International Journal for Multidisciplinary Research (IJFMR)
E-ISSN: 2582-2160 ● Website: www.ijfmr.com ● Email: editor@ijfmr.com

Evaluative Exploration of Comparative Analysis of Information Systems Design Methods

Semen Levin[1], Vitaly Dunaevsky[2]

[1]Professor, Department of Automated Control Systems, Tomsk State University of Control Systems and Radioelectronics
[2]Department of Automated Control Systems, Tomsk State University of Control Systems and Radioelectronics

Abstract
In today's rapidly evolving technological landscape, selecting an appropriate method for designing information systems holds paramount importance. This paper aims to ascertain the most effective design methods for various project types, underlining their significance in contemporary scenarios. The investigation fulfills its objectives through a comprehensive review, analysis, and comparison of existing design methodologies. The literature review delineates the historical evolution and classification of design methodologies, including traditional models such as the waterfall and V-model, agile methodologies like Scrum and Kanban, and modern approaches such as DevOps and continuous integration and deployment. Methodological considerations include defining evaluation criteria encompassing flexibility, development speed, cost, product quality, and user satisfaction and elucidating the data collection and analysis process. A comparative analysis evaluates methodologies based on their respective advantages, disadvantages, and real-world applications. The discussion interprets the findings, offering insights into the relative effectiveness of each methodology vis-à-vis project conditions and types.

Keywords: information system design, software development methods, Agile methods, Waterfall model, Scrum, Kanban, DevOps, Continuous integration and deployment

1. Introduction

Data The modern world, characterised by rapid technological advancements and information systems development, imposes high demands on the methods used for their design. In the era of globalisation and digital business transformation, the role of IT professionals, project managers, and business leaders in choosing the suitable design method becomes a critical factor in determining a project's success and an organisation's competitiveness. One of the essential aspects highlighting the relevance of selecting information systems design methods lies in the need to adapt to rapidly changing market requirements [1]. Modern business processes demand high flexibility and the ability to respond promptly to changes. In this context, traditional methods such as the waterfall model often fail to provide the required level of adaptability due to their rigid structure and sequential approach to development [2].

Conversely, agile methodologies like Agile and Scrum offer more dynamic approaches, allowing for quick changes and adjustments throughout development [3]. Another crucial aspect is risk management. Effective risk management becomes a key element of successful information systems design in an environment of high uncertainty and constantly evolving market demands. Agile development methods,

based on an iterative approach, enable identifying and mitigating risks at early project stages. It significantly reduces the likelihood of critical errors and ensures sustainable project development.

Meanwhile, traditional methods are often less effective in risk management due to their sequential nature and limited ability to incorporate changes after completing individual stages. Considerable attention should also be given to the factor of time. In a highly competitive environment, the speed of product deployment becomes a decisive factor. Traditional design methods, such as the V-model [4], require the completion of one stage before proceeding to the next, often leading to prolonged development timelines. Agile methodologies enable concurrent execution of multiple stages, significantly accelerating the process of creating an information system and allowing for faster adaptation to changing user and market requirements.

The quality of the end product also depends on the chosen design approach. Traditional methods often suffer from inadequate feedback in the early stages of development, which can result in creating a product that does not fully meet user requirements. In contrast, agile methodologies involve continuous interaction with the customer and regular checks of intermediate results, enabling timely adjustments and improvements [5]. This not only enhances the quality of the final product but also increases user satisfaction, a key factor in the success of any project.

Equally important are the costs. Projects implemented using agile methodologies often have a more transparent resource allocation process, allowing for cost optimisation and avoiding unnecessary expenses [6]. Traditional design methods, especially in large and complex projects, can lead to significant overspending due to the need for late-stage changes and insufficient flexibility in resource management.

The importance of integrating new technologies and approaches into designing information systems is growing in digital business transformation. Technologies such as artificial intelligence, machine learning, and the Internet of Things require developers to utilise modern and flexible methodologies capable of quickly adapting to new requirements and capabilities. Agile development methods, such as DevOps, ensure continuous integration and deployment, enabling the most efficient utilisation of modern technologies and rapid response to changes in the technological landscape [7].

Modern information systems are becoming increasingly complex and multi-layered, requiring developers to use methods that facilitate high levels of interaction between different system components [8]. Agile methodologies, such as Kanban, offer tools for effectively managing task flows and ensuring continuous interaction between various project parts. It is essential in distributed teams, where it is necessary to ensure smooth operation and coordination of project participants in different geographical locations.

The importance of choosing a method for designing information systems is also driven by requirements for security and reliability. Modern information systems must provide high data protection and resilience to cyber threats. Thanks to their iterative approach and continuous testing, Agile methodologies allow for identifying and eliminating vulnerabilities at the early stages of development, significantly enhancing the security and reliability of the final product.

2. Historical Overview: Evolution of Information System Design Methods

The evolution of information system design methods reflects the overall progress in information technology and project management. Over the past decades, design methods have undergone significant changes, adapting to new business requirements and technological innovations. Let us consider the critical stages in the development of these methods.

When information systems began to develop in the 1950s and 1960s, their design process was relatively

primitive. Projects were implemented manually, often needing a clearly defined methodology. The emphasis was primarily on hardware, while software was considered secondary. Information systems of that time were centralised, and their design was carried out at the level of large mainframes [9].

In the 1970s, as the complexity of information systems increased, there arose a need for more formalised design methods. During this period, a structured design approach was developed, including methodologies such as structured analysis and structured programming. These methods involved dividing the system into modules and using flowcharts to describe program logic. One of the critical methodologies of this period was the "cascade" (waterfall) model [10].

The cascade model, proposed in 1970 by Winston Royce, was one of the first formalised software development methodologies. This model divides the development process into clearly defined stages: requirements analysis, design, implementation, testing, integration, and maintenance. Each stage must be fully completed before proceeding to the next. The main advantage of the cascade model was its structured nature and clear documentation, providing predictability and control over the development process. However, the main drawback of this model was its inflexibility and inability to make changes in the later stages of the project [11].

In the 1980s, there was a need for more flexible design methods. Software lifecycle models such as the V-model and the spiral model were developed during this time. The V-model represented an enhanced version of the cascade model with a stronger emphasis on validation and verification at each stage of development.

Barry Boehm's 1986 proposal of the spiral model was one of the first examples of an evolutionary approach to software development. This model combined elements of the cascade model and the iterative process, allowing developers to assess and minimise risks at each spiral turn. The spiral model involved four main phases: planning, risk analysis, engineering, and evaluation. After each iteration, results were reviewed, and necessary adjustments were made [12].

With the development of the Internet and the growing business need for rapid and flexible responses to changes, agile methodologies became popular in the 1990s. The main principle of these methodologies is iterative development with constant feedback from users. One of the first examples of such methodologies was the Rapid Application Development (RAD) methodology, proposed by James Martin in 1991 [13].

RAD was a method for rapid application development that used an iterative approach and component reuse. The main stages of RAD include requirement planning, design, construction, and deployment. Thanks to the iterative approach and focus on user feedback, RAD significantly reduced development time and improved the quality of the final product.

In the mid-1990s, methodologies such as Scrum and Extreme Programming (XP) emerged, becoming foundational in the Agile movement. Scrum, developed by Jeff Sutherland and Ken Schwaber, proposed project management through a series of short iterations called sprints. This methodology emphasised self-managed teams and constant adaptation to changing requirements [14].

XP, proposed by Kent Beck, focused on practices such as pair programming, continuous testing, and rapid integration. These practices allowed teams to adapt to changes and improve software quality quickly.

Adopting the Agile Manifesto in 2001 firmly established agile methodologies in the software development industry. The Agile Manifesto proclaimed values and principles aimed at increasing the flexibility and adaptability of development processes. Simultaneously, the concept of DevOps began to take shape, merging development and operations to enhance both the quality and speed of software delivery [15]. DevOps prioritizes continuous integration and deployment (CI/CD), automating processes and fostering

close cooperation between developers and operators. Its core principles are collaboration, automation, and monitoring, which enable swift responses to changes and enhance system performance and quality [16].
Today, methods for designing information systems continue to advance, incorporating new technologies such as artificial intelligence and machine learning. These technologies necessitate the integration of complex algorithms and substantial volumes of data, creating new demands on design methods. Modern approaches include data-driven methodologies, which facilitate decision-making based on data analysis and machine learning.

3. Description and Classification of Design Methods

Today, various methods are used in information system design, classified into traditional, agile, and modern approaches. Each method has unique traits, benefits, and drawbacks, allowing for selecting the most appropriate option based on specific project conditions and needs.

Traditional Methods

The ***Waterfall Model***, also known as the sequential or linear model, is one of the most well-known traditional methods for designing information systems. This model divides the development process into sequential stages, each of which must be fully completed before the next one begins. The main stages include requirements analysis, design, implementation, testing, integration, and maintenance [17].
*Advantages***:** The Waterfall Model provides structure and sequence, facilitating easy project management and documentation of all development stages. This method suits projects with clearly defined requirements and a low likelihood of changes.
*Disadvantages***:** The main drawback of the Waterfall Model is its inflexibility. Changes in the later stages of development become complicated and costly. This method is not suitable for projects with a high degree of uncertainty and changing requirements.
The ***V-Model*** is an enhanced version of the Waterfall Model, focusing on testing and validation at each stage of development. In this model, the design and testing stages are arranged in the shape of a "V," reflecting the sequence of their execution [18].
*Advantages***:** The V-Model ensures early detection and correction of defects through regular testing. This method is suitable for projects requiring high reliability and software quality.
Disadvantages: Like the Waterfall Model, the V-model is not flexible and unsuitable for projects with rapidly changing requirements.

Agile Methods

Scrum is one of the most popular agile methodologies designed to manage projects in high uncertainty. In Scrum, the project is divided into a series of short iterations called sprints, which typically last one to four weeks. Each iteration includes planning, development, testing, and review [19].
Advantages: Scrum provides high adaptability and flexibility, allowing for rapid response to changes and regular feedback from the customer. Self-managed teams and clearly defined roles (Product Owner, Scrum Master, development team) promote efficient collaboration and productivity.
Disadvantages: Scrum requires a high level of discipline and team experience. With proper management and adherence to methodology principles, the project may avoid problems related to organisation and control.
Kanban is a project management method that visualises workflows and manages task flows. Its primary tool is a board with cards representing tasks that move between columns, reflecting different stages of completion [20].

Advantages: Kanban provides transparency and visibility of the workflow, making it easy to track progress and identify bottlenecks. This method is suitable for projects with a continuous flow of tasks and high variability of requirements.
Disadvantages: Kanban may need to be more structured than other agile methodologies, requiring high self-organization and discipline from the team.

Modern Approaches

DevOps is a methodology that combines development and operations processes to improve the quality and speed of software delivery. Its key principles include continuous integration (CI), continuous deployment (CD), process automation, and close collaboration between developers and operators [21].
Advantages: DevOps accelerates development and deployment while enhancing software quality and stability through automation and continuous monitoring. This approach allows for a quicker response to changes and improves coordination between various teams.
Disadvantages: Implementing DevOps necessitates substantial changes in organisational culture and processes, as well as investments in automation tools and employee training. It may encounter scalability issues and integration challenges without proper management and coordination.
The **Continuous Integration and Deployment (CI/CD)** model is a critical element of DevOps, streamlining the integration and deployment of code changes [22]. CI/CD involves the regular integration of code (Continuous Integration) and the automated testing and deployment of that code (Continuous Deployment).
Advantages: The CI/CD model supports the rapid and secure implementation of software changes, minimising the risk of errors and reducing the time required to deploy new versions. Automation in testing and deployment enhances product quality and reduces the effort needed for manual processes.
Disadvantages: Effective implementation of CI/CD demands extensive automation and infrastructure, as well as highly skilled specialists. Additionally, this method requires continuous monitoring and support to maintain its effectiveness.
Each design method has unique characteristics that make it more or less suitable for different projects. Traditional methods, such as the waterfall and V-models, offer structure and order, which is ideal for projects with well-defined requirements and minimal changes. Conversely, agile methodologies provide flexible and iterative approaches better suited for dynamic and uncertain projects. Modern methods, including DevOps and CI/CD, focus on automating and integrating development and operations, allowing quicker and higher-quality software delivery.
Ultimately, the selection of a design method should be based on the specific needs and conditions of the project, considering its scale, complexity, deadlines, and budget, as well as the level of uncertainty and variability in requirements. The contemporary world of information technology demands flexibility and adaptability, making agile methodologies and modern approaches increasingly desirable and effective for managing projects in software development.

4. Defining criteria for comparing design methods

To effectively assess information system design methods, clear criteria must be established. These criteria facilitate an objective comparison of methods, highlighting their strengths and weaknesses and aiding in selecting the most suitable approach for a specific project. This section delves into the key assessment criteria in detail.

Flexibility and Adaptability

The importance of a design method's flexibility and adaptability is paramount in modern environments where requirements and external factors can change rapidly. This criterion examines how swiftly and efficiently a method can respond to such changes. Flexibility allows for modifications in the development process at any stage without causing significant delays or costs. Adaptability ensures the method remains relevant in evolving project conditions, reducing risks and increasing the likelihood of project success.

Risk Management

Risk management is crucial in any project. It involves identifying, assessing, and mitigating potential issues and threats throughout various development stages. This criterion evaluates how well a method can predict potential risks and devise strategies to mitigate them. Effective risk management prevents serious errors and failures, which is especially vital for complex and large-scale projects [23].

Development Speed

Development speed pertains to the time required to complete a product from initial design to final deployment. This criterion is critical as speed directly affects project completion deadlines and overall success. In highly competitive and fast-changing markets, the ability to bring a product to market quickly can be a decisive factor. Assessing methods based on this criterion helps determine how quickly each method can achieve the ultimate goal [24].

Costs

The costs associated with designing and developing a system include expenditures on resources, time, and team efforts. This criterion evaluates the economic efficiency of the method, considering both initial costs and potential expenses for error correction and changes. Costs significantly influence the project budget and profitability, making them a crucial factor in method selection [25].

Integration of New Technologies

This criterion assesses a method's ability to incorporate and utilise modern technological solutions in the development process. Effective integration of new tools, platforms, and technologies is vital in the face of rapid technological advancement. The ability to integrate new technologies swiftly and seamlessly ensures that the product remains relevant and competitive [26].

Simplicity and Clarity of Project Management

This criterion evaluates the ease and transparency of the project management process. It includes task allocation, team coordination, and monitoring of work progress. Ensuring efficient project execution, minimising misunderstandings, and enhancing communication within the team are critical aspects. A straightforward management process reduces the likelihood of organisational and coordination issues [27].

Scalability

Scalability measures a method's ability to adapt to projects of different scales, from small to large and complex. This criterion assesses how well the method suits various project sizes and complexities. A scalable method enables efficient resource and process management in both small and large projects, making it a versatile tool [28].

Documentation

This criterion refers to the completeness and quality of documentation produced during design and development. It evaluates how well the method ensures detailed and accurate documentation of all project stages. Good documentation is essential for project maintenance and evolution and for knowledge transfer among team members and new hires [29].

Collaboration and Communication

This criterion assesses the effectiveness of interaction and information exchange among team members and other stakeholders. It evaluates how well the method improves team communication, collaboration, and knowledge sharing. Effective collaboration enhances product quality, accelerates development, and reduces misunderstandings [30].

Process Automation

Process automation evaluates a method's capability to automate various aspects of software development and deployment. This criterion assesses how well the method supports the use of automated tools and processes to improve development efficiency and quality. Automation reduces the likelihood of errors, speeds up the process, and boosts overall team productivity [31].

Compatibility with Organisational Structure

This criterion assesses how well the method aligns with the existing structure and processes of the organisation. Good compatibility minimises resistance to change and ensures a smooth transition to the new method, enhancing team efficiency [32].

Feedback

This criterion evaluates the ability to receive regular and constructive feedback from customers and users at various project stages. It assesses how well the method incorporates customers' and users' opinions for necessary improvements and adjustments. Regular feedback helps identify and address issues promptly, improving product quality and user satisfaction [33].

Support and Training

This criterion assesses the availability of resources for training the team and supporting the implementation of the method within the organisation. It evaluates how well the method provides training and support for team members, crucial for transitioning to and effectively using a new method. Good support and training facilitate quick adaptation to the new method, reduce errors, and enhance overall team productivity [34].

5. Research Methodology

In this study, we applied a detailed approach to collecting and analysing data, thoroughly evaluating the methods of designing information systems. The methodology consisted of multiple stages, each focused on gathering the most objective and comprehensive information about the considered methods.

The initial stage involved an in-depth literature review. We analysed scientific articles, books, technical reports, and conference materials related to information system design methods. The goal was to identify existing methods, their features, and their application in various contexts. This review established a knowledge base and identified key criteria for further comparison of the methods.

In the second stage, primary data was collected collaboratively. This process included surveys and interviews with experts in information system design. Specialists from various industries participated in the surveys, providing a wide range of opinions and experiences. The interviews aimed to gather detailed information about the practical use of the methods, including their strengths and weaknesses.

After data collection, a comprehensive analysis was conducted. Various statistical and qualitative analysis techniques were used to process the information. Statistical analysis included calculating averages, standard deviations, and other indicators for the quantitative data from the surveys. Qualitative analysis focused on examining the interviews in-depth, identifying key themes and trends.

Using the collected and analysed data, we compared information system design methods according to predefined criteria. Criteria weights were assigned in percentages based on their importance. Each method was evaluated against these criteria on a ten-point scale. The scores for each criterion were then adjusted according to their weights and summed to produce the final assessment for each development method.
A validation stage ensured the reliability and accuracy of the results. This involved additional experts reviewing the collected data and conclusions. These experts analysed the methodology, data collection process, and results to confirm their validity and reliability.

- Weights assigned to each evaluation criterion for assessing information system design methods (in percentages):
- Flexibility and adaptability: 17%
- Risk management: 15%
- Development speed: 12%
- Costs: 12%
- Integration of new technologies: 8%
- Project management simplicity and clarity: 6%
- Scalability: 6%
- Documentation: 4%
- Collaboration and communication: 4%
- Process automation: 4%
- Compatibility with organisational structure: 4%
- Feedback: 4%
- Support and training: 4%

Assessment of Information Systems Design Methods by Criteria:

- Cascade Model
- V-Model
- Scrum
- Kanban
- DevOps
- Continuous Integration and Deployment Model (CI/CD)

6. Results

The evaluation of different information systems design methods provided quantitative data to understand the effectiveness of each approach in different settings. Data for each criterion are shown in Table 1.

Table 1: Scores for Compared Criteria of Design Methods

Criterion	Cascade Model	V-Model	Scrum	Kanban	DevOps	CI/CD
Flexibility and Adaptability	2	3	9	8	8	9
Risk Management	5	7	8	7	9	9
Development Speed	3	4	8	7	9	9
Costs	6	6	7	7	8	8
Integration of New Technologies	3	4	7	7	9	9
Project Management Simplicity and Clarity	6	7	6	8	7	7
Scalability	6	7	8	8	9	9
Documentation	9	9	6	5	7	7
Collaboration and Communication	4	5	9	8	8	8
Process Automation	2	3	5	5	9	10
Compatibility with Organizational Structure	8	7	7	8	6	7
Feedback	3	4	9	8	9	9
Support and Training	5	6	7	7	8	8

Justification of Ratings

Flexibility and Adaptability:

- Cascade Model (2): Low flexibility due to rigid sequential stages.
- V-Model (3): Slightly more flexible, but still follows a linear sequence.
- Scrum (9): Highly flexible, with an iterative process that allows for quick adaptation.
- Kanban (8): High flexibility thanks to visual workflow and task management.
- DevOps (8): Flexibility through continuous integration and deployment.
- CI/CD (9): Very high flexibility due to continuous processes enabling rapid adaptation.

Risk Management:

- Cascade Model (5): Limited risk management in early stages, more feasible in later stages.
- V-Model (7): Better risk management through stage-wise testing.
- Scrum (8): Regular sprints and retrospectives facilitate timely risk identification and mitigation.
- Kanban (7): Continuous visual management aids in risk control.
- DevOps (9): Effective risk management through continuous monitoring and automated testing.
- CI/CD (9): High level of risk management due to continuous integration and deployment.

Development Speed:

- Cascade Model (3): Slow process due to strict sequential stages.

- V-Model (4): Slightly quicker but still follows a sequence.
- Scrum (8): Fast development due to short iterations.
- Kanban (7): Speeds up development through visual task management.
- DevOps (9): High speed due to automation and continuous processes.
- CI/CD (9): Very fast development thanks to continuous processes.

Costs:

- Cascade Model (6): Moderate costs, which can rise with changing requirements.
- V-Model (6): Moderate costs with better risk control.
- Scrum (7): Efficient resource utilisation but requires an experienced team.
- Kanban (7): Efficient resource management, though requires discipline.
- DevOps (8): Cost-effective due to automation.
- CI/CD (8): Cost-effective through continuous processes.

Integration of New Technologies:

- Cascade Model (3): Difficult to integrate new technologies due to rigid stages.
- V-Model (4): Slightly better, but still limited.
- Scrum (7): Easier to integrate new technologies through iterations.
- Kanban (7): Facilitates new technology integration through constant task management.
- DevOps (9): High integration capability due to automation.
- CI/CD (9): Very high integration of new technologies due to continuous processes.

Project Management Simplicity and Clarity:

- Cascade Model (6): Clear structure but complex change management.
- V-Model (7): Clear structure with better risk management.
- Scrum (6): Requires discipline and team experience.
- Kanban (8): Simple and visually clear management.
- DevOps (7): Complex implementation but effective management post-integration.
- CI/CD (7): Complex implementation but clear management once processes are set up.

Scalability:

- Cascade Model (6): Suitable for large projects but not for dynamic changes.
- V-Model (7): Good scalability for projects requiring high control.
- Scrum (8): Scales well for various project types.
- Kanban (8): Scales effectively for projects of any size.
- DevOps (9): Excellent scalability for large and complex projects.
- CI/CD (9): High scalability due to automation and continuous processes.

Documentation:

- Cascade Model (9): High level of documentation.
- V-Model (9): Excellent documentation due to clear structure.
- Scrum (6): Documentation can be minimal; emphasis on code and interaction.
- Kanban (5): Documentation varies by team and can be minimal.
- DevOps (7): Good documentation, but focus on automation and processes.
- CI/CD (7): Good documentation, but emphasis on automation and processes.

Collaboration and Communication:

- Cascade Model (4): Limited collaboration, more formal communication.
- V-Model (5): Improved communication but still limited by structure.

- Scrum (9): High collaboration and communication through daily meetings and retrospectives.
- Kanban (8): Good collaboration due to task visualisation.
- DevOps (8): High collaboration between development and operations teams.
- CI/CD (8): High collaboration through continuous processes and automation.

Process Automation:

- Cascade Model (2): Low level of automation.
- V-Model (3): Slightly better but still limited.
- Scrum (5): Automation is possible but not the focus.
- Kanban (5): Automation is possible but not the focus.
- DevOps (9): High level of automation.
- CI/CD (10): Very high level of automation due to continuous processes.

Compatibility with Organisational Structure:

- Cascade Model (8): Well-suited for organisations with rigid structures.
- V-Model (7): Suitable for organisations with rigid structures.
- Scrum (7): Requires adaptation of structure but can work well.
- Kanban (8): Easily integrates into any structure.
- DevOps (6): Requires significant changes to organisational structure.
- CI/CD (7): Requires changes, but less complex than for DevOps.

Feedback:

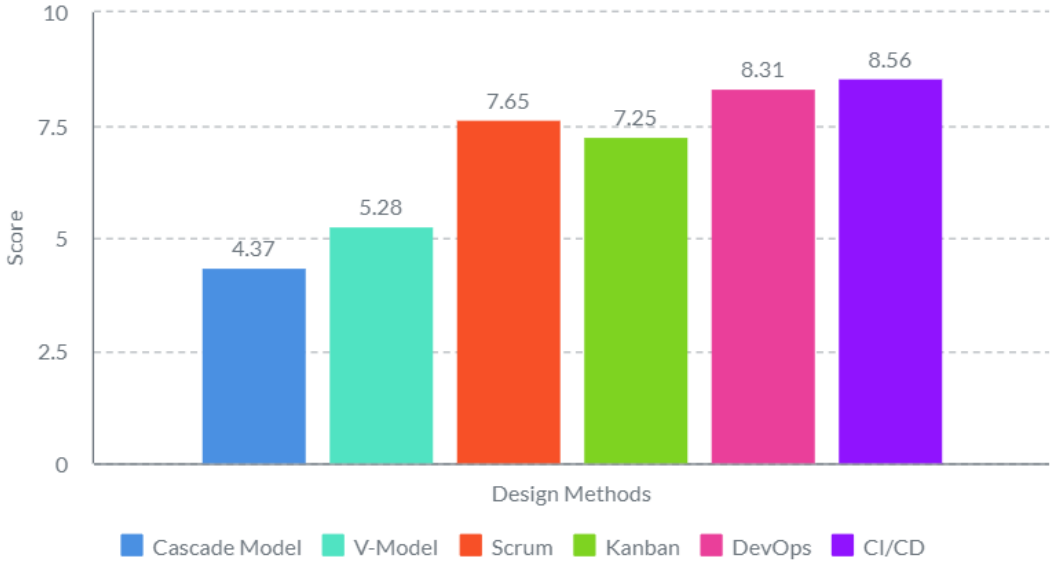
- Cascade Model (3): Limited feedback due to sequential stages.
- V-Model (4): Slightly better but still limited.
- Scrum (9): Regular, constructive feedback through sprints and retrospectives.
- Kanban (8): Regular feedback through continuous task management.
- DevOps (9): High level of feedback through continuous monitoring.
- CI/CD (9): High level of feedback through continuous monitoring.

Support and Training:

- Cascade Model (5): Average level of support and training.
- V-Model (6): Slightly better than the cascade model.
- Scrum (7): Requires significant training and support.
- Kanban (7): Requires training but relatively easy to learn.
- DevOps (8): High level of support and training.
- CI/CD (8): High level of support and training

The final scores for each design method, taking into account criteria weights, are shown in Figure 1.

Figure 1: Evaluation Summary



7. Conclusions

The evaluation of various methods for designing information systems produced measurable data, providing insights into each approach's effectiveness under different conditions. The assessed methods include the Cascade Model, V-Model, Scrum, Kanban, DevOps, and Continuous Integration/Continuous Deployment (CI/CD).

The Cascade Model, with its sequential development stages, scored the lowest overall, achieving 4.37 out of 10. Its main drawback is its inflexibility, making it difficult to adapt quickly to changing requirements. Despite scoring well in documentation (9/10) due to its structured approach, it falls short in flexibility and adaptability (2/10), crucial for dynamic projects. This lack of flexibility also affects its development speed (3/10) and the integration of new technologies (3/10). However, it remains relevant for projects with stable and predictable requirements.

The V-Model, a variant of the Cascade Model that includes validation and verification at each stage, shows slight improvements but still struggles with flexibility, scoring 3/10. It excels in risk management (7/10) due to its thorough testing protocols, which help detect and mitigate defects early. With an overall score of 5.28, it ranks slightly higher than the Cascade Model but is still less effective than more agile methodologies. It is suitable for projects with clearly defined and stable requirements.

Scrum is notable for its high flexibility and adaptability, scoring 9/10. Its iterative process and regular feedback loops, such as sprints and retrospectives, allow for rapid adjustments. Scrum also scores high in collaboration (9/10) and proactive risk management (8/10). Its overall score of 7.65 makes it ideal for projects that require frequent reassessment and quick adaptation, fitting well in fast-paced development environments.

Kanban, known for its simplicity and continuous workflow, achieved an overall score of 7.25. It offers good flexibility (8/10) and helps manage ongoing changes without disrupting progress. Kanban's visual task management system enables teams to adapt quickly, which is crucial in environments with evolving requirements. While slightly less structured than Scrum, it fosters strong team collaboration and communication (8/10), essential in fast-paced and complex projects.

DevOps, nearing the capabilities of CI/CD, scored 8.31 overall. It integrates development and operations for smooth and efficient production cycles, emphasising automation and continuous processes. This approach scores high in risk management (9/10) and development speed (9/10). DevOps improves project outcomes through enhanced collaboration across all teams and promotes a culture of continuous improvement and high efficiency.

CI/CD, with the highest overall score of 8.56, excels in areas requiring high flexibility (9/10) and rapid development cycles (9/10). Its structure supports continuous integration of new code and immediate deployment, significantly reducing development time. The high degree of automation (10/10) minimises human error and boosts efficiency, making it ideal for projects needing frequent updates and quick turnaround times.

8. Discussion

The research on information system design methods has highlighted several noteworthy and important aspects that need further examination. This section discusses the key findings from the analysis and explores their practical applications and significance for various projects and organisations.

A major conclusion of our research is the marked superiority of modern design methods like CI/CD and DevOps over traditional approaches such as the cascade and V-models. This advantage is evident in greater flexibility, faster development speeds, and better integration of new technologies. In today's world, where rapid change and the need for quick adaptation are crucial, these characteristics are essential for project success.

Modern methods enable companies to quickly respond to changes in requirements and market conditions, significantly enhancing the quality and reliability of the final product. For instance, the high level of automation in CI/CD and DevOps minimises human error, reducing the chance of mistakes, which is vital for software requiring high quality and security.

Flexible methodologies like Scrum and Kanban have shown high performance due to their ability to adapt rapidly and provide regular feedback. Scrum, with its sprints and retrospectives, allows teams to promptly respond to changes and improve the development process continuously. This method is beneficial for projects where requirements might evolve throughout the development lifecycle.

Kanban, although less structured than Scrum, also demonstrates high flexibility and ease of implementation. It adapts easily to existing processes and organisational structures, making it attractive to companies seeking gradual improvements without significant changes. Kanban is especially useful in environments requiring constant and quick responses to emerging tasks and issues.

Despite their drawbacks, traditional methods such as the cascade and V-models can still be effective under certain conditions. They provide good documentation and structure, crucial for projects with strict reliability and security requirements. However, their main limitations are low flexibility and slow responsiveness to changes. In fast-changing market conditions and requirements, these methods can significantly slow development and increase costs.

A key factor influencing the choice of design method is the company's organisational structure and culture. Modern methods like DevOps and CI/CD require significant changes in organisational structure and processes, which may be challenging for employees and management to accept. Implementing these methodologies requires not only technical preparation but also cultural shifts, which can be complex and time-consuming.

In contrast, flexible methodologies like Scrum and Kanban may be easier to implement within existing structures without major changes. These methods demand fewer adjustments in organisational culture and can be gradually adapted to meet the company's specific needs.

The choice of design method also has a substantial impact on business processes. Modern methods such as CI/CD and DevOps significantly accelerate the development and implementation process, thereby improving the company's market competitiveness. High levels of automation and integration reduce delays and enhance product quality, crucial in environments with stringent reliability and security requirements. Flexible methodologies like Scrum and Kanban also enhance business processes by increasing transparency and improving communication within the team. Regular sprints and retrospectives enable quick identification and resolution of issues, boosting overall efficiency and development quality.

The research indicates that the choice of information system design method should be carefully justified and take into account all project aspects and organisational characteristics. However, numerous factors can still influence the effectiveness of a particular method under specific conditions. Further research could focus on a more detailed examination of the influence of specific factors, such as team size, project complexity, and security requirements, on the choice of design method.

Additionally, it is important to consider that technologies and design methods continue to evolve. New tools and approaches, such as artificial intelligence and machine learning, can significantly change the future development and implementation process of information systems. Researching these new technologies and their integration with existing design methodologies could be a crucial direction for future studies.

In conclusion, selecting an information system design method is a complex and multifaceted process that requires consideration of numerous factors. Modern methods such as CI/CD and DevOps have shown high performance across many criteria, but their implementation demands significant effort and resources. Flexible methodologies like Scrum and Kanban demonstrate high efficiency, especially in environments with rapidly changing requirements and market conditions. Despite their limitations, traditional methods can still be effective in certain scenarios. Therefore, the choice of method should be based on thorough analysis and consideration of all specific project needs and organisational characteristics.

9. Conflict of Interest

None

10. References

1. Buede, D. M., & Miller, W. D. (2024). The engineering design of systems: models and methods. John Wiley & Sons.
2. Vayansky, I., & Kumar, S. A. (2020). A review of topic modeling methods. Information Systems, 94, 101582.
3. Al-Saqqa, S., Sawalha, S., & AbdelNabi, H. (2020). Agile software development: Methodologies and trends. International Journal of Interactive Mobile Technologies, 14(11).
4. Sarhadi, P., Naeem, W., Fraser, K., & Wilson, D. (2022). On the Application of Agile Project Management Techniques, V-Model and Recent Software Tools in Postgraduate Theses Supervision. IFAC-PapersOnLine, 55(17), 109-114.
5. Russo, D. (2021). The agile success model: a mixed-methods study of a large-scale agile transformation. ACM Transactions on Software Engineering and Methodology (TOSEM), 30(4), 1-46.

6. Fagarasan, C., Cristea, C., Cristea, M., Popa, O., & Pisla, A. (2023). Integrating Sustainability Metrics into Project and Portfolio Performance Assessment in Agile Software Development: A Data-Driven Scoring Model. Sustainability, 15(17), 13139.
7. Palle, R. R. (2020). Compare and contrast various software development methodologies, such as Agile, Scrum, and DevOps, discussing their advantages, challenges, and best practices. Sage Science Review of Applied Machine Learning, 3(2), 39-47.
8. Cao, X. L., & Hu, Y. Y. (2021, April). Development Trend Layered Technology Optimization Strategy in Computer Software Development. In Journal of Physics: Conference Series (Vol. 1881, No. 3, p. 032046). IOP Publishing.
9. M. Leonard and F. Adreit, "Strategic issues of information systems modelling," *Proceedings of IEEE Systems Man and Cybernetics Conference - SMC*, Le Touquet, France, 1993, pp. 343-348 vol.1
10. Peffers, K., Tuunanen, T., Rothenberger, M. A., & Chatterjee, S. (2007). A Design Science Research Methodology for Information Systems Research. Journal of Management Information Systems, 24(3), 45–77.
11. Nugraha, Y. (2020). Information system development with comparison of waterfall and prototyping models. JURNAL RISTEC: Research in Information Systems and Technology, 1(1), 126-131.
12. Boehm, B. W. (1988). A spiral model of software development and enhancement. Computer, 21(5), 61-72.
13. Martin, J. (1991). Rapid application development. Macmillan Publishing Co., Inc.
14. Salo, O., & Abrahamsson, P. (2008). Agile methods in European embedded software development organisations: a survey on the actual use and usefulness of Extreme Programming and Scrum. IET software, 2(1), 58-64.
15. Coronel, P. V., & Benítez, A. Y. (2020). Scrum with eXtreme Programming: An Agile Alternative in Software Development. Innovation and Research: A Driving Force for Socio-Econo-Technological Development, 1277, 350.
16. Narendiran, A., Abhishek, D., Adithya, P., Ray, D., Auradkar, P. K., & Phalachandra, H. L. (2023, April). Integrated log-aware CI/CD pipeline with custom bot for monitoring. In 2023 8th International Conference on Cloud Computing and Big Data Analytics (ICCCBDA) (pp. 257-262). IEEE.
17. Pargaonkar, S. (2023). A Comprehensive Research Analysis of Software Development Life Cycle (SDLC) Agile & Waterfall Model Advantages, Disadvantages, and Application Suitability in Software Quality Engineering. International Journal of Scientific and Research Publications (IJSRP), 13(08).
18. Khan, A. A., Akram, M. U., Butt, W. H., & Sirshar, M. (2024). An Enhanced Agile V-Model: Conformance to regulatory bodies and experiences from model's adoption to medical device development. Heliyon.
19. Hron, M., & Obwegeser, N. (2022). Why and how is Scrum being adapted in practice: A systematic review. Journal of Systems and Software, 183, 111110.
20. Herdika, H. R., & Budiardjo, E. K. (2020, September). Variability and commonality requirement specification on agile software development: Scrum, xp, lean, and kanban. In 2020 3rd International Conference on Computer and Informatics Engineering (IC2IE) (pp. 323-329). IEEE.
21. Yarlagadda, R. T. (2021). DevOps and its practices. International Journal of Creative Research Thoughts (IJCRT), ISSN, 2320-2882.

22. Taibi, D., Cai, Y., Weber, I., Mirakhorli, M., Godfrey, M. W., Stough, J. T., & Pelliccione, P. (2023). Continuous Alignment Between Software Architecture Design and Development in CI/CD Pipelines. In Software Architecture: Research Roadmaps from the Community (pp. 69-86). Cham: Springer Nature Switzerland.
23. Mora, M., Wang, F., Phillips-Wren, G., & Gómez, J. M. (2021). The role of DMSS analytics tools in software project risk management. Volume II Project Risk Management, Walter de Gruyter GmbH: Berlin, Engemann K and O'Connor R, eds, 49-74.
24. Palczynski, J., & Becker, M. (2023). Reconciling Software Development Speed and Robustness with Optimally Balanced Static Application Security Testing.
25. Rodríguez Sánchez, E., Vázquez Santacruz, E. F., & Cervantes Maceda, H. (2023). Effort and Cost Estimation Using Decision Tree Techniques and Story Points in Agile Software Development. Mathematics, 11(6), 1477.
26. Al-Saqqa, S., Sawalha, S., & AbdelNabi, H. (2020). Agile software development: Methodologies and trends. International Journal of Interactive Mobile Technologies, 14(11).
27. Drury-Grogan, M. L. (2021). The changes in team cognition and cognitive artifact use during agile software development project management. Project Management Journal, 52(2), 127-145.
28. Sati, S. O., Sati, M., & Emshiheet, M. (2024, January). Control Plane Scalability of Software Defined Networking. In 2024 ASU International Conference in Emerging Technologies for Sustainability and Intelligent Systems (ICETSIS) (pp. 1830-1834). IEEE.
29. Ebert, F. (2020, September). From Transient Information to Persistent Documentation: Enhancing Software Documentation. In 2020 IEEE International Conference on Software Maintenance and Evolution (ICSME) (pp. 849-853). IEEE.
30. Qian, C., Cong, X., Yang, C., Chen, W., Su, Y., Xu, J., ... & Sun, M. (2023). Communicative agents for software development. arXiv preprint arXiv:2307.07924.
31. Tyagi, A. K., Fernandez, T. F., Mishra, S., & Kumari, S. (2020, December). Intelligent automation systems at the core of industry 4.0. In International conference on intelligent systems design and applications (pp. 1-18). Cham: Springer International Publishing.
32. Nielsen, J. E., Babić, V., Stojanović-Aleksić, V., & Nikolić, J. (2019). Driving forces of employees' entrepreneurial intentions-leadership style and organizational structure. Management: Journal of Sustainable Business and Management Solutions in Emerging Economies, 24(3), 59.
33. Kannan, V., Basit, M. A., Youngblood, J. E., Bryson, T. D., Toomay, S. M., Fish, J. S., & Willett, D. L. (2017, November). Agile co-development for clinical adoption and adaptation of innovative technologies. In 2017 IEEE Healthcare Innovations and Point of Care Technologies (HI-POCT) (pp. 56-59). IEEE.
34. Ngxongo, T. S. P., & Sibiya, M. N. (2014). Challenges regarding the implementation of the basic antenatal care approach in eT hekwini District, Kwazulu-Natal. Journal of Nursing Management, 22(7), 906-913.