic and regular steps are needed in solving problems with the purpose of clearly know the procedures and sequences taken [5]. Thus, the results of problem solving can be optimal. To build software engineering, it is necessary to have automatic or semi-automatic support for carrying out the method so that computer-assisted software engineering can be built [6]. In practice, however, it is often the case that using the same methods does not produce the same results or, in other words, does not affect a software project in a comparable way [4].

Augmented Reality is a technology capable of displaying a visual spectrum in real-time which is currently developing rapidly. Its presence not only brings different nuances in the human visual screen but is also able to combine several technologies to bring digital information into visual perception [7]. To build an AR project, a special method is needed because not all digitally enhanced media today is "Augmented" reality. 2D images or Photoshop conversions are not AR. As is the case with images produced from television or films such as "Jurassic Park" and "Avatar" are also not AR. In contrast, a game that uses live feed and provides a real experience is AR. Not all visual searches can be categorized as AR. Therefore, something is said to be AR if it has the main characteristics of the AR system, namely [8]:

- Real time interactivity
- Use of 3D virtual elements
- Mixture of virtual elements with real elements

According to research conducted by Bean [9], the 3D animation industry, such as film, games, television, advertising and visual effects, uses three main stages in each segment that must be carried out. The three stages are pre-production, production, and post-production. In the pre-production stage,

the team will think about ideas and make sure to tell a compelling story. Determination of character, environment and other supporting factors need to be made as attractive as possible. At the production stage, if the previous stages have been well structured and mature, the project can be carried out. Then in the post-production stage, 2D visual effects and 3D animations are created. In this stage color correction, sound, and other effects need attention so the project will look fantastic. In addition to these three stages, it is necessary to consider the software and hardware components that will be used to create AR application contexts [10].

II. Literature Review of Software Methods

Software is a computer program that is a key component of most businesses such as products, services, industrial processes, and back-office functions. Responding to changing customer needs and market conditions requires the ability to improve software quality quickly and continuously. This approach is known as continuous software engineering [11].

Continuous software engineering is a paradigm for streamlining software engineering by conducting gradual and continuous delivery of software with the purpose of quickly get feedback from customers and improve software quality according to customer requirements [3]. With this gradual and continuous method, the potential for improvement is easier to plan and strengthen. Smaller portions of software updates allow developers to focus on a collaborative, experience-based business model. The essence of continuous software engineering is continuous integration and delivery. This end principle is expected to cover the entire software engineering process from initiation, analysis, development, integration, validation, verification, delivery, to gathering feedback and planning the next software iteration step.

The stages in the software method generally consist of planning, analysis, design, and programming stages [4]. In the last two decades there has been a change in the stages of software development to address challenges that occur in the field. Based on the results of surveys and interviews conducted by Fitzgerald [1], it is known that only 6% of software developer practitioners apply formally defined methods. The rest do not follow the rules of the established method. Then what causes it? Why is there a difference between the method described and the one implemented? From the results of the literature review there are those who argue that software practitioners do not get sufficient knowledge, are not educated, and do not receive proper training. In the research conducted by Parnas [12] in his writings, he acknowledged that the reasons for non-compliance from software practitioners observed were disciplinary methods and norms.

From the application that has been carried out in previous research it is known that the agile method has the advantage of being more flexible and able to accept change. Its extraordinary adaptability made this method quickly accepted in today's software environment [13]. Very complex software projects require integrated, distributed teams. This plays an important role as a bridge between the developer and other stakeholders so that if there is an error it will be easily detected and immediately corrected. The software development method aims to obtain quality and conformity to user requirements[14]. Currently the need for integrated practice has increased. As an example, for extreme agile methods where their popularity is steadily increasing. Then the DevOps method recognizes that software development and operational deployment must be continuous [15]. From the various results of the literature that has been reviewed, it is known that there is a relationship between business strategy and software development. These two things must be continuously assessed and improved. In research conducted by Fitzgerald [1] the combination of the two is abbreviated as BizDev.

The software development methods commonly used by AR today and which will be used as comparison material are as follows, Table I:

a) Waterfall
b) Agile
c) Scrum
d) RAD
e) Prototype
f) DevOps
g) Kanban
h) Rational Unified Process
i) Spiral
j) 3D Pipeline Production

AR technology is still relatively new for software developers to develop, and it is not easy to choose the right development method. According to Dennis, there are several considerations in choosing a software development method based on several criteria, namely: clarity of user needs, mastery of technology, level of system complexity, level of system reliability, execution time and visibility of the implementation schedule [16]. In research conducted by Fitzgerald [1] on continuous software engineering, it was found that there are four activities, namely, strategy and business planning, development, operation and improvement and innovation. Of the four main activities, there are sub-sections as follows:

Strategy and business planning
- *Continuous planning*
- *Continuous budgeting*

Development
- *Continuous integration*
- *Continuous delivery*
- *Continuous deployment*
- *Continuous verification*
- *Continuous testing*
- *Continuous compliance*
- *Continuous security*
- *Continuous evolution*

Operation
- *Continuous use*
- *Continuous trust*
- *Continuous run-time monitoring*

Improvement and Innovation
- *Continuous improvement*
- *Continuous innovation*
- *Continuous experimentation*

TABLE I. COMPARISON OF EACH METHOD

Method	Superiority	Weakness
Waterfalls	1. Easy in management because almost whole requirements have been identified and documented. 2. Sequential stages linearly, complete identification and documentation, making the process easy understood by all the team involved or project owner.	1. No flexibility to change occurring needs in stage development system. 2. Almost no tolerance errors, especially in stages planning and designing.
Spiral	1. High amount of risk analysis. 2. Good for large projects and mission critical.	1. Can be an expensive model for use. 2. Risk analysis needs very specific skills. 3. Success projects are highly dependent on the risk analysis stage.
Agile Development	1. Method light in accordance small project and medium project. 2. Produce cohesion good team. 3. Emphasize final product. 4. Approach based test for terms and guarantees quality.	1. Not suitable for handling complex dependencies. 2. There is risk to sustainability, maintenance, and time longer. 3. Needed mature plans and roles leader existing project experienced.
Scrums	1. Scrums can help a team finish results project fast and efficiently. 2. Scrum confirmed the use of time and money effective. 3. Developed code and tested during the sprint review. 4. Teams earn clear visibility through scrum meetings. 5. Scrums, with nimble, adopt bait come back from customers and stakeholders' interests. 6. Short sprints possible change based on bait come back with easier.	1. Scrums often lead to scope creep because lack of date definite end. 2. Opportunity failure project high if individual have no committed or cooperative. 3. Only can succeed with members who have experienced team. 4. Meeting daily sometimes makes member team frustrated. 5. Quality difficult applied until team through an aggressive testing process
RAD	1. Efficiency time delivery. 2. Change needs can accommodate. 3. Cycle time can be short with the use of powerful RAD tools. 4. Use tools and frameworks.	1. Complexity management. 2. Suitable for system-based components and measurable. 3. Need involvement users all over cycle system. 4. Requires highly skilled personnel. 5. Dependency high on modeling ability.
Prototype	1. Accurate identification requirements because it has input from the project owner so that the appearance of the resulting prototype can be adjusted to the immediate needs of the user. 2. Errors and redundancies can be minimized because of the good identification process to prototype shape.	1. Each evaluation and input for prototype changes will add to the complexity of the system being developed. 2. Give burden addition to programmers. 3. There is a need for additional costs related to making a prototype display.
DevOps	1. Faster application development and deployment. 2. Be more responsive and fast to market changes. 3. Advantages in terms of software delivery time and transportation costs. 4. Improve customer experience and satisfaction. 5. Simplifies collaboration as all tools are placed in the cloud for customer access.	1. Professionals in the field of DevOps yet adequate. 2. Cost management infrastructure in relative DevOps methods is high. 3. Lack of understanding of DevOps methods can cause problem in integration project continuous automation.
Kanban	1. Kanban's are very simple and easy method understood so that make it easy management company. 2. Kanban method is spot on time or Just in Time (JIT). 3. The Kanban method is a very responsive system and does not cause slowness or delay.	1. Kanban cannot be used as an independent tool. This method is not quite suitable for a single application but can be combined with other methods such as Scrum, JIT, and others. 2. As assignments constantly move between columns of the kanban board, a certain predictive time for the completion of a task or activity becomes difficult. 3. Kanban is not suitable for dynamic environments. 4. Kanban will be very difficult to implement if there are too many related joint activities or tasks in a system.
Rational Unified Process (RUP)	1. RUP is a complete methodology with all easily available documentation. 2. RUP is publicly published, distributed, and supported. 3. Training is available where RUP can guide users through the process with step-by-step tutorials. Many institutions also offer training courses. 4. Requirements Change where proactive resolution of changing client conditions and associated risks. 5. Reduce integration time and effort. 6. The reuse rate becomes higher.	1. The process is too complex, too difficult to learn, and too difficult to apply properly. 2. Integrated process does not capture the sociological aspects of software development and details about how the software develops step by step.
3D Pipeline Production	1. Method This possible 3D animation is shown in a manner realistic, dynamic, and detailed. 2. Method This support technology modern 3D animation possible movement displayed with fluid and very detailed. 3. At the realistic and dynamic rendering stage, 3D animation also has Power strong visual appeal. They have shown for interesting more Lots attention than animation similar in 2D.	3D animation uses more complex techniques than 2D animation. This means it generally takes more time to develop 3d animations. It also requires more expensive software to create 3D animations.

The method shown in Fitzgerald's research [1] is a common method in software engineering. This study tries to explore AR as its focus so that a theory is needed for continuous software engineering for AR considering that AR is not just an ordinary application but there are certain 3D elements and components that must be met so that it can be called AR.

III. Trends Landscape of Software Engineering

The Agile method at its inception was focused on the software development function and was considered only suitable for small businesses or only for small teams [17]. But over time there has been a change with the emergence of many studies which identify that the concept of agile has entered the scale of large companies. The seven general principles of common practice for agile methods are Scrum, XP, DSDM [18]. These seven general principles address the agile approach in several dimensions. In research conducted by Leffingwell [19] said that the experience of organizations adopting agile methods in companies uses agile scale frameworks to explain practices and activities, roles, and artifacts. Basically, the organization must be responsive in responding to environmental changes that occur [20]. The DevOps method emerged because there was a gap in the development and operations functions in the company for large software scales because the bigger the team, the higher the responsibility. Automation, DevOps relies on full automation of build, deploy and test to achieve short lead times, and consequently get rapid feedback from users. Sharing happens on many levels, from sharing knowledge, tools, and infrastructure, celebrating successful releases to bringing development and operations teams closer together.

There are several tools that can be used to practice continuous software engineering, including:

- Jenkins

Jenkins is an open-source tool that is often used especially in the continuous integration/continuous deployment (CI/CD) stage. Jenkins makes it easy to automate the integration and code testing process through features provided to speed up the delivery process introducing automation [17].

- GitLab

GitLab is a complete software development platform that provides a CI/CD pipeline to integrate, test, and deploy code automatically [18].

- Travis CI

Travis CI is a CI/CD tool that provides integration with Git and GitHub repositories resulting in fast integration with various programming languages as well as a flexible testing environment [19].

- Circle CI

Circle CI is a cloud-based tool that provides integrated Git repository features that enable automated testing and deployment and supports multiple programming languages [20].

The tools above are just a few examples while the choice of the right tool depends on the needs of the project and the preferences of the development team.

IV. Proposed Continuous Software Engineering for Augmented Reality

The stages in the software method generally consist of planning, analysis, design, programming, and maintenance [4]. This research proposes additional stages in the software development method for AR, namely continuous planning, continuous analysis, continuous design, continuous programming, continuous integration, and continuous maintenance, Table II.

TABLE II. Proposed Stages of the Software Development Method for AR

No	Stages	Activity	Step in AR
A	Sustainable planning	Continuous planning	Idea, story
		Sustainable budgeting	
B	Continuous analysis	Continuous deployment	Experimental data
		Continuous verification	
C	Sustainable Design	Continuous testing	Animatic and Design
		Continuous compliance	
		Continuous security	
		Continuous evolution	
D	Continuous programming	Continuous use	Research and Development.
		Continuous trust	
		Continuous run-time monitoring	
E	Continuous integration	Continuous release	Interconnected
		Continuous delivery	
F	Continuous Maintenance	Continuous improvement	Correction action
		Continuous innovation	
		Continuous experimentation	

A. Sustainable Planning

In the context of software development, planning is an initial idea that is episodic from the current set of problem formulations [21]. Over time, changes occur, and the business environment does require planning activities to be carried out more frequently to ensure alignment between the needs of the business context and software development. Not only the business environment but covering all areas of human life will certainly always experience changes [22]. In sustainable planning there are stages of continuous planning and sustainable budgeting.

- Continuous planning

The relationship between planning and portfolio becomes important at this stage. Planning must be iterated and released, a portfolio that includes product planning activities must be carried out [23]. However, the portfolio still has the possibility of failure even though the project is generally successful. The portfolio approach tends towards the organization. In previous research, it is known that there are still many cases where the approach to individual project management has received more attention than the portfolio approach [1].

In AR, there are stages in the ideas and stories sections that should always be refreshed so that users always get new findings in the AR applications they use. Periodically new

ideas should be rolled out and new story additions or new characters in AR can provide users with an unforgettable immersive experience [24].

- Sustainable budgeting

The budget is a financial plan that is prepared in detail and coordinated from management plans with the aim of achieving targets within a predetermined period [1]. In general, budgeting is an annual or even multi-year event that describes annual events in the 'short term' and for three years or more in the 'long term'. To overcome this, the concept of "sustainable budgeting" is needed so that budgeting becomes more flexible and makes managers more flexible in making operational decisions at their own discretion to deal with unexpected situations that cannot be predicted in the master budget plan[25]. The budgeting system must be open and transparent so that the team can manage finances independently according to a predetermined budget but not limited to the annual cycle [26].

B. Continuous Analysis

At this stage it is possible not only to measure the overall behaviour of the mechanism, but also an in-depth analysis of the software engineering being executed. Experimental data to verify the continuous analysis method is strongly recommended [21].

- Continuous deployment

Continuous deployment is the continuous delivery of code changes to applications that are released during software development. This stage must be carried out in a series of predetermined tests. After going through the testing phase, the system will confirm that the software is ready to be released to the customer and that it is in accordance with customer requirements [1].

- Continuous verification

Continuous verification is the discipline of proactive experimentation or the process of monitoring applications for post-deployment anomalies. An anomaly is any interruption in normal operation that may affect application users especially in AR, including [27]:

a) High response latency
b) Server-side error
c) Client-side error
d) Downtime of any application component
e) Unexpected scaling or failover events

The purpose of continuous verification is to collect data from implemented implementations and then analyze them through machine learning and create a basis for good implementations. The benefit of this ongoing verification is to identify that something is wrong and take corrective action—for example, reverting the app to a stable or default version. Precautions should be implemented as quickly and smoothly as possible before the customer becomes aware of the problem [1].

C. Sustainable Design

In the sustainable design for AR, design components and final appearance of the project that has been decided can be updated according to market demand. At the beginning of creating AR, designers created conceptual art using media ranging from pens, charcoal, pencils, dyes, or oil paints to using computers such as Adobe Photoshop software. There is no problem using any media if the concept to be conveyed can be understood by users. Artwork often reflects the artist who created it, so the artist's mood and instincts should be an important aspect to pay attention to. The reason why this stage must be continuous is because when making art at the beginning, it is limited by a deadline. So that continuity is needed to ensure the results are delivered in accordance with the desired initial concept and get user feedback. Therefore, updates at this stage need to be done to perfect the design [28].

- Continuous testing

Continuous testing is a software development process where applications are tested continuously throughout the software development life cycle (SDLC). The goal of continuous testing is to evaluate software quality throughout the system life cycle to provide critical feedback earlier and enable faster, higher quality delivery. The advantage of this stage is when the context is still fresh in the mind of the developer and an error occurs, it is quickly resolved before the problem becomes unexpectedly deep and widespread. The benefit to software developers is that testing can be performed effectively and helps reduce overall development time by up to 15%. This is proof that continuous testing can be used as a tool to reduce waste of waiting time [29].

- Continuous compliance

At the beginning of its implementation, Agile methods were considered only suitable for small projects and placed in a crisis context, not safety. However, in recent decades Agile methods have been successfully applied to large, distributed projects. In the research conducted by Fitzgerald [1] there was discussion about adapting the Scrum method to R-Scrum or Regulated-Scrum. This is actually a disguise from a waterfall approach to an agile approach that allows developers to complete projects faster [1].

- Continuous security

Security is a priority in all phases of software development life, even after implementation, security is an important thing that must be done. Security can finally be said to be a non-functional requirement that is often unintentionally delegated to a lower priority. Therefore it is necessary to apply an intelligent and lightweight approach to be able to identify vulnerabilities to security issues [1].

- Continuous evolution

Software evolution is fundamentally dependent on the expertise of the developer as well as the changeability inherent in the software product itself [1].

D. Continuous Programming

In continuous programming in AR, it requires continuity in the research and development (R&D). R&D is a component

that covers from pre-production to post-production in 3D animation flow [9]. For example, in the animated film Finding Nemo, a team of artists from various components had to think about how to make the water feature look as if it were real. Likewise with other objects that float in the water must really look alive. When it was first released in 2003, there were no 3D animated films depicting water because it was considered especially difficult to render efficiently. Making this film is not easy, it takes years and provides its own challenges for the R&D team. But the resulting effort is well worth it, Finding Nemo succeeds in bringing the animation to life in the expected form [30].

- Continuous use

Continuous use provides advantages in terms of time and cost because the system is not built from scratch but from a continuation of an existing system. This continuous use does not mean that it can be used automatically because there must be some initial consideration and decision to use the software. The cost of acquiring new customers is estimated to be up to ten times that of retaining existing customers [21]. Previous research also stated that developers tend to stick with the system rather than designing from scratch [1].

- Continuous trust

Continuous trust is a process that takes time so that there is confidence that the vendor will do all its expertise to meet customer expectations according to their needs. Continuous use is highly dependent on continued trust where the relationship is a complex relationship. In Hoehle et al [1] research, trust is a very important start to be achieved in a transaction. For long-term activities like Cloud remote services, AWS and others require ongoing trust. Software developers need to pay attention to the issue of trust in sustainability because even if it starts well, the trust itself can erode from the user experience both internally, such as inconveniences in certain features or external factors, such as news regarding vulnerable security issues and others [21].

- Continuous run-time monitoring

A classification scheme is provided to help understand the approach chosen by a system designer with a particular method. This stage can also be used to analyse ongoing projects and consider functional and non-functional aspects that are interesting to study in software development projects [21].

E. Continuous Integration

Continuous integration is defined as a process that is usually triggered automatically and consists of interconnected steps such as compiling code, running unit and acceptance tests, validating code coverage, checking compliance with coding standards and building deployment packages. While some form of automation is typical, the frequency of integration is also important as it needs to be regular enough to ensure rapid feedback to developers. Finally, continuous integration failures are high-profile events that may have several visible ceremonies and artefacts to help ensure that the issues causing these failures are prioritized for resolution as quickly as possible by whoever is held responsible [31].

- Continuous Release

When a component is released, its version number declaration is updated, as well as its dependency declarations, because those dependencies always refer to the component that was also released. This makes the component-based release process recursive. There are significant costs associated with this method of release. The more frequently a dependent component is released, the more frequently a component that depends on it must be released to take advantage of the additional quality functionality it contains. Furthermore, with each dependency release, all components that use it should be tested for integration, before they can be released on their own [21].

- Continuous delivery

Continuous delivery is a prerequisite if developers are going for continuous adoption, but not necessarily the other way around. Continuous delivery focuses on being able to deploy software to multiple environments but not necessarily to customers. As opposed to continuous deployment which is obligatory to release valid software to its users [23].

F. Continuous Maintenance

Continuous maintenance in AR is more focused on maintaining the long-term sustainability of the software both during development and after production. This stage has a series of other advanced stages, namely continuous improvement, continuous innovation, and continuous experimentation.

- Continuous improvement

Continuous improvement activities are efforts to add customer value through reactive initiatives from software developers. Therefore, innovation is needed as a proactive strategy to emphasize customer satisfaction [1].

- Continuous innovation

Innovation in a business context is new ideas that are transformed through processes to create business value for customers. An example of this innovation is discovery plus exploitation[31]. Research on software engineering has conducted a lot of research on innovation and these keywords are the most searched for by software developers and the business community, especially regarding open innovation. The concept of Lean startups is an example of continuous innovation. Testing is done by beta testing which is used to get customer feedback before the official release of a software product. Continuous innovation is a continuous interaction between operations, gradual improvement, learning, and radical innovation to identify added value features aimed at combining operational and strategic effectiveness or known as exploitation and exploration [1].

- Continuous experimentation

Continuous experimentation has another benefit for teams in that it generates measurable metrics of change and progress so that team goals are clearly defined. The application of continuous experiments in several domains requires solutions and challenges that range from infrastructure challenges,

measurement challenges and social challenges. These challenges may only be relevant in one domain, but the solutions developed can be applied to other domains. This domain-specific challenge is closely related to the resulting solution [31].

In many publications on perpetual experimentation, the merits of the experiment are mentioned only as a motivation, i.e., improving product quality based on selected metrics. Further studies are needed to determine, for example, if there are more benefits, whether they apply to all companies involved in the experiment, or whether they could be obtained through other means. Another benefit is the potential use of continuous experimentation for software quality assurance. Continuous experimentation can support or even change the way quality assurance is performed for software. Software changes, for example, can only be applied if key metrics are not derived in the associated change experiment. Thus, the loss of quality can become measurable and measurable. Although some papers mention the use of continuous testing for software quality assurance [31].

V. Discussion

When working on AR projects, designers need to weigh quality against time and budget. When using today's rendering engines and shaders, 3D artists must find a happy medium between perfect looks and reasonable render times. New technologies are released every year in various forms—from computer hardware with faster speeds and data transfers, to software with advanced capabilities, and technologies that make workflows smoother. Some of the new trends being pursued by the 3D animation industry include full-body and detail motion capture, stereoscopic 3D output, point-cloud data, real-time workflow capabilities, and virtual studios. Each will provide a faster project turnaround and will allow artists to focus on the art of the project and not the technical hurdles of the production line. Continuous software engineering relies on a basic set of principles that govern every field of technology and includes modelling activities and other descriptive techniques.

VI. Conclusion

In general, the methods in software engineering include planning, analysis, design, programming, and maintenance. In this paper we present our initial findings about the continuous stages of the software method, namely continuous planning, continuous analysis, continuous design, continuous programming, continuous integration, and continuous maintenance then the continuous process of AR system development can be updated according to market demand.

For future work, tool support is available for continuous software engineering but suitable tool support for the whole concept from coding to delivery and data management for decision making needs attention in the future.

References

[1] B. Fitzgerald and K. J. Stol, “Continuous software engineering: A roadmap and agenda,” J. Syst. Softw., vol. 123, pp. 176–189, 2017, doi: 10.1016/j.jss.2015.06.063.

[2] C. Pang and A. Hindle, “Continuous maintenance,” Proc. - 2016 IEEE Int. Conf. Softw. Maint. Evol. ICSME 2016, pp. 458–462, 2017, doi: 10.1109/ICSME.2016.45.

[3] E. Klotins and T. Gorschek, “Continuous Software Engineering in the Wild,” Lect. Notes Bus. Inf. Process., vol. 439 LNBIP, pp. 3–12, 2022, doi: 10.1007/978-3-031-04115-0_1.

[4] I. Sommerville, Software Engineering (9th ed.; Boston, Ed.). Massachusetts: Pearson Education. 2011.

[5] M. Mahalakshmi and M. Sundararajan, “Traditional SDLC Vs Scrum Methodology – A Comparative Study,” Int. J. Emerg. Technol. Adv. Eng., vol. 3, no. 6, pp. 2–6, 2013.

[6] R. S. Pressman, Software Engineering: A Practitioner’s Approach, Seventh. McGraw Hill, 2010.

[7] R. Yung and C. Khoo-Lattimore, “New realities: a systematic literature review on virtual reality and augmented reality in tourism research,” Curr. Issues Tour., vol. 22, no. 17, pp. 2056–2081, 2019, doi: 10.1080/13683500.2017.1417359.

[8] A. B. Craig, Understanding Augmented Reality: Concepts and Application. Elsevier Science, 2013.

[9] A. Beane, 3D Animation Essentials. John Wiley & Sons, 2012. Accessed: Nov. 23, 2021. [Online]. Available: https://books.google.com/books/about/3D_Animation_Essentials.html?id=62FrKLO2M3AC

[10] H. Hwangbo, Y. S. Kim, and K. J. Cha, “Use of the Smart Store for Persuasive Marketing and Immersive Customer Experiences: A Case Study of Korean Apparel Enterprise,” Mob. Inf. Syst., vol. 2017, 2017, doi: 10.1155/2017/4738340.

[11] E. Klotins and E. Peretz-Andersson, “The unified perspective of digital transformation and continuous software engineering,” Proc. - 5th Int. Work. Software-Intensive Bus. Towar. Sustain. Softw. Business, IWSiB 2022, pp. 75–82, 2022, doi: 10.1145/3524614.3528626.

[12] Y. Dittrich, “What does it mean to use a method? Towards a practice theory for software engineering,” Inf. Softw. Technol., vol. 70, pp. 220–231, Feb. 2016, doi: 10.1016/J.INFSOF.2015.07.001.

[13] O. J. Okesola, A. A. Adebiyi, A. A. Owoade, O. Adeaga, O. Adeyemi, and I. Odun-Ayo, Software Requirement in Iterative SDLC Model, vol. 1224 AISC, no. August. Springer International Publishing, 2020. doi: 10.1007/978-3-030-51965-0_2.

[14] A. Adel and B. Abdullah, “A Comparison Between Three SDLC Models Waterfall Model, Spiral Model, and Incremental/Iterative Model,” IJCSI Int. J. Comput. Sci. Issues, vol. 12, no. 1, pp. 106–111, 2015, [Online]. Available: https://www.academia.edu/10793943/A_Comparison_Between_Three_SDLC_Models_Waterfall_Model_Spiral_Model_and_Incremental_Iterative_Model

[15] A. Mishra and Z. Otaiwi, “DevOps and software quality: A systematic mapping,” Comput. Sci. Rev., vol. 38, p. 100308, 2020, doi: 10.1016/j.cosrev.2020.100308.

[16] J. Westenberger, K. Schuler, and D. Schlegel, “Failure of AI projects: Understanding the critical factors,” Procedia Comput. Sci., vol. 196, no. 2021, pp. 69–76, 2021, doi: 10.1016/j.procs.2021.11.074.

[17] D. Yang et al., “DevOps in practice for education management information system at ECNU,” Procedia Comput. Sci., vol. 176, pp. 1382–1391, 2020, doi: 10.1016/j.procs.2020.09.148.

[18] M. S. Arefeen and M. Schiller, “Continuous Integration Using Gitlab,” Undergrad. Res. Nat. Clin. Sci. Technol. J., vol. 3, no. 8, pp. 1–6, 2019, doi: 10.26685/urncst.152.

[19] K. Gallaba, C. Macho, M. Pinzger, and S. McIntosh, “Noise and Heterogeneity in Historical Build Data,” Proc. 2018 33Rd Ieee/Acm Int. Conf. onAutomted Softw. Eng. (Ase’ 18), pp. 87–97, 2018.

[20] V. Sochat, “Containershare: Open Source Registry to build, test, deploy with CircleCI,” J. Open Source Softw., vol. 3, no. 28, p. 878, 2018, doi: 10.21105/joss.00878.

[21] D. Ameller, C. Farre, X. Franch, D. Valerio, and A. Cassarino, “Towards continuous software release planning,” SANER 2017 - 24th IEEE Int. Conf. Softw. Anal. Evol. Reengineering, pp. 402–406, 2017, doi: 10.1109/SANER.2017.7884642.

[22] R. Lozano, M. Y. Merrill, K. Sammalisto, K. Ceulemans, and F. J. Lozano, "Connecting competences and pedagogical approaches for sustainable development in higher education: A literature review and framework proposal," Sustain., vol. 9, no. 10, pp. 1–15, 2017, doi: 10.3390/su9101889.

[23] J. Bosch, Continuous software engineering, vol. 9783319112. 2014. doi: 10.1007/978-3-319-11283-1.

[24] D. Amin and S. Govilkar, "Comparative Study of Augmented Reality Sdk's," Int. J. Comput. Sci. Appl., vol. 5, no. 1, pp. 11–26, 2015, doi: 10.5121/ijcsa.2015.5102.

[25] F. Boer, Harry; Gertsen, "From continuous improvement to continuous innovation : a (retro)(per) spective Harry Boer and Frank Gertsen *," Int. J. Technol. Manag., vol. 26, no. 8, pp. 805–827, 2003.

[26] R. E. Cole, "From Continuous Improvement to Continuous Innovation," Qual. Manag. J., vol. 8, no. 4, pp. 7–21, 2001, doi: 10.1080/10686967.2001.11918977.

[27] M. Gupta, A. Mandal, G. Dasgupta, and A. Serebrenik, "Runtime monitoring in continuous deployment by differencing execution behavior model," Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics), vol. 11236 LNCS, no. October 2020, pp. 812–827, 2018, doi: 10.1007/978-3-030-03596-9_58.

[28] S. C. Yuen and E. Johnson, "AR-an-overview-five-directions-for-AR-in-ed.pdf," vol. 4, pp. 119–140, 2011, [Online]. Available: http://austarlabs.com.au/wp-content/uploads/2014/01/AR-an-overview-five-directions-for-AR-in-ed.pdf

[29] F. Auer, R. Ros, L. Kaltenbrunner, P. Runeson, and M. Felderer, "Controlled experimentation in continuous experimentation: Knowledge and challenges," Inf. Softw. Technol., vol. 134, no. February, 2021, doi: 10.1016/j.infsof.2021.106551.

[30] S. A. H. Morales, L. Andrade-Arenas, A. Delgado, and E. L. Huamanı, "Augmented Reality: Prototype for the Teaching-Learning Process in Peru," Int. J. Adv. Comput. Sci. Appl., vol. 13, no. 1, pp. 806–815, 2022, doi: 10.14569/IJACSA.2022.0130194.

[31] B. Fitzgerald and K. J. Stol, "Continuous software engineering and beyond: Trends and challenges," 1st Int. Work. Rapid Contin. Softw. Eng. RCoSE 2014 - Proc., no. May, pp. 1–9, 2014, doi: 10.1145/2593812.2593813.