Exploring Effectiveness in Software Development: A Comparative Review of System Analysis and Design Methodologies

Edward Matthew T. Sanmocte* and Jefferson A. Costales

College of Computing and Information Technologies, National University, Philippines
Email: etsanmocte@national-u.edu.ph (E.M.T.S.); jacostales@national-u.edu.ph (J.A.C.)
*Corresponding author
Manuscript received June 18, 2024; revised August 9, 2024; accepted October 9, 2024; published March 6, 2025

***Abstract*—The effectiveness of system analysis and design methodologies plays a pivotal role in the success of software development projects. This research conducts a comparative analysis of various methodologies, including the Waterfall Model, Spiral Model, Prototyping Model, Iterative Model, Unified Process, Object-Oriented Analysis and Design, Joint Application Design, Computer Aided Software Engineering Tools, Rapid Application Development / Rapid Systems Development, Feature- Driven Development, Extreme Programming, Agile Method, and DevOps, to explore their respective strengths and considerations. The study examines key aspects such as flexibility, documentation, risk management, complexity, and suitability across these methodologies. Findings reveal that each methodology offers unique advantages and challenges, influencing their applicability in different project contexts. The research underscores the importance of aligning methodology choices with project-specific requirements, organizational culture, and environmental factors to optimize software development outcomes. Moreover, it suggests that hybrid approaches, combining elements from multiple methodologies, may offer a balanced approach to address diverse project needs and maximize effectiveness. By providing insights into the comparative effectiveness of system analysis and design methodologies, this research contributes to informed decision-making and improved project success in software development endeavors.**

***Keywords*—Systems Analysis and Design Methodologies (SADM), Life Cycle / Waterfall Model, Spiral Model, Agile Model, Prototyping Model, Extreme Programming Model (XPM), Unified Process Model (UPM), Iterative Model, Object-Oriented Analysis and Design (OOAD), Computer Aided Software Engineering (CASE), Joint Application Design (JAD), Rapid Application Development (RAD), Rapid Systems Development (RSD), Extreme Programming (XP), DevOps**

I. Introduction

A. Research Background

In today's software development, the selection and application of System Analysis and Design (SAD) methodologies are crucial to the success of software development projects. These methodologies provide structured frameworks for understanding, designing, and implementing systems that align with organizational goals and user needs. Properly designed systems streamline processes and optimize workflows, in- creasing efficiency and productivity by automating repetitive tasks and minimizing human error. Additionally, effective SAD methodologies focus on resource optimization and operational efficiency, which can result in significant cost savings by identifying and eliminating waste. In an environment where technological advancements are constant, systems must be adaptable to remain relevant, and SAD methodologies ensure flexibility, scalability, and the integration of new technologies, fostering innovation and maintaining a competitive edge. Moreover, SAD provides a systematic approach to problem-solving, allowing organizations to address issues sustainably rather than relying on ad-hoc fixes, ensuring that solutions are durable and preventing problems' recurrence. Involving end-users in the design process is also essential for tailoring systems to their needs and preferences, leading to higher user satisfaction, increased adoption, improved usability, and a more positive overall experience [1, 2].

B. Research Objective

The research aims to conduct a comprehensive comparative study of these methodologies such as the Waterfall Model, Spiral Model, System Prototyping, Iterative Model, Unified Process, CASE Tools, Prototyping, Rapid Application Development (RAD), Joint Application Design (JAD), Object-Oriented Methodology, Feature-Driven Development (FDD), Extreme Programming (XP), Agile Method and DevOps approaches. By identifying their fundamental principles, characteristics, and practical implications, the study aims to provide valuable insights for selecting the most suitable methodology for various development scenarios.

C. Significance of the Study

The study seeks to provide insights into the effective-ness, adaptability, and suitability in different project contexts. Through this analysis, project leaders and teams can make an informed decision to enhance the project outcomes and align the methodology choice with the organizational goals and project requirements.

D. Research Method

This study employs a systematic literature review method- ology to conduct a comprehensive comparative analysis of various software development methodologies. The review aims to evaluate and synthesize existing research to uncover the fundamental principles, characteristics, and practical implications of methodologies including Waterfall, Spiral, System Prototyping, Iterative Model, Unified Process, CASE Tools, Prototyping, Rapid Application Development (RAD), Joint Application Design (JAD), Object-Oriented Methodology, Feature-Driven Development (FDD), Extreme Programming (XP), Agile

Method, and DevOps.

II. Related Literature Review

A. System Analysis and Design

Systems Analysis and Design (SAD) is a comprehensive methodology for developing high-quality information systems that integrate technology, people, and data to meet business needs [3]. The SAD process typically involves multiple stages, including planning, implementation, testing, documentation, deployment, and maintenance [4]. Well-designed input forms should be easy to fill out, meet their intended purpose, ensure accurate completion, and maintain visual appeal.

System Analysis and Design (SAD) methods are vital because they offer a methodical way to analyze user requirements, ensure effective use of resources, control risks, maintain quality standards, allow for flexibility and scalability, promote stakeholder collaboration and communication, and ultimately produce solutions that either exceed or meet customer expectations. By adhering to SAD methodologies, software systems can be developed by organizations that efficiently cater to user needs, adjust to changing requirements, and enhance business success via their functionality, reliability, and usability.

B. Lifecycle/Waterfall Model

The waterfall model is categorized as the traditional Software Development Lifecycle (SDLC) and is known for its linear fashion phase development. The waterfall model comprises six phases which are analysis, design, development, testing, implementation, and maintenance [5, 6]. Fig. 1 illustrates the waterfall model.

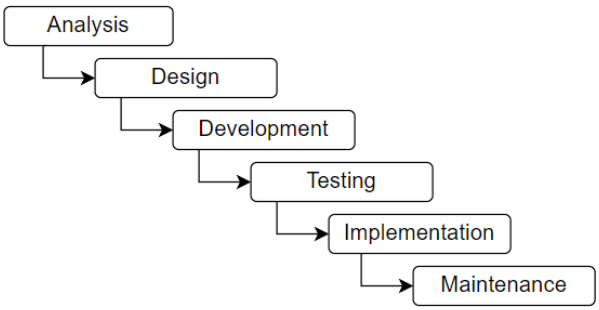

Fig. 1. Waterfall Model.

Thus, this model is favored by many of the software engineering teams because of its ease of management, well-defined deliverables and milestones established before project initiation, thorough construction of project initiation and planning, and clear outline of all phases and activities.

C. Spiral Model

The spiral model, a software development approach combining elements of waterfall and prototyping models, and it has gained attention in various fields. It offers advantages such as systematic evaluation, sequential execution, and focused risk analysis at each stage. The model's flexibility allows for adaptation to different project types, including large-scale agent-based modeling. It can support interdisciplinary teams and accommodate evolving requirements during development [7]. Fig. 2 illustrates the Spiral Model.

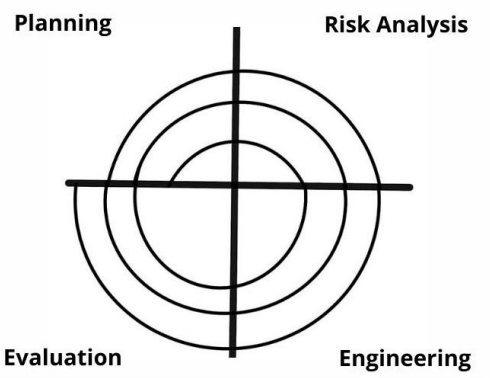

Fig. 2. Spiral Model.

This model is characterized by its iterative cycles consisting of Planning, Risk Analysis, Engineering, and Evaluation phases. Project objectives, scope, and timeline are defined in the Planning phase. The Risk Analysis phase identifies potential risks and develops mitigation strategies. The Engineering phase involves design, coding, and testing. Finally, the Evaluation phase reviews outcomes and gathers feedback for improvements. This iterative approach allows flexibility, effective risk management, and continuous refinement, making it suitable for projects with evolving requirements or unexpected changes.

D. Prototyping Model

The prototyping model involves rapidly developing and testing working models through iterative processes [8]. It can analyze functional and non-functional requirements, create system designs, and develop user interfaces [9]. The model typically consists of three stages: listening to customers, building and refining mockups, and viewing and testing mockups [8].

The Prototype Model involves creating a basic version of a system, called a prototype, to gather feedback and refine requirements iteratively. It starts with gathering initial requirements, followed by outlining a preliminary system design. A working prototype is then developed for user interaction. Users test this prototype in the Customer Evaluation phase to ensure it meets their needs. Once satisfied, the system proceeds to full development, rigorous testing, and maintenance to ensure ongoing functionality and updates. Fig. 3 illustrates Prototyping Model.

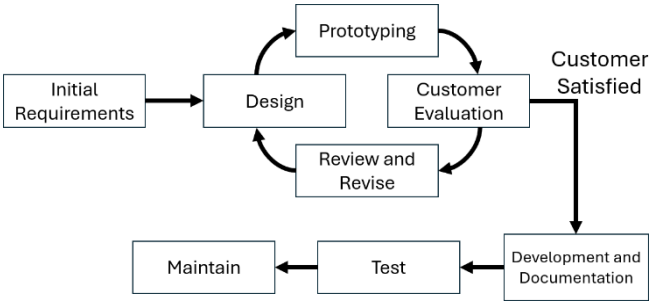

Fig. 3. Prototyping Model.

E. Iterative Model

Iterative approaches in system analysis and design offer significant benefits across various domains. While various soft- ware development lifecycle models exist, including iterative models, comparative analysis of these methodologies is crucial for organizations transitioning from

manual to automated systems [10]. These studies collectively demonstrate the value of iterative approaches in optimizing system performance, enhancing safety analysis, improving human-machine interactions, and guiding software development processes. Iterative methodologies prove to be versatile and effective across diverse engineering applications. Fig. 4 illustrates the Iterative Model.

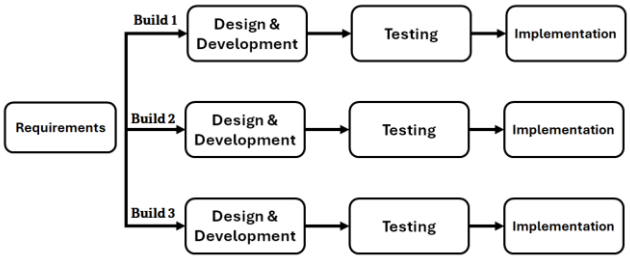

Fig. 4. Iterative Model.

The Iterative Model involves repetitive development cycles, including Requirement Analysis, Design and Development, Testing, and Implementation. Initially, developers assess client needs and system specifications. The system is then built in increments, with each iteration focusing on implementing and testing specific features. After testing and verification, features are integrated into the existing system. This approach allows for continuous refinement and enhancement, resulting in a robust, high-quality system that closely aligns with client needs.

F. Unified Process

The Unified Process (UP) is a widely used system analysis and design approach, incorporating Unified Modeling Language (UML) as a key tool. UML enhances communication and documentation in software development processes. It is particularly useful for object-oriented programming, providing standardized visual models for specifying, describing, constructing, and documenting software systems [11]. Fig. 5 illustrates the Unified Process Model.

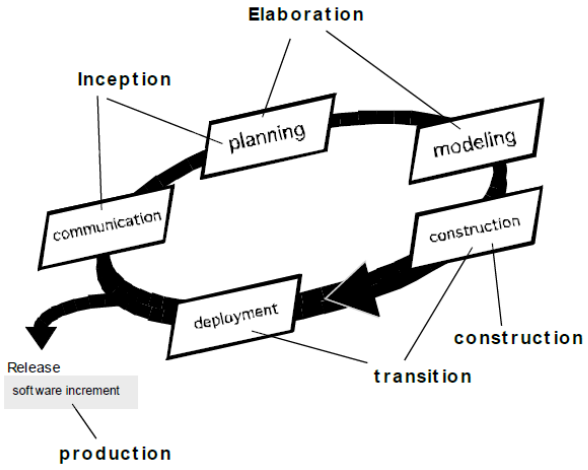

Fig. 5. Unified Process Model.

The Unified Process (UP) Model consists of iterative phases: Inception (establishing project requirements and planning), Elaboration (refining requirements and system architecture), Construction (actual system development), Transition (smooth deployment into production), and Production (ongoing support and enhancement). Each phase includes planning and modeling, with a focus on collaboration, communication, and continuous feedback. This ensures the delivered software closely aligns with client needs and evolves to meet changing requirements.

G. Object-Oriented Analysis and Design

Object-Oriented Analysis and Design (OOAD) is a method- ology used to analyze and design information systems from the perspective of classes and objects. It has been applied in various contexts, including warehouse management systems, university class rescheduling applications, and internet interference complaint resolution systems. OOAD offers a new approach to problem-solving by creating models based on real-world concepts, with objects as the foundation. This methodology employs various diagrams such as use case, sequence, and activity diagrams to analyze system requirements and design. The implementation of OOAD has shown effectiveness in addressing challenges in different domains, from improving company performance in inventory management to enhancing complaint resolution processes in government agencies. Overall, OOAD proves to be a versatile and efficient approach for system analysis and design across diverse applications [12, 13].

This model, based on object-oriented programming principles, involves several interconnected phases for systematic software development. It begins with formulating the problem, where project requirements and objectives are documented. Next is the Object-Oriented Analysis, identifying objects and their interactions to model the system's behavior and structure. This is followed by the Object-Oriented Design, defining system architecture and relationships between objects, emphasizing modularity and encapsulation. A Reusability Survey identifies reusable components to expedite development. If available, these components are used in Construction and Testing. If not, new components are developed and added to a reusable com- ponents library. After development, the application is tested, constructed, and installed. If client requirements are unmet, the process iterates back to Object-Oriented Analysis for further refinement. This approach ensures the software evolves iteratively, meeting client needs while promoting reusability and maintainability. Fig. 6 illustrates the OOAD Model.

H. Joint Application Design (JAD)

Joint Application Design (JAD) is a collaborative approach to system development that actively involves stakeholders from the outset. One study applied JAD in developing a website- based women's clothing sales information system, resulting in improved system quality and solutions better aligned with user expectations and business needs [14]. The method was integrated throughout the development stages, including needs analysis, design, and implementation. This approach aims to design technologies that are both usable and useful for individuals engaged in joint activity with machines and other people.

The Joint Application Development is a collaborative soft- ware development approach where clients work

together to define requirements and design solutions through four main phases. It starts with Defining Objectives, outlining project goals to align with organizational aims and client expectations. Next, Session Preparation involves planning sessions by selecting participants, setting agendas, and preparing materials. The core phase, Session Conduct, facilitates workshops for clients to collaboratively brainstorm, discuss, and prioritize requirements and design features. Finally, Documentation captures session outcomes, including requirements, decisions, and actions, for future reference and development. This iterative approach fosters communication, consensus-building, and rapid decision-making, leading to software solutions that effectively meet client needs. Fig. 7 illustrates the JAD Model.

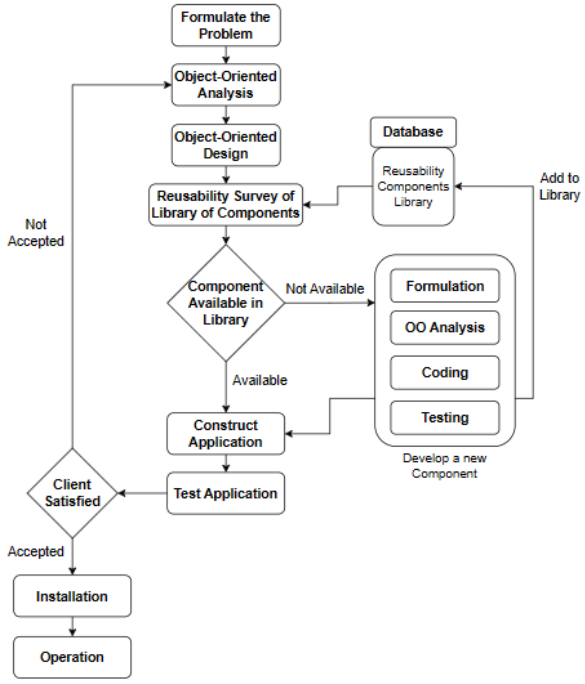

Fig. 6. The Object-Oriented Analysis and Design (OOAD) Model.

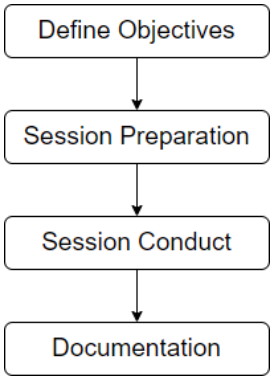

Fig. 7. Phases of Joint Application Design (JAD).

I. Computer Aided Software Engineering (CASE) Tools

Computer Aided Software Engineering (CASE) tools play a crucial role in modern software development, particularly in system analysis and design. These tools automate various processes, reducing time and cost while enhancing reliability and integrity. Modern CASE methodologies emphasize analysis and design phases, utilizing specialized tools to partially automate system development through features like code generation and database structure construction [15].

This tool comprises a suite of six interconnected components designed to streamline and enhance various aspects of the software development process. The components are: Design Editor (creates visual representations of system designs), Code Generator (automates translation of designs into executable code), Report Generator (creates comprehensive documentation), Design Translator (facilitates seamless communication between design and code), Design Analyzer (identifies potential issues in design specifications), and Program Editor (responsible for manual code modifications). These components are integrated into a centralized Project Repository, serving as the main hub for storing and managing project artifacts. This model promotes productivity, improves quality, and facilitates collaboration among team members, ultimately contributing to the successful delivery of software projects. Fig. 8 illustrates the CASE Tool Architecture.

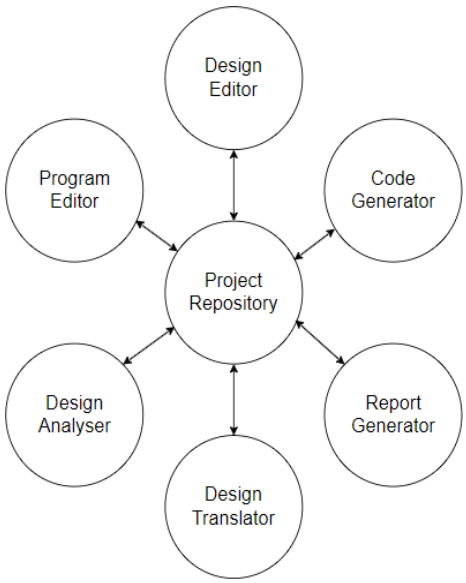

Fig. 8. CASE Tool Architect.

J. Rapid Application Development (RAD) / Rapid Systems Development (RSD)

Rapid Application Development (RAD) is a software development methodology which focuses on rapid prototyping and reducing planning time. Its implementation aims to speed up the software development process and enhance product quality. RAD is essentially an expedited version of the traditional waterfall model, prioritizing development over planning. It is well-suited for projects divisible into modules, regardless of scale, and emphasizes efficiency and effectiveness in delivering software solutions [16].

This model focuses on rapid prototyping and iterative development to speed up software delivery. It includes several interconnected phases: Business Modeling (understanding and documenting business processes and requirements), Data Modeling (designing the data structure and organization), Process Modeling (defining the system's logic and flow), Application Generation (rapidly creating software prototypes using tools and frameworks), and Testing and Turnover (ensuring the software meets quality standards and is ready for deployment). By iteratively

cycling through these phases, the model facilitates fast development, quick feedback, and continuous refinement, leading to the rapid delivery of functional software systems. Fig. 9 illustrates the RAD Model.

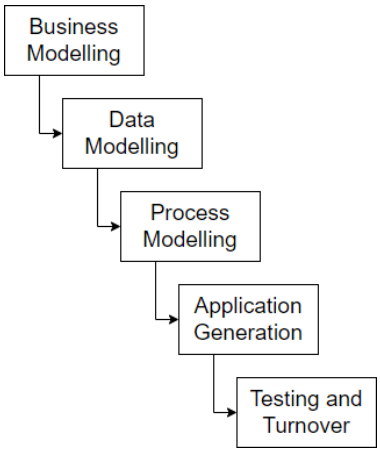

Fig. 9. RAD Model.

K. Feature-Driven Development

Feature-Driven Development (FDD) is an agile methodology that emphasizes iterative development and quality features. The method is particularly suitable for projects requiring feature parity across platforms and efficient integration of functionalities [17]. In the broader context of software engineering, feature-based analysis is emerging as a valuable approach for understanding and analyzing the machine learning development lifecycle, offering potential for improved collaboration between software engineering and machine learning experts [18].

This model is an iterative and incremental methodology focused on delivering functional features in short iterations. It involves several phases: first, developing an Overall Model of the system to identify major components and their interactions. Next is building a Feature List to prioritize and organize functionalities. Then, Planning By Feature breaks down features into smaller tasks, estimating effort, and scheduling development. Design By Feature creates detailed designs for each feature to address specific needs. Finally, Building By Feature involves implementing, testing, and integrating features incrementally. This approach enables rapid development, continuous feedback, and timely delivery of valuable features, ensuring high-quality software. Fig. 10 illustrates the FDD Model.

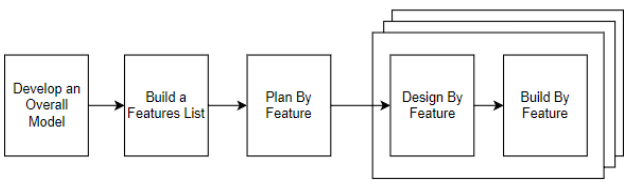

Fig. 10. FDD life cycle.

L. Extreme Programming (XP)

Extreme programming is a type of an agile model that was invented by Kent Beck in the year of 1996. He introduced his works about the Extreme programming in a much sophisticated and advanced form in the shape of a book known as “Extreme Programming Explained”. It is quite simple, uncomplicated, and more adaptable methodology of development with the capacity to oversee unclear, ambiguous, or quickly varying requirements. This model emphasizes more on the user satisfaction [19]. Fig. 11 illustrates the Extreme Programming Model.

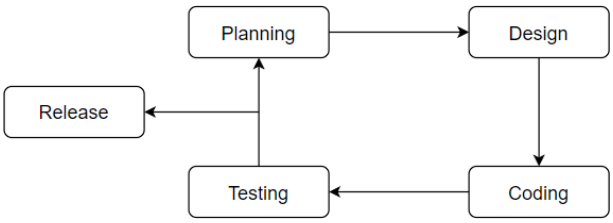

Fig. 11. Extreme Programming Model.

This agile software development model prioritizes customer satisfaction, collaboration, and adaptability to changing requirements. The process starts with Planning, where project requirements are defined and tasks scheduled. This is followed by Design, outlining the system’s architecture and details. Coding involves developers working in pairs to collaboratively write and review code. Testing ensures features meet requirements and quality standards. Notably, there is a release point between Planning and Testing for mid-cycle deployment, allowing early feedback and validation from clients. This iterative cycle of planning, coding, testing, and releasing promotes continuous improvement and refinement of the software.

M. Agile Method

Agile methodologies have become increasingly popular in software development, particularly for web-based information systems [20]. Among various Agile approaches, Scrum and Extreme Programming have emerged as the most widely used in recent years [21]. These methodologies emphasize collaboration, flexibility, and iterative development. While Agile methods offer numerous advantages, they also present challenges in terms of understanding and implementation, particularly for practitioners and students. Therefore, comprehensive tutorials and pedagogical tools are valuable for teaching and applying Agile methodologies in systems analysis and design. This model features iterative cycles of planning, execution, and feedback. It starts with Planning, defining and prioritizing project objectives and requirements. The Design phase follows, involving the development, testing, and release of software increments in short, time-boxed iterations known as sprints. Regular releases enable clients to review progress and provide feedback, which is incorporated into subsequent iterations. This approach ensures flexibility, adaptability, and continuous improvement, leading to high-quality software that meets evolving requirements and client needs. Fig. 12 illustrates the Agile Method.

N. DevOps

DevOps, a concept integrating development and operations, is gaining prominence in IT organizations due to increasing customer demands and external threats [22]. It involves cross- functional teams responsible for both software development and operations, utilizing automation to accelerate delivery processes [23]. The DevOps concept can be represented as a system of entities encompassing production, support, management, and their interrelationships [24]. However, companies adopting

DevOps often struggle with demonstrating control to auditors due to decentralized decision-making and high automation [23]. To address this, a situational control frame- work has been proposed, suggesting suitable risk mitigation practices based on an organization's risk appetite and DevOps maturity, often involving a mix of traditional manual controls and automated controls [23]. Fig. 13 illustrates the DevOps Model.

Fig. 12. Agile Model.

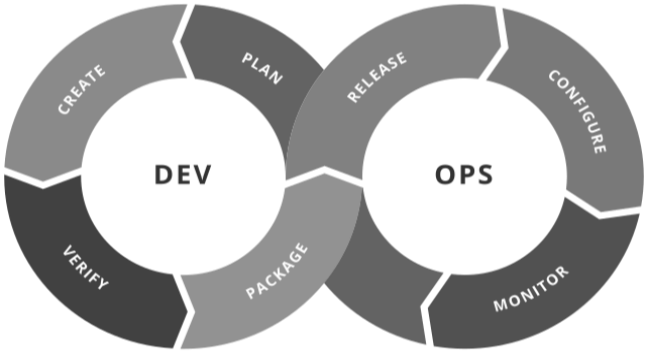

Fig. 13. DevOps Model.

This model emphasizes collaboration and automation throughout the software development lifecycle. It starts with Planning, where project goals and requirements are defined. This is followed by Code Creation, automated Testing, and Packaging of deployable units. The Releases phase involves deploying to production or staging environments with auto- mated configurations. After release, Continuous Monitoring ensures software performance, availability, and security. By integrating development and operations, this model promotes rapid delivery, increased reliability, and continuous improvement, enabling efficient and effective delivery of high-quality software.

III. Findings and Analysis

This presents a comparative analysis of various System Analysis and Design Methodologies employed in software development. Through comparative analysis approach, it seeks to unveil the differences of various methodologies that are used in software development. The Project success in software development is contingent upon an in-depth understanding of methodologies, given the diverse complexities inherent in these kinds of projects. This study aims to clarify the different aspects and operational dynamics of each methodology by delving deeply into them. Flexibility, documentation, risk management, and complexity emerge as critical considerations that influence the adaptability, guidance, resilience, and feasibility of methodologies in a variety of project settings.

When choosing the best System Analysis and Design Methodologies (SADM), key factors like flexibility, documentation, risk management, and complexity play a crucial role. Flexibility ensures that the methodology can adapt to evolving project needs and constraints, allowing for adjustments as the project progresses. Comprehensive documentation serves as a reference point for the entire team, ensuring clarity, consistency, and continuity throughout the project lifecycle. Effective risk management is essential to anticipate, mitigate, and manage potential challenges that could derail the project. Lastly, understanding and managing complexity is vital to ensure that the chosen methodology aligns with the project's scale and intricacy, avoiding unnecessary complications and ensuring successful outcomes. The purpose of this study is to provide software development practitioners and decision- makers with useful insights based on empirical research and real-world experiences, so they can make decisions that will maximize project outcomes.

A. Flexibility

In Table 1, the waterfall model exhibits a low flexibility due to its sequential nature which makes it challenging to accommodate changes once the development process has progressed. Alternatively, methodologies like Object Oriented Analysis and Design (OOAD), CASE tools, Agile, Extreme Programming (XP), DevOps and Rapid Application Development offers a high flexibility by emphasizing their iterative development which allows for any frequent changes and adaptability to evolving requirements. Whereas Prototyping and Joint Application Design also score high in flexibility as they involve quick iterations and clients involvement in the design process.

Table 1. Comparative Analysis of SADM

Comparative analysis of System Analysis and Design Methodologies				
Methodologies	**Flexibility**	**Documentation**	**Risk Management**	**Complexity**
1. Waterfall Model	Low	High	Moderate	High
2. Spiral Model	Moderate	Moderate	High	High
3. Prototyping Model	High	High	Low	Moderate
4. Iterative Model	Moderate to High	Moderate	Moderate	Moderate
5. Unified Process	Moderate to High	High	High	High
6. OOAD	High	High	Moderate	High
7. JAD	High	Moderate	Low	Moderate

8. CASE Tools	Moderate	High	Moderate	High
9. RAD	High	Low	Low	Low
10. FDD	Moderate to High	Moderate	Moderate	Moderate
11. XP	High	Low	Moderate	Moderate
12. Agile Method	High	Low to Moderate	Moderate	Moderate
13. DevOps	High	Moderate	High	High

B. Documentation

Table 1 illustrates that methodologies such as Unified Process, Waterfall Model, and Object-Oriented Analysis and Design (OOAD) prioritize comprehensive documentation to ensure a clear understanding and traceability of system requirements, design, and architecture. Likewise, Computer Aided Software Engineering (CASE) Tools uses an extensive documentation through automation and tool-assisted processes. While on the other hand, Agile-based methodologies, Extreme Programming, and Rapid Application Development tend to have lower documentation requirements, focusing more on the working software over comprehensive documentation.

C. Risk Management

In Table 1, the Spiral Model is most notable for its explicit emphasis on risk management through its iterative development cycles, by incorporating risk analysis and mitigation strategies. Agile methodologies, including Extreme Programming and DevOps, also prioritize risk management by promoting frequent feedback, adaptation, and continuous improvement. Unified Process manages risk through an iterative and incremental approach, which is accompanied by rigorous documentation and control mechanisms, whereas the Waterfall Model and Prototyping Model may have a lower risk management due to their linear or exploratory nature.

D. Complexity

In Table 1, methodologies such as Unified Process, Object- Oriented Analysis and Design (OOAD), and Computer Aided Software Engineering (CASE) Tools are suited for projects that are complex due to their emphasis on systematic analysis, design, and rigorous documentation to manage their complexity effectively. The Spiral Model, Agile methodologies, and DevOps all address complexity through iterative development, collaboration, and continuous integration practices. On the other hand, Rapid Application Development and Joint Application Design may be better suited for less complex projects due to their emphasis on quick iterations and client involvement.

IV. Conclusion

In conclusion, the comparative analysis of System Analysis and Design Methodologies in Software Development reveals that each methodology offers unique strengths and considerations across various aspects such as flexibility, documentation, risk management, and complexity.

According to the results and findings the researchers determined the suitability of these methodologies in different projects. The Waterfall Model, Unified Process, and Object- Oriented Analysis and Design are well-suited for projects with well-defined requirements and a focus on rigorous planning and control. The Spiral Model and Iterative Model are suitable for projects with evolving requirements or high uncertainty, which allows them for the continuous refinement and adaptation. The Prototyping Model, Rapid Application Development, and Feature Driven Methodologies are ideal for projects with fast-changing requirements or tight deadlines, which enables rapid prototyping, iterative development, and feature-centric delivery. The Agile Method, Extreme Programming, and DevOps are suited for dynamic environments where customer or client feedback and rapid response to change are essential, fostering teamwork, flexibility, and innovation. The Joint Application Design and Computer Aided Software Engineering Tools facilitate collaboration and efficiency in software development, they provide tools and methodologies to streamline development processes and promote stakeholder involvement in the design and development phases.

Overall, the choice of methodology depends on project-specific factors such as its requirements, constraints, and organizational culture. Organizations should carefully evaluate the strengths and considerations of each methodology to determine what is the most suitable approach for their software development projects. Additionally, hybrid approaches that combine elements from multiple methodologies may offer benefits in addressing diverse project needs and maximizing the project's success. For more in-depth probability—and simulation-based research, the reader is referred to [25, 26], and including Quality Assurance and Implementation to [27].

Conflict of Interest

The authors declare no conflict of interest.

Author Contributions

E.T.S. conducted the research and wrote the paper; J.A.C. provided valuable insights on the paper's content. All authors had approved the final version.

Acknowledgment

The authors would like to extend their gratitude to the authors of the research papers referenced in this study, whose work provided valuable insights for conducting this research.

References

[1] DOOR3. (November 2023). Systems analysis and design: Exploring modern systems. *DOOR3*. [Online]. Available: https://www.door3.com/blog/system-analysis-and-design

[2] J. Hudgens. The importance of systems analysis and design for solving real world software problems. [online]. Available: https://vtm.devcamp.com/full-stack-development-javascript-python/guide/importance-systems-analysis-and-design-for-solving-real-world-software-problems

[3] R. Hoehndor, "Design and analysis of a system," *Journal of Information Technology & Software Engineering*, p. 1, 2021.

[4] A. S. W. Jelantik, P. T. H. Permana, and N. M. Estiyanti, "Analysis and design of a point-of-sale system using agile development methods at Eka Putra Sukawati Store," *Jutisi Jurnal Ilmiah Teknik Infor- matika Dan Sistem Informasi*, vol. 10, no. 2, p. 185, Aug. 2021. doi: 10.35889/jutisi.v10i2.660

[5] N. Rofiq, A. Perdananto, and N. Jaya, "Application of the Waterfall Model in a waste bank application," *Infotech Journal of Technology Information*, vol. 7, no. 1, pp. 19–26, Jun. 2021. doi: 10.37365/jti.v7i1.102

[6] N. Yahya and S. S. Maidin, "The Waterfall Model with agile scrum as the Hybrid Agile Model for the software engineering team," in *Proc. 2022 10th International Conference on Cyber and IT Service Management (CITSM)*, Sep. 2022. doi: 10.1109/citsm56380.2022.9936036

[7] M. Malikov, F. A. Aloraini, H. Kavak, W. G. Kennedy, and A. Crooks, "Developing a large-scale agent-based model using the spiral software development process," in *Proc. 2023 Annual Modeling and Simulation Conference (ANNSIM)*, 2023, pp. 282–293.

[8] Y. Firmansyah, R. Maulana, and D. O. Hutagalung, "Implementation of the Prototyping Model in the development of a spare parts sales information system," *Jurnal Sistem Informasi Akuntansi*, vol. 2, no. 1, pp. 63–71, Mar. 2021. doi: 10.31294/justian.v2i01.366

[9] N. Lasminiasih, G. E. Saputra, N. R. B. Utomo, and N. E. Wiseno, "Using prototyping method for analysis and design of information systems for student registration in Sekolah Master," *International Journal Science and Technology*, vol. 1, no. 2, pp. 19–29, Jul. 2022. doi: 10.56127/ijst.v1i2.140

[10] H. Prabowo, F. L. Gaol, and A. N. Hidayanto, "Comparison of the system development life cycle and prototype model for software engineering," *International Journal of Emerging Technology and Advanced Engineering*, vol. 12, no. 4, pp. 155–162, 2022.

[11] A. Gadhi, R. M. Gondu, C. M. Bandaru, K. C. Reddy, and O. Abiona, "Applying UML and machine learning to enhance system analysis and design," *International Journal of Communications Network and System Sciences*, vol. 16, no. 5, pp. 67–76, Jan. 2023. doi: 10.4236/ijcns.2023.165005

[12] A. Mulyana, "Design of a Warehouse Stock Management Information System using Object-oriented analysis and design method," *Jurnal Informatika Kesatuan*, vol. 2, no. 2, pp. 239–248, Sep. 2022. doi: 10.37641/jikes.v2i2.1832

[13] I. Permatahati, R. T. R. L. Bau, H. A, and S. S. R. Abdul, "Enhancing internet interference complaint resolution: A case of Object-oriented systems approach at Kominfo Kota Gorontalo," *Jurnal Riset Sistem Dan Teknologi Informasi*, vol. 1, no. 2, pp. 12–21, Aug. 2023. doi: 10.30787/restia.v1i2.1261

[14] R. T. Aldisa, "Application of the Joint Application Design (JAD) method in developing a women's clothing sales information system website based at Aldisa Boutique', *West Science Interdisciplinary Studies*, vol. 2, no. 3, pp. 726–731, 2024.

[15] И. И. Ляшенко and Е. В. Прокопец, "On the use of modern methods of conceptual design for information systems," *Bulletin of Toraighyrov University Physics & Mathematics series*, 2023.

[16] M. A. Fauzi, H. Tribiakto, A. Moniva, F. Amir, I. K. Ilyas, and E. Utami, "Systematic literature reviews on rapid application development information system," *Bulletin of Comp. Sci. Electr. Eng.*, vol. 4, no. 1, pp. 57–64, Jun. 2023.

[17] A. R. Chrismanto, A. Wibowo, L. Chrisantyo, and M. N. A. Rini, "Implementation of feature driven development to facilitate feature quality and adaptation in the development of dutatani web and mobile portal," *Jurnal Edukasi dan Penelitian Informatika (JEPIN)*, 2022.

[18] B. C. Hu and M. Chechik, 'Towards feature-based analysis of the machine learning development lifecycle," in *Proc. 31st ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering*, San Francisco, CA, USA, 2023, pp. 2087–2091.

[19] A. Akhtar, B. Bakhtawar, and S. Akhtar, "Extreme programming vs scrum: A comparison of agile models," *International Journal of Technology, Innovation and Management*, vol. 2, no. 2, Oct. 2022. doi: 10.54489/ijtim.v2i2.77

[20] S. H. Nova, A. P. Widodo, and B. Warsito, "Analysis of Agile Methods in the development of information systems based on website: Systematic literature review," *Techno. Com.*, vol. 21, no. 1, pp. 139–148, Feb. 2022. doi: 10.33633/tc.v21i1.5659

[21] L. Trihardianingsih, M. Istighosah, A. Y. Alin, and M. R. G. Asgar, "Systematic literature review of trend and characteristic agile model," *Jurnal Teknik Informatika*, vol. 16, no. 1, pp. 45–57, 2023.

[22] R. Amaro, R. Pereira, and M. M. Da Silva, "Capabilities and metrics in DevOps: A design science study," *Information & Management*, vol. 60, no. 5, p. 103809, Jul. 2023. doi: 10.1016/j.im.2023.103809

[23] O. H. Plant, J. Van Hillegersberg, and A. Aldea, "Rethinking IT governance: Designing a framework for mitigating risk and fostering internal control in a DevOps environment," *International Journal of Accounting Information Systems*, vol. 45, p. 100560, Jun. 2022. doi: 10.1016/j.accinf.2022.100560

[24] P. Maslianko and I. Savchuk, "DevOps–concept and structural representation," *KPI Science News*, no. 4, pp. 39–51, Feb. 2022. doi: 10.20535/kpisn.2021.4.261938

[25] W. Kramer, M. Sahinoglu, D. Ang, "Increase return on investment of software development life cycle by managing the risk—A case study," *Defense AR Journal*, vol. 22, no.2, pp. 174–191, April 2015.

[26] M. Sahinoglu, S. Stockton, S. Morton, M. Eryilmaz, "Metrics to assess and manage software application security risk," in *Proc. International Conference on Security and Management (SAM), The Steering Committee of The World Congress in Computer Science, (WorldComp)*, Las Vegas, NV, USA, July 20, 2014, pp. 275–282.

[27] K. E. Kendall and J. E. Kendall, *Systems Analysis and Design*, 8th Ed., Pearson Education Inc., Prentice Hall, New Jersey, 07458, 2011.