A Comparative Analysis of Software Development Methodologies

Gbaranwi, Barima Precious, Ojekudo, Nathaniel Akpofure

Department of Computer Science, Faculty of Natural and Applied Sciences, Ignatius Ajuru University of Education, Rivers State, Nigeria

***Abstract* – Due to the advancement in technology, the operations of many organizations are automated in recent years. This is achieved using different softwares. Each software is developed using a particular method or a combination of two methods. The software development method adopted plays a significant role in the overall software development process. Nevertheless, most organizations and developers have difficulty in choosing the right software development methodology. This study discusses software development methodologies, different software development stages. It provides an analysis of various methodologies, highlights their strengths and drawbacks. It also examines the diverse methodologies in terms of suitability and when it is appropriate to use them.**

***Keywords*: Software engineering, software development stages, methodologies, comparison, suitability**

I. INTRODUCTION

The operations of many organizations are automated in recent years due to the advancement in information technology. This is achieved using different softwares. Nevertheless, most organizations have difficulty in choosing the right software development methodology. The software development methods adopted plays a significant role in the overall software development process. Software development methods are needed to help streamline the activities so that it is not only completed within the right time frame but also must have good quality. There are various methods of software development. Each method of software development has its strengths and drawbacks. Thus, the selection of software development methods to be used for the development of the software should align with the expectation and operations of the organization and users. There are different methodologies and stages of software development. It is therefore essential to determine the suitability of the various methodologies. This work is aimed at providing organizations, project teams and software developers with vast information and direction on when it is appropriate to adopt specific software development methodologies based on the domain and needs of organizations and users.

The rest of the work is organized as follows: Section II reviews related work; Section III discusses different stages of software development, Section IV provides analysis of different methodologies, strengths, and weaknesses. It also examines different methodologies to explore their suitability and usage, and the work is concluded in Section V.

II. RELATED WORK

There are different researches on software development methodologies and their effect on the end product, which is a functional system. Some of the researches are highlighted thus.

Each method of software development has its drawbacks. Thus, the selection of a specific methodology should conform to the expectations of the software to be developed [1] noted. The study compared three methodologies which include parallel, v-shaped and iterative. However, it was noted that v-shaped and iterative methods perform better in terms of delivery time than waterfall whereas waterfall is less expensive than the others. The study however had a limited scope. Other methodologies were not considered. It was suggested that further research consider the methodologies that are appropriate to apply to a particular group which is the gap that this study seeks to bridge.

Many organizations are shifting from traditional to modern methods of software development such as the agile methodology to get timely deliveries [2]. Software projects have become essential in many organizations in recent years. The success of a software project depends on which process model or method is used for development [3]. Software engineering has different principles and it is characterized by high dynamics of technology [4] posited.

It is expected that a useful and quality product is derived from the software development process within the timeframe at a considerable cost, but this is not always the case [5]. Some parameters that affect a software project includes stakeholder participation, software metrics, quality of delivered system or software, among others were explored.

In order to achieve effective planning and management, the software development is divided into different phases. This is referred to as software development methodology [6]. When the process of development begins, requirements, priorities, and technology may change. The choice of a methodology in recent time depends on several factors. One of such is the ability to adapt or accommodate changing requirements.

[7] noted that organizations are frequently changing their software requirements to meet the modern competitive business environment. Timely delivery of developed software are also much anticipated.

There are several methods for developing a software. Choosing a suitable methodology could be very difficult [8]. It requires developers with vast technical and management skills. Project details, complexity of the project, and experience of the team lead have a bearing on the choice of methodology. Frequently changing requirements have negative effect on the project timeline as it increases the budget and elongates the completion time.

III. SOFTWARE DEVELOPMENT STAGES

3.1 Software development stages

Software development is a process consisting of several distinct stages. In software engineering, the software development activities are divided into different phases so as to achieve better planning and management. Depending on the nature of the project, certain stages gain additional weight in the overall effort to implement the software product. The various phases could be found in the following discussions.

3.1.1 Requirement Specification

This is the stage where the project owner and the project team or developers meet to consider the product to be developed and explicitly define what the software is intended to do, the requirements, and the constraints if any. Having interacted with the customer or project owner, the developers led by the team lead or project manager will evaluate the requirements from both business and technical perspectives to determine its feasibility.

3.1.2 Planning

This is the stage where all of the elements are in place for the software to be developed. Defining the overall flow of the application is the first step in planning. The next stage is to break down the flow into smaller subassemblies that are easier to handle. A thorough set of functionalities must be defined for each subassembly. The technology that will be used to develop the application will be decided by the project manager or team lead, in collaboration with members of the project team.

3.1.3 Design

The requirements are assigned to either hardware or software in the design phase. The architecture and layout of the application to be built are mapped out, along with the interconnections between the various levels of abstraction. The planning and programming stages may overlap with the design stage. It is an important phase because it allows the project owner or end user to see a preview of the application before it is produced. If necessary, the project owner might make more revisions or come up with new needs at this point.

3.1.4 Development

The stage of development is when code is written and the software application is created. The development stage begins with the setup of the development and testing environments. The development and test environments should always be synced with the same protocol. Progress monitoring is another crucial part of the development stage. The project manager must assess real progress and compare it to the original plan. Software engineers are expected to debug their code while writing it in order to deliver bug-free updates to the testing environment. While writing, software developers should utilize comments to make it easier to comprehend afterwards or for others to understand.

3.1.5 Testing

Testing is the process of identifying and correcting programming and design flaws. Programming mistakes occur when a program performs in a way that is inconsistent with its intended architecture. Usability difficulties are also part of programming faults. It is considered inappropriate if the application is vulnerable to assaults and hence allows attackers access to private data. If users complain about the application's delayed response time, this is also a programming issue. Inconsistencies between what the project owner wanted and what the project team ended up executing are known as design flaws. Design errors arise at the planning stage, have a large influence on the project, and are typically more difficult to correct. Detecting design flaws is much easier when the project owner is involved, as he is the one who came up with the application requirements.

3.1.6 Setup

Setup refers to the process of installing the application in a live environment. The setup stage comes before the actual use of the software product. Configuring the live environment in terms of security, hardware, and software resources is part of the setup. The software product's actual setup entails copying the source code, importing the database, and, if necessary, installing third-party programs. After the application has been installed, it will go through a second round of testing. Content is added to the application after testing is completed.

3.1.7 Maintenance

Maintenance is the stage that involves software development after the program has been set up, as well as verifying that the application is operating within the parameters that have been set. In addition, tracking traffic data will provide useful information about any issues that could impair the application's performance. Systematically checking functionality for mistakes that were not found in the testing stage or for issues that were not reflected in the error logs is an important aspect of the maintenance stage. The maintenance stage also allows you to enhance the developed program by adding new features or functionalities. They may be found under several naming standards depending on software development approach, with overlapping and shifting order.

IV SOFTWARE DEVELOPMENT METHODOLOGIES

4.1 Software Development Methodologies

There are various software development methodologies or models. However, the following are considered in this research. The Waterfall, spiral, Prototyping, agile, scrum, extreme programming, rapid application development, v-shaped model-driven methodology, etc

4.1.1 The Waterfall Methodology

The Waterfall methodology involves a sequential process where each phase starts only after at the preceding phase is completed. Each stage has its own deliverables. The principal phases of this methodology conform to the fundamental software development activities. The feedback from the project owner or sponsor is only received after the software has been completely developed and tested. It is suitable for small scale projects where all the requirements are known, clearly defined from the beginning and it is not expected to change throughout the entire process.

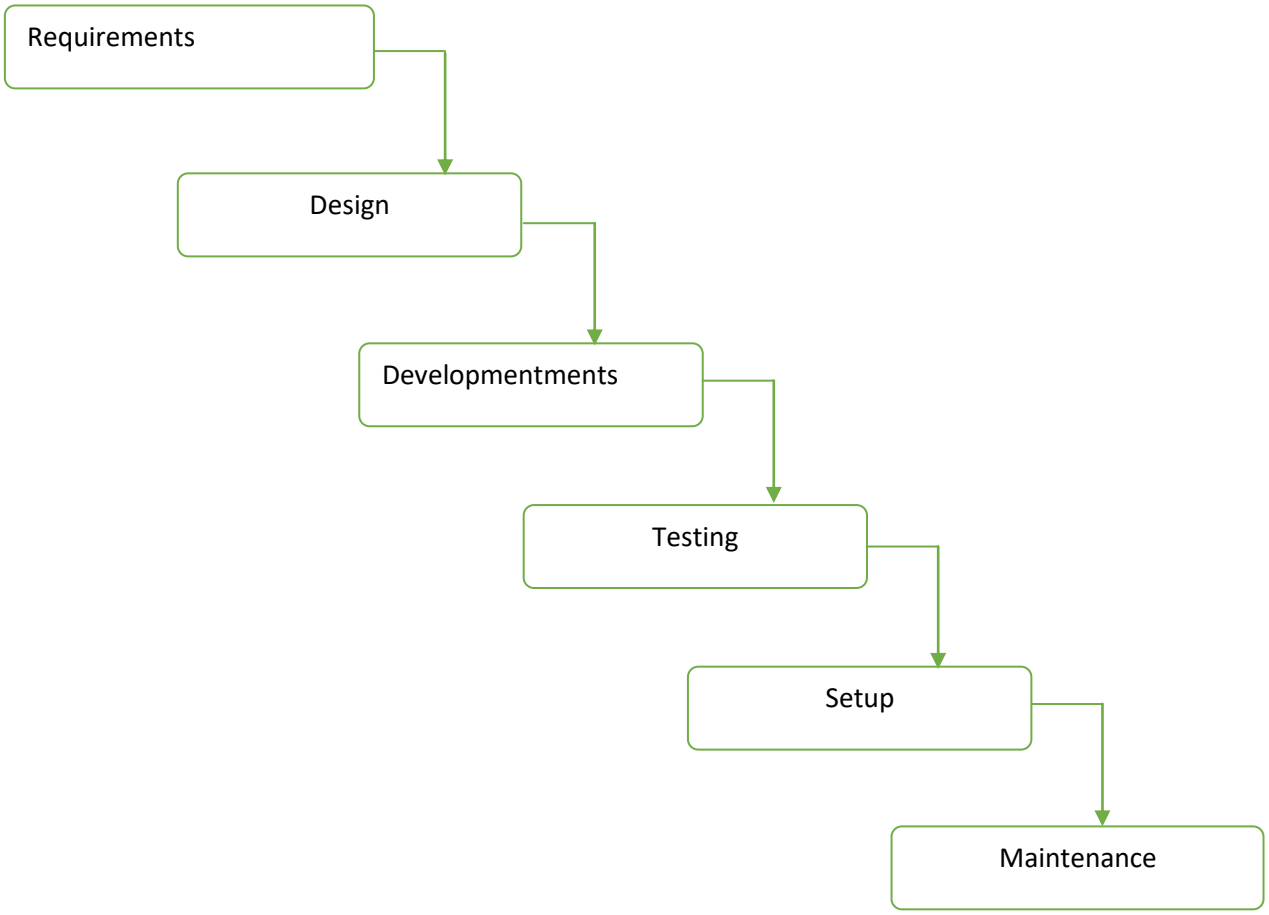

Figure 1. Waterfall methodology

4.1.2 Incremental Methodology

This methodology involves the development of different versions of the system. After developing an initial version, another version is developed; more functions are added thereby increasing performance. After the development of each version, unit testing is conducted before full implementation. It is better than the waterfall. The risk of failure is reduced, and it is easy to improve on subsequent versions of the software. After each iteration, the project owner is contacted for comments. This approach favours design over documentation and is appropriate for medium to big projects. The incremental process breaks down an issue into small chunks, preventing the development team from becoming overwhelmed by long requirements. The understanding of the requirements for later versions becomes easier based on the experience gained from developing the earlier versions.

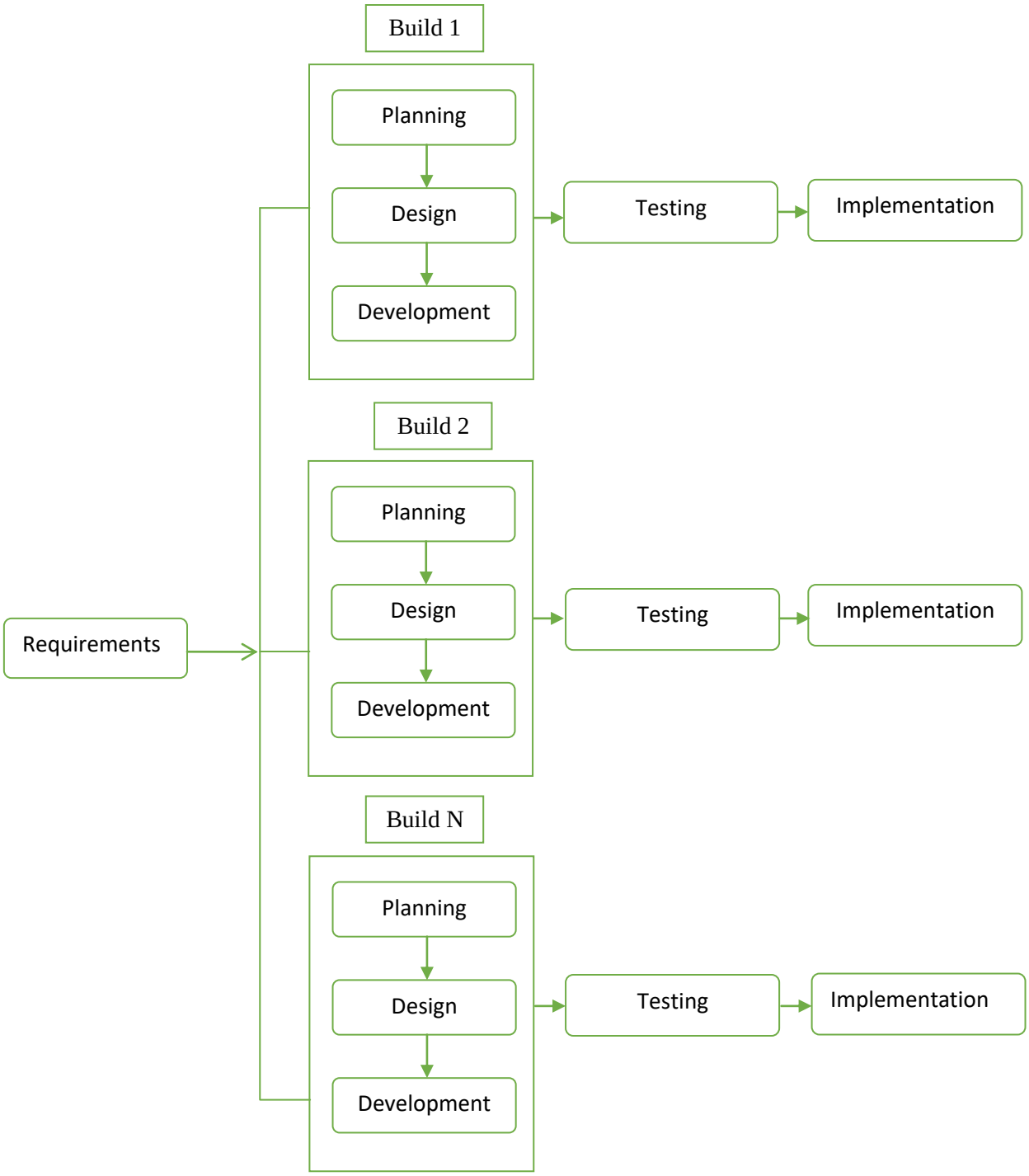

Figure 2. Incremental methodology

4.1.3 Prototyping

Prototyping is a practice that arose from the need to clearly define specifications, and it requires creating a demo version of the product with all of the necessary features. Initial requirements are just defined to offer enough information to construct a prototype. The prototype is utilized to fine-tune specifications and serves as a communication baseline between the project team and the project sponsor. The prototype is not intended to be developed further into a finished software product. After the prototype is finished, the project owner provides input. It is appropriate for large-scale projects when it's tough to establish all of the requirements in detail before starting to code. Prototyping is useful for projects that are unique or original and have no precedent.

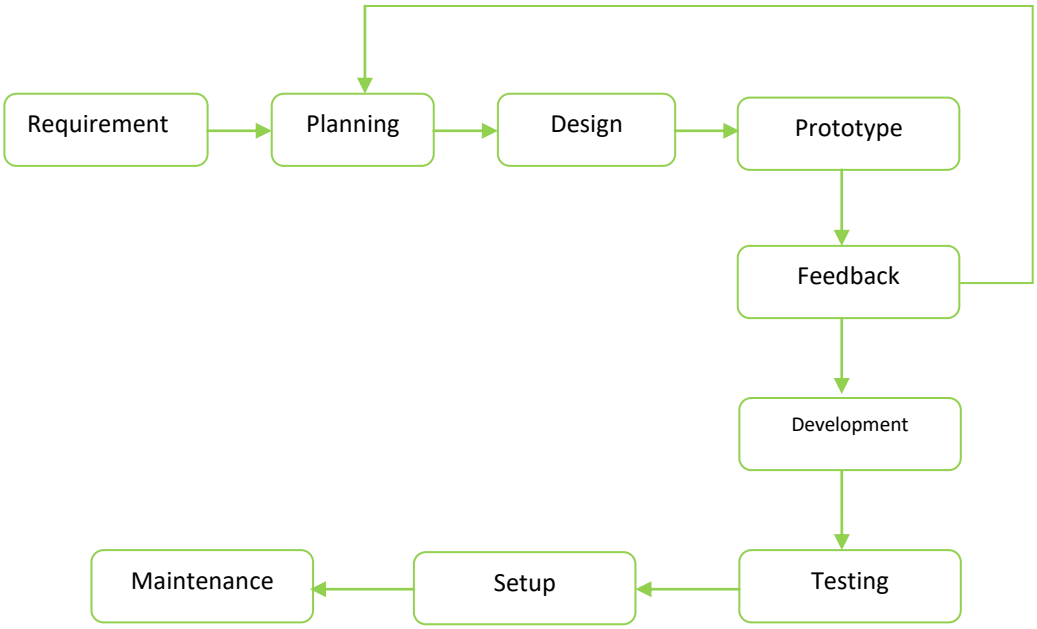

Figure 3. Prototyping methodology

4.1.4 Rapid Application Development

This methodology provides faster development and higher-quality results than those achieved with the traditional methodologies. It requires less time and emphasis on planning tasks and more emphasis on the actual development. Development cycles are time-bound and multiple cycles can be developed at the same time. The project owner's feedback is received after each module is completed. The Rapid application development methodology is suitable for small, medium and large scale projects with a modular structure.

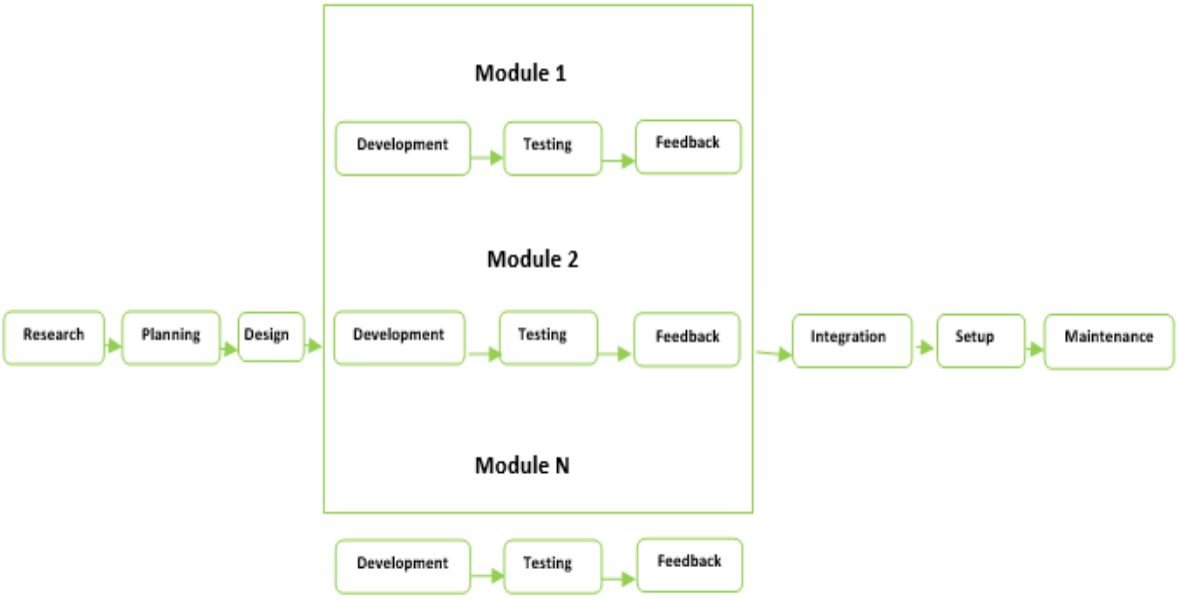

Figure 4. Rapid application development

4.1.5 V-Shape Methodology

It is a software development process which is an extension of the waterfall model. It emphasizes thorough testing by pairing each software development stage with a matching phase of testing. The feedback on the developed software is received in the form of acceptance testing after the entire application is completed. It is suitable for small and medium scale projects.

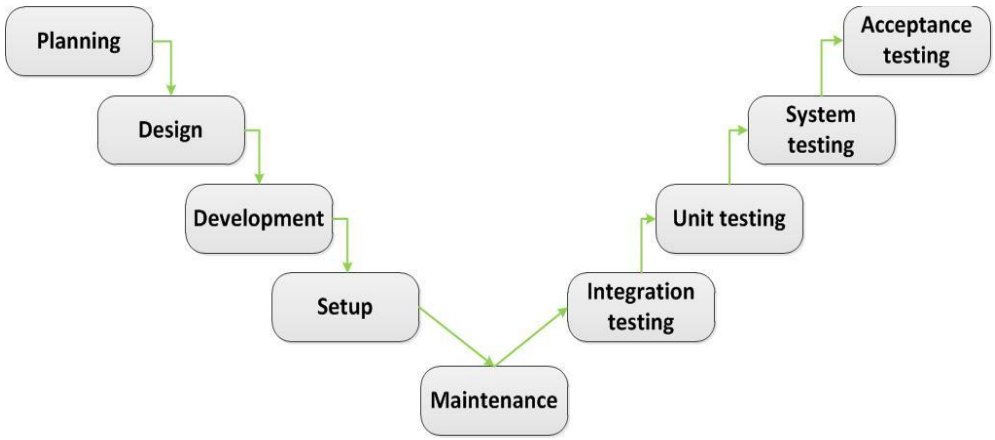

Figure 5. V-Shape Methodology

4.1.6 Model-driven Methodology

The Model-driven methodology uses domain models as ways of handling requirements. The models are developed based on customer's or users requirements. It is platform independent. The model can be migrated onto any environment. Based on the model, the actual code is then generated. This methodology is suitable for small, medium and large scale projects. It is also recommended for projects that will have long development and utilization period as meta-models can be easily migrated and modified to adapt new technologies.

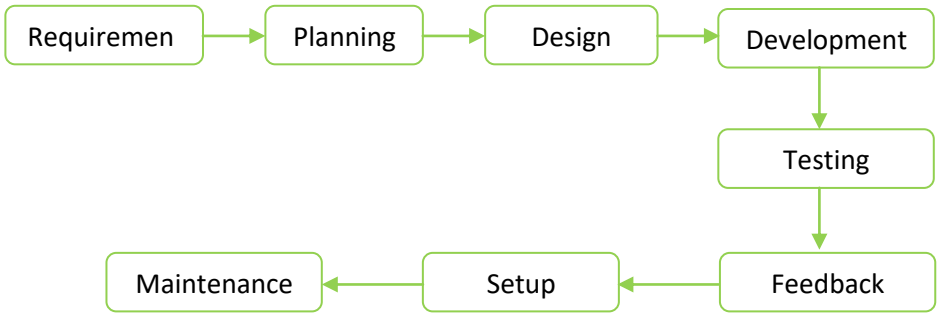

Figure 6. Model-driven methodology

4.1.7 Feature-driven Methodology

Feature-driven development methodology is focused on actual functionality. Each feature in this methodology reads as a requirement, which is understandable by the project owner, it describes the true business value. The project owner's feedback is received after the application is already setup but constant interaction between development team and project owner takes place throughout the entire duration of the project. This development methodology is suitable for small, medium and large scale projects.

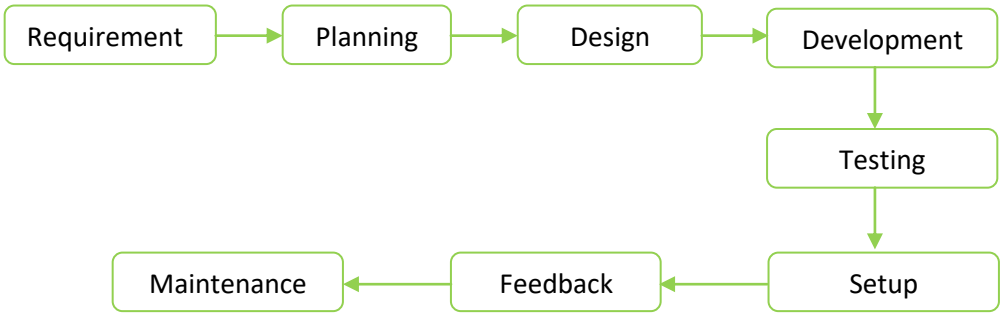

Figure 7. Feature-driven methodology

4.2 Strengths and Drawbacks of Software Development Methodologies

The comparison of the various software development methodologies in terms of strengths and their drawbacks are presented in Table 1.

Table 1. Comparison of Strengths and Drawbacks

Methodology	Strengths	Drawbacks
Waterfall	- It is easy to manage - Easy to understand - Provides comprehensive documentation	- It is affected by changing requirements. - The customer cannot preview the system until very late, almost at the end of the project.
Incremental	- Risk of failure is reduced. - Initial delivery schedule is fast. - Project momentum is easily maintained	- It does not allow for iteration. - Clearly defined interfaces are required. - It requires rigorous planning and design
Prototyping	accurate identification of application requirements - early feedback from the project owner - improved user experience - early identification of missing or redundant functionality	leads to unnecessary increase of the application's complexity - increased programming effort - costs generated by building the prototype
Rapid Application Development	- Development is fast. - Components can be reused. - It requires less manpower - Provides preview of the product	- Skilled and highly trained personnel required. - integration of different modules can be difficult - Poor documentation - It is not cost effective
V-Shaped	- It enhances tracking of progress - It is easy to use - It encourages verification of deliverables	-It does not support iteration - It is affected by change in requirements. - Concurrent activities are not supported.
Model-driven	- Maintenance cost is reduced. - Improved or greater productivity - The degree of compatibility and portability is high. - it requires less time to market.	- Technical expertise is required - Non-domain experts cannot easily grasp the documentation.
Feature-driven	- Multiple teams can work simultaneously on the project - Good progress tracking and reporting capabilities - It is easy to understand and adopt.	- Individual code ownership - Iterations are not well defined

4.3 Suitability and Usage of Software Development Methodologies

The comparison of the various software development methodologies in terms of their suitability and usage are presented in Table 2.

Table 2. Comparison of Suitability and Usage of Software Development MethodologieS

Methodology	Suitability and Usage	Percentage (%)
Waterfall	Small scale projects with no change in requirements.	90
Incremental	Medium and large scale projects with new technology.	80
Prototyping	Large scale projects with unclear requirements, no such projects or previous sample exist.	85
Rapid Application Development	Small, medium and large scale projects with modularity.	95
V-Shaped	Small and medium scale projects	75
Model-Driven	Small, medium and large scale projects, with changes and new technologies, as well as modularity and reusability.	100
Feature-Driven	Small, medium and large scale projects with progress tracking and reporting.	90

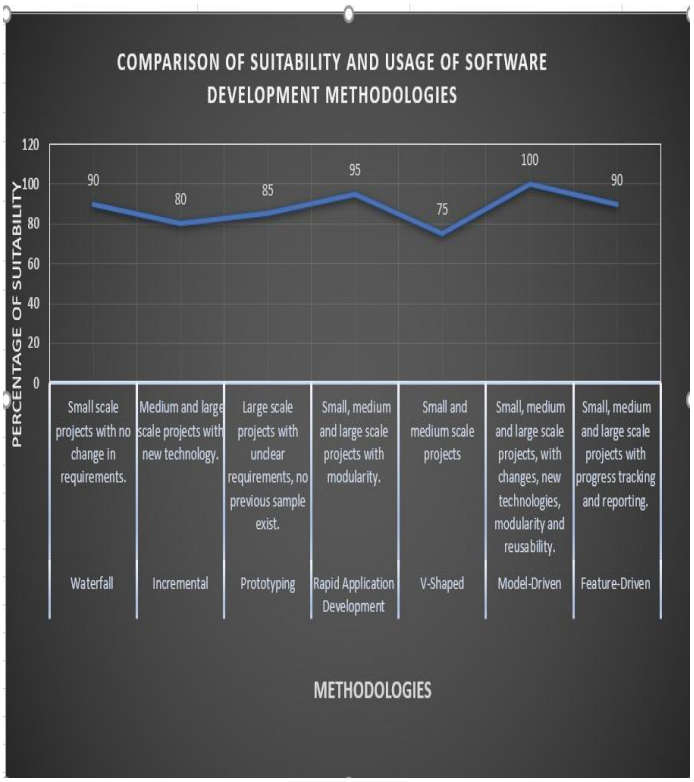

Figure 8. Comparison of suitability of software development methodologies

V CONCLUSION

In this study, a profound discussion has been provided on software development methodologies, software development stages. Different methodologies have been explored including their strengths and weaknesses. A concise, but robust analysis has been provided on when it is

appropriate or suitable to adopt specific software development methodologies based on the system or software to be developed considering specific domain and needs of organizations and users. Compared to other software development methodologies explored in this study, the model-driven methodology can be used for many scenarios. Model-Driven methodology is suitable for small, medium and large scale projects, adapt to changes and new technologies, and as well support modularity and reusability. It is clear that when the appropriate methodology is adopted, development time and cost are reduced, and there is improved quality too. Nevertheless, continuous changing requirements can affect software development project deadlines.

REFERENCES

[1] Nugroho, S., Waluyo, S. H., & Hakim, L. (2017). Comparative analysis of software development methods between parallel, v-shaped and iterative. *International Journal of Computer Applications*, *169*(11), 7 – 11.

[2] Rai, P., & Dhir, S. (2014). Impact of different methodologies in software development process. *International Journal of Computer Science and Information Technologies*, *5*(2), 1112 – 1116.

[3] Dhami, H. P. S. (2016). Comparative study and analysis of software process models on various merits. *International Journal of Advanced Research in Computer Science and Software Engineering*, *6*(9), 234 – 243.

[4] Despa, M. L. (2014). Comparative study on software development methodologies. *Database Systems Journal*, *5*(3), 37 – 55.

[5] Kumar, G., & Bhatia, P. K. (2014). Comparative analysis of software engineering models from traditional to modern methodologies. https://www.researchgate.net/260952270

[6] Kiran, H., Aditya, D., Rahul, C., & Manjula, R. (2016). Comparative analysis of agile software development methodologies – a review. *International Journal of Engineering Research and Applications*, *6*(3), 80 – 85.

[7] Moniruzzaman, A. B. M., & Hossain, S. A. (2013). Comparative study on agile software development methodologies. *Global Journal of Computer Science and Technology*, *13*(7), 5 – 12.

[8] Suranya, P., Monica, V., Priyadharshini, J., & Deepa, N. (2017). Comparative study of software development methodologies. *International Research Journal of Engineering and Technology*, *4*(5), 172 – 179.