# Software Project Management and Underlying Development Factors

Princewill Ibeabuchi and Sayeed Sajal

Department of Math and Computer Science

Minot State University

500 University Ave W, Minot, ND 58707

Email: sayeed.sajal@ndus.edu

## Abstract

A software project has two fundamental activity dimensions; the engineering aspect and project management. In our current generation, businesses have become more reliant on technology for their day to day business operations. Software managers are under increasing pressure to create and deliver advanced software applications on schedule and with limited resource consumptions while taking the risk, controls, management, delivery, productivity factors, cost models, schedule, organizational issues, safety and standard issues, and maintenance into account. The purpose of this tutorial is to showcase the fundamental ideas of current software project management processes and indicate how these concepts relate to the models of developing software projects both from the developers and manager's perspective respectively.

## 1. Introduction

There has been an increase in over-reliance on software systems by the majority of today's companies in their business setting. At the point when the business forms are changed to meet new necessities, the product frameworks need to experience a relating change or grow new supporting programming. This is additionally the situation when an organization presents new business forms. Currently, software systems are frequently interrelated to other systems, building up another framework much of the time includes alterations of old frameworks too. Due to these, factors affecting the advancement of software development becomes hard to anticipate. The improvements of completely new development frameworks and the adjustments of old frameworks are frequently performed in ventures, realizing out that 23% of all product ventures are dropped before fulfillment. Moreover, of the finished ventures, just 28% are conveyed on time, with the normal venture overwhelming the financial plan with 45% [2]. Such has been the size of the venture of time, exertion, and cash in software systems, that making progress in their advancement and organization has expended numerous analysts (and obviously, specialists) over the most recent three decades. As far back as 1975, researchers were endeavoring to distill the determinants of venture achievement and disappointment, so that an effective result was almost certain [1], with blended outcomes. The commitment of this paper is thusly proposed to study 5 of the widely used development models. Additionally, a portion of the elements contemplated is as of now utilized in the current models. Subsequently, a portion of these variables are approved by this examination and some are tested.

## 2. Outline

The of the paper is structured as follows: Section I recounts diverse software development process models and development phases. Section II describes different factors involved in the software project development, from the project idea, development models utilized, and the success or failure of a software project; factors that decide the choice of software to be developed are examined, with what constitutes a successful project and an unsuccessful one. A comparative study of the 5 major models chosen in this paper is highlighted, with how different factors affect their impact on a software project in section III. Finally, section IV summarizes the paper with conclusions.

## Section I

### A. Waterfall model

The Waterfall SDLC (Software Development Life Cycle) model is a successive software development process model, described to be downward flowing, hence where it got the name Waterfall; its process motion is similar through different phases that must be carried out for the software system to be successfully developed. Software process models, with some variations, all evolved from the classical waterfall model, developed in the late 1960s and early 1970s. It is still the best and most widely used framework for describing the software development process [7], by its step by step breakdown of the development processes to minimize their complexity. The diagram below highlights the different phases of the Waterfall model.

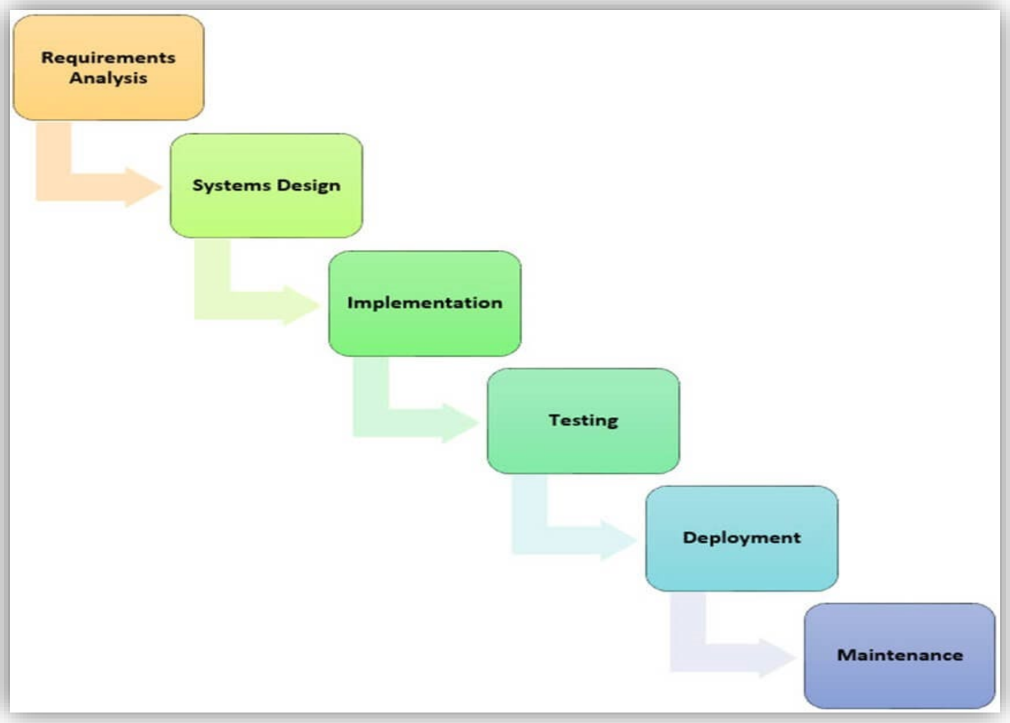

Fig. 1 Waterfall Model [5]

**Requirement Analysis Stage:** Also known as Software Requirements Specification (SRS), is a comprehensive description of the software to be developed. It sets the functional and non-functional requirements. Functional requirements are characterized by means of case utilization per the users' interactions with the software. This includes requirements such as purpose, capacity, outlook, functions, user traits, functional specification and database necessities. While the non-functional requirement deals with the constraints, several criteria, limitations, and requirements imposed on the design and operation of the software rather than on particular behaviors. It includes such properties as reliability, scalability, testability, availability, maintainability, performance, and quality standards.

**System Design Phase:** This process consists of planning and brainstorming for a software solution. The developers and designers have to work together to lay a solution plan that includes the architectural design, efficient algorithm, database specification, user interface design, and data structures.

**Implementation Stage:** It alludes to the acknowledgment of business plan details and prerequisites into a solid executable program, database, site, or programming part, through programming and sending. This is the stage where the genuine code is composed and aggregated into an operational application, and where the database and content records are made. At the end of the day, it is the way toward changing over the entire necessities and diagrams into a generation domain.

**Testing Stage:** It is also called confirmation and approval, which is a procedure for ensuring that a product arrangement meets the first necessities and particulars, and it achieves its planned reason. Indeed, a check is the way toward assessing programming, to decide if the results of a given improvement stage fulfill the conditions intended from the beginning of that stage. While approval is the path to assessing programming amid or towards the end of the improvement procedure, to decide if it fulfills indicated necessities. It is also the stage for debugging the system to correct errors or glitches.

**Deployment Stage:** Refers to the delivery or guarantee that the developed system is available for use to the users.

**Maintenance Stage:** It is the process of changing or updating a software system after deployment or delivery, to correct errors, upgrade performance and quality. Extra upkeep exercises can be performed in this stage including adjusting the system to new developing conditions, obliging new client preference, and expanding programming dependability.

## B. Prototype Model

Prioritizes the development of the actual software, instead of concentrating on documentation. This way, the actual software could be released in advance. Prototyping requires more user involvement and allows them to see and interact with a prototype to provide better and more complete feedback and specifications [4].

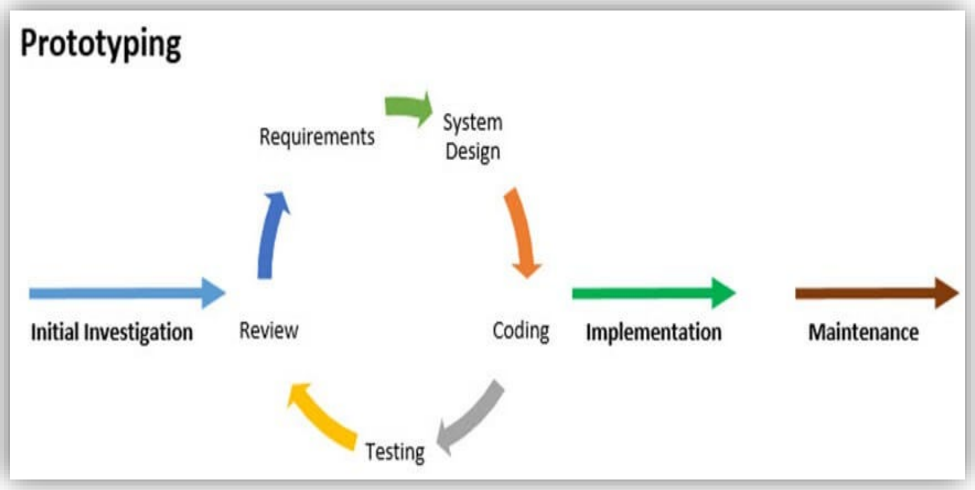

Fig. 2 Prototype Model [5]

## C. Spiral Model

It is a headway methodology depicted by repeatedly underlining a ton of fundamental improvement shapes, and adequately reducing the risk involved. A prototype is produced at the end of risk analysis. Its evaluation phase allows the customer to assess the output of the project to date before the project goes to the next spiral [8].

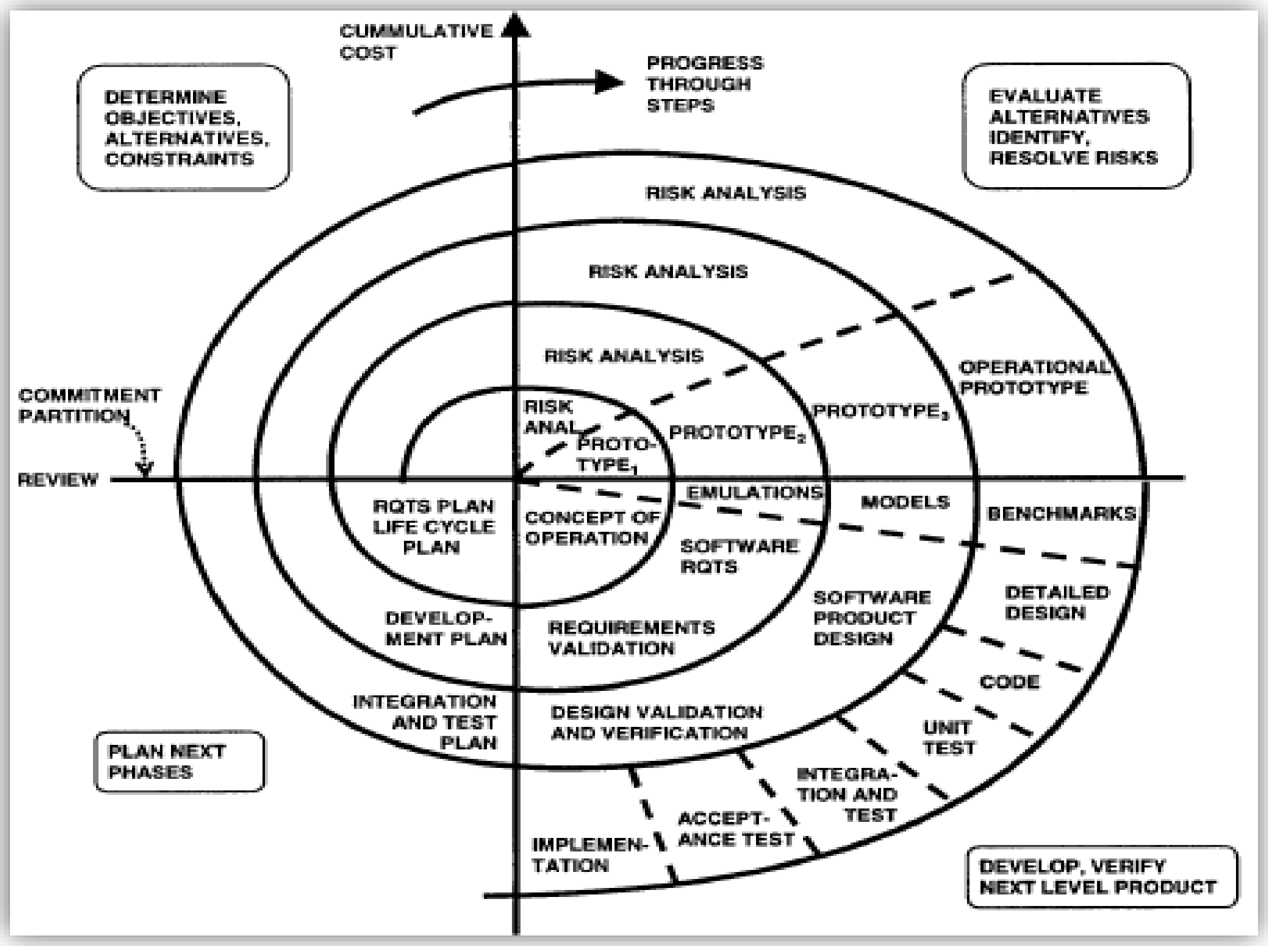

Fig. 3 Spiral Model [6]

## D. Iterative Model

The iterative model is a specific usage of a product advancement life cycle (SDLC), which centers on an underlying rearranged execution. Additional features are designed, tested and, added to the system with the iterations.

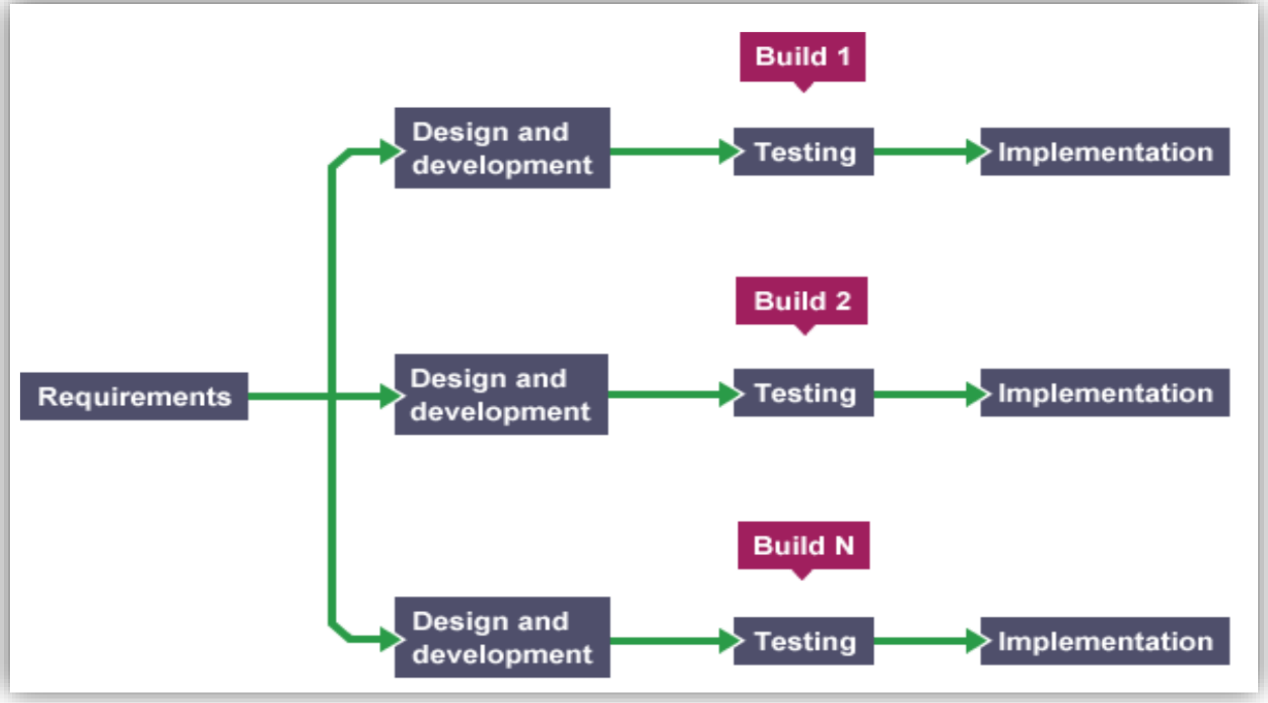

Fig. 4 Iterative Model [10]

## E. Agile Model

It alludes to a gathering of programming advancement philosophies, dependent on iterative improvement, where prerequisites and arrangements advance through a coordinated effort between self-sorting out cross-practical groups.

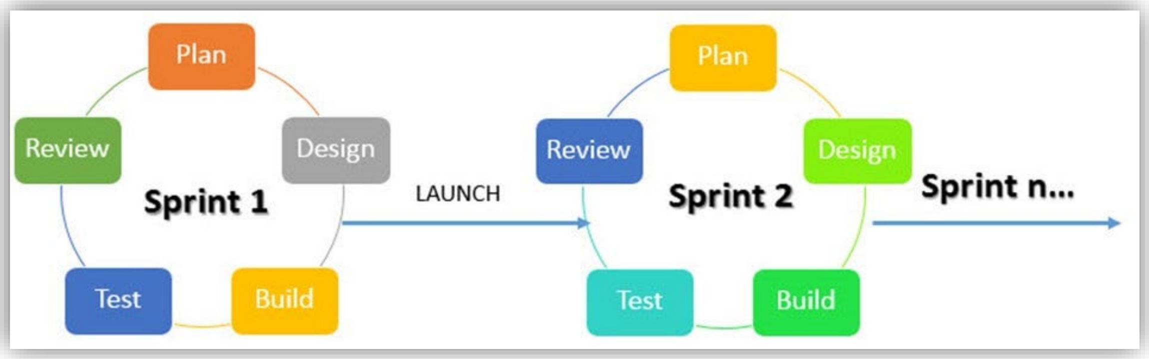

Fig. 5 Agile Model [5]

## Section II

### Deciding Factors on Software Choice and Development Utilized

| ***Factor name*** | ***Definition/Examples*** |
|---|---|
| *Project type* | Example; Desktop, Web-Based, Scheduling, etc. |
| *Size of Project* | An estimation used to quantify the total extent of the development process, including duration, cost, and complexity. E.g. Function Point. |
| *Duration of Project* | Total time to be used in the development process. |
| *Complexity of Project* | Taking into account the complicated factors that affect the development process. E.g. Cost, Time, etc. |
| *Expected Risks* | Estimation, Productivity Issues, Conflict among the Team, etc. |
| *Comprehension of User Requirement* | The poorly specified requirement would definitely lead to development failure. |
| *Comprehension of the Application area* | E.g. Image Processing, Robotics Controllers, Digital Filters. |
| *Involvement of Users/Customers/Shareholders etc.* | Provide data and information directly to the developers, and help clarify the needs of the users, with an initial understanding of the system. |
| *Skills and Experience of Developers* | Knowledgeable and well-practiced in multiple frameworks and coding languages. |
| *Team Size* | How many people working on the project at a time. |
| *Man-machine interaction **(MMI)*** | General control of machines with buttons, remotes, etc. |
| *Availability of Tools and Technology* | They can speed up or limit the efficiency and production rate of the development process. |
| *Level of Consistency* | Consistency in coding makes the process easier to meta-program, debug, refactor and test the system. |

Table1: Factors affecting the choice of Software [8]

### Successful and Unsuccessful Project Planning

| **Successful Projects** | **Failing Projects** |
|---|---|
| Effective project planning | Inadequate project planning |
| Effective project cost estimating | Inadequate cost estimating |
| Effective project measurements | Inadequate measurements |
| Effective project milestone tracking | Inadequate milestone tracking |
| Effective project change management | Inadequate change management |
| Effective project quality control | Inadequate quality control |

Table 2: Project Planning [3]

## A. Findings:

Table 2, displays a comparative study of the 5 software development models briefly explained earlier in this paper. They are compared on the basis of varying factors/features and how they affect a software development project lifecycle. Each model has a unique response to these factors.

## Section III

Comparative Study of Different Software Development Models and their Effects on Development Factors

| **Model/Features** | **Waterfall Model** | **Prototype Model** | **Spiral Model** | **Iterative Model** | **Agile Model** |
|---|---|---|---|---|---|
| Precise Requirement Specifications | Beginning | Frequently Changed | Beginning | Beginning | Frequently Changed |
| Understanding Requirements | Well Understood | Not Well Understood | Well Understood | Not Well Understood | Well Understood |
| Cost | Low | High | Expensive | Low | Very Expensive |
| Guarantee of Success | Low | Good | High | High | Very High |
| Resource Control | Yes | No | Yes | Yes | No |
| Cost Control | Yes | No | Yes | No | Yes |
| Simplicity | Simple | Simple | Intermediate | Intermediate | Complex |
| Risk Involved | High | Low | Low | High | Reduced |
| Expertise Required | High | Medium | High | High | Very High |
| Changes Incorporated | Difficult | Easy | Easy | Easy | Difficult |
| Risk Identification | Initial Level | No | Yes | No | Yes |
| User Involvement | Initial Level | High | High | Intermediate | High |
| Overlapping Phases | No | Yes | Yes | No | Yes |
| Flexibility | Rigid | Very Flexible | Flexible | Less Flexible | Very Flexible |
| Time Frame | Long | Short | Long | Long | Least Possible |

Table 3: Comparison of Different Development Models [9]

## Section IV

## 3. Future Findings

This current study was undertaken for this research work and still has the potential for further extension, by collecting numerous range of data of how individual organizations in different environments, and specialization of software development, have been impacted by their utilized models. There are vast ranges of factors that can impact the outcome of a software project development, like the motivation of the team members, their work environment, access to more advanced evidence-based analysis, and all these play a vital part in the success or failure of project development.

## 4. Conclusion

The Waterfall, Prototype, Spiral, Iterative and Agile models have been studied and analyzed in depth. The utilization of each model varies, as they are best used for a different project with varying features and different end goal. Many of the models have their advantages and disadvantages. The waterfall is one of the oldest models but has its own disadvantages such as no fair division of phases in the life cycle. Not all the errors/challenges related to a phase are resolved during the same phase but instead carried forward to the next stage, which depriving it a lot of time. Whereas the spiral model resolves such issues [9]. The prototype model poses a great challenge in the identification of risks; Spiral is very expensive to utilize; Understanding requirements of the iterative model is hard and makes the development process complicated, and finally, the agile model makes it difficult to incorporate changes to the process.

## References

[1] L. Mcleod, S. G. Macdonell, Factors that Affect Software Systems Development Project Outcomes: A Survey of Research.

[2] R. Lagerstrom, L. M. von Wurtemberg, H. Holm and O. Luczak. Identifying Factors Affecting Software Development Cost. Industrial Information and Control Systems the Royal Institute of Technology Stockholm, Sweden.

[3] C. Jones. Software Project Management Practices: Failure Versus Success. Software Productivity Research LLC. October 2004.

[4] R. G. Sabale, A. R. Dani. Comparative Study of Prototype Model for Software Engineering with System Development Life Cycle. ISSN: 2250-3021 Volume 2, Issue 7(July 2012), PP 21-24.

[5] Guru99. MIS Development Process with SDLC, Waterfall, Prototype & Agile. [Online]. Available: https://www.guru99.com/mis-development-process.html.

[6] B. Boehm, W. J. Hansen. Spiral Development: Experience, Principles, and Refinements. Spiral Development Workshop, February 9, 2000.

[7] J. Jaak. "Software Project Management: The Manager's View," Communications of the Association for Information Systems: Vol. 2, 1999. Article 17.DOI: 10.17705/1CAIS.00217

[8] R. R. Raval, H. M. Rathod. Comparative Study of Various Process Model in Software Development. International Journal of Computer Applications (0975 – 8887). Volume 82 – No 18, November 2013.

[9] D. Jamwal. Analysis of Software Development Models. IJCST Vol. 1, Issue 2, December 2010.

[10] Martin. 7 Basic Software Development Life Cycle (SDLC) Methodologies: Which One is best? [Online]. Available: https://www.cleverism.com/software-development-life-cycle-sdlc-methodologies/