RESEARCH ARTICLE OPEN ACCESS

Evolving a Ultra-Flow Software Development Life Cycle Model

Divya G.R.*, Kavitha S.**
*(*Computer Science, Auxilium College, and vellore*)
**(*Computer Science, Auxilium College, and Vellore*)

----------**********----------

Abstract:

Software development life cycle or simply SDLC (system and software are interchanged frequently in-accordance to the application scenario) is a step-by-step highly structured technique employed for the development of any software. In the era of the software development, there exist a large number of models to develop software. Each model has its own characteristics, limitations, and working environment. According to the requirements, software industry people use different models to develop different software. There are various models but none of them are capable to address the issues of client satisfaction fully. Therefore, this paper has proposed a new kind of software development model called ultra-flow software development life cycle. The ultra-flow SDLC is used for software development that lays a special emphasis on highly structured lifecycle and defining an output with each stage, and also tries to fulfill the objective of the software engineering of developing a high-quality product within schedule and budget. The new proposed model is designed in such a way that it allows a client and a developer to interact freely with each other in order to understand and implement requirements in a better way.

Keywords—Requirements gathering, software development lifecycle (SDLC) model, software engineering, testing, ultra-flow model.

----------**********----------

I. INTRODUCTION

Software engineering is a discipline whose aim is the production of quality software. Software, which is delivered on time within the budget, satisfies its requirements. Software engineering is the area, which is constantly growing. It is a very interesting subject to learn as all the software development industry based on this specified area. There exist the various models to develop software. But most of the existing software development models pay less or a very little attention toward client satisfaction. It matters not only to the client but also to the developer, because it costs far less to retain a client with happy than it does to find a new client. Satisfying client is an essential element for staying in this modern world of global competition. Client satisfaction is so important for the acceptance and delivery of the software product. Software project fails due to the absence of client satisfaction. Software development model must satisfy and even delight client with the value of the software products and services [8], [9].

II. SOFTWARE DEVELOPMENT LIFE CYCLE

Software development lifecycle (SDLC) is a process used by the systems analyst to develop an information system, including requirement, validation, training, and user (see Fig. 1). Any SDLC should result in a high-quality system that meets or exceeds customer expectations, reaches completion within time and cost estimates, works efficiently in the current and planned information technology infrastructure, is inexpensive to maintain, and is cost-effective to enhance.

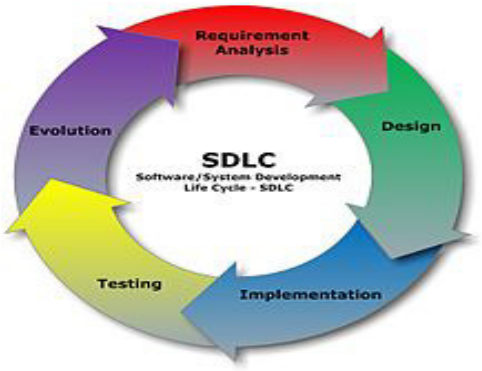

Fig.1 Lifecycle of SDLC

The steps or phases generally involved in a software lifecycle model are as follows:

- Customer communication
- Requirement Analysis
- Design
- Implementation
- Testing
- Deployment and Maintenance

A. Requirement Analysis

Requirement analysis is the initial phase of the SDLC. The goal of this phase is to understand the client's requirements and to document them properly. The emphasis in the requirement analysis is an identifying what is needed from the system. It is the most crucial phase in the SDLC. The output of the requirement analysis is software requirement specification [10].

B. Design

It is the first step to move from the problem domain toward the solution domain. It is the most creative phase in the SDLC. The goal of this phase is to transform the requirement specification into a structure. The output of this phase is software design document (SDD).

C. Coding

In this phase, SDD is converted into code using some programming language. It is the logical phase of the SDLC. The output of this phase is program code.

D. Testing

This is most important and powerful phase. Effective testing will contribute to the delivery of high-quality software products, more satisfied users, lower maintenance costs, and more accurate and reliable results [7].

E. Maintenance

This phase is started after the delivery of the product. If any error occurred or modification needed, it is implemented in this phase.

III. SDLC MODELS

A programming process model is an abstract representation to describe the process from a particular perspective. There are a number of general models for the software process, such as waterfall model, evolutionary development, formal systems development, and reuse-based development. This research will view the following models:

- Waterfall model
- Prototyping model
- V-shaped model
- Spiral model

A. Waterfall Model

This model is one of the oldest models and is widely used in government projects and in many major companies. As this model emphasizes planning in early stages, it ensures design flaws before they develop. The model begins with establishing the system requirements and the software requirements and continues with an architectural design, detailed design, coding, testing, and maintenance [3]. The waterfall model serves as a baseline for many other lifecycle models, as shown in Fig. 2.

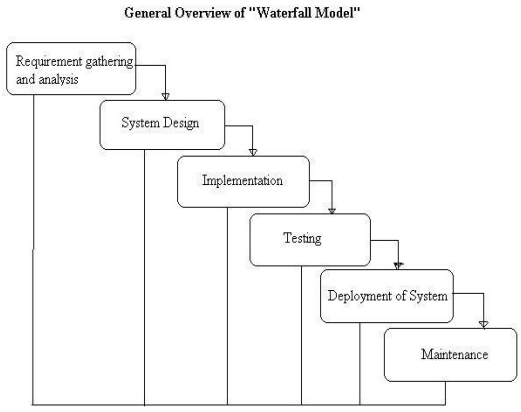

Fig. 2 Waterfall model

B. Prototyping Model

The prototyping methodology makes the use of: 1) developments in information technology, namely, advanced application generators that allow for fast and easy development of the software prototypes and 2) active participation in the development process by customers and users capable of examining and evaluating prototypes.

When applying the prototyping methodology, the future users of the system are required to comment on the various versions of the software prototypes prepared by the developers. In response to customer and user comments, the developers correct the prototype and add parts to the system on the way to presenting the next generation of the software for user evaluation. This process is repeated till the prototyping goal is achieved or the software system is completed [3]. A typical application of the prototyping methodology is shown in Fig. 3.

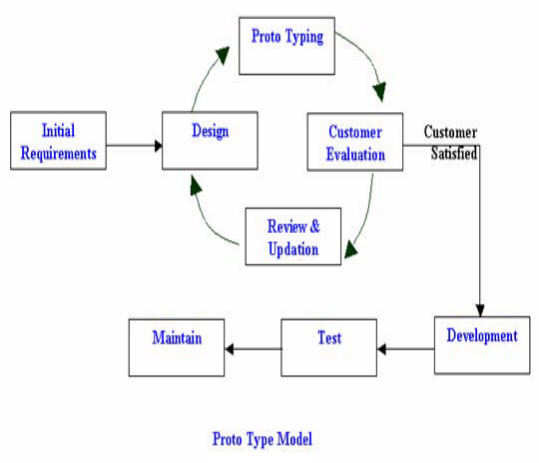

Fig. 3 Prototype model

C. Spiral Model

The spiral model, as revised by Boehm (1988, 1998), offers an improved methodology for overseeing the large and more complex development projects displaying higher prospects for failure, typical of many projects begun in the last two decades. It combines an iterative model that introduces and emphasizes risk analysis and customer participation into the major elements of the SDLC and the prototyping methodologies. According to the spiral model, software development is perceived to be an iterative process [3]. At each iteration, the following activities are performed:

- Planning
- Risk analysis and resolution
- Engineering activities according to the stage of the project: design, coding, testing, installation, and release
- Customer evaluation, including comments, changes, and additional requirements.

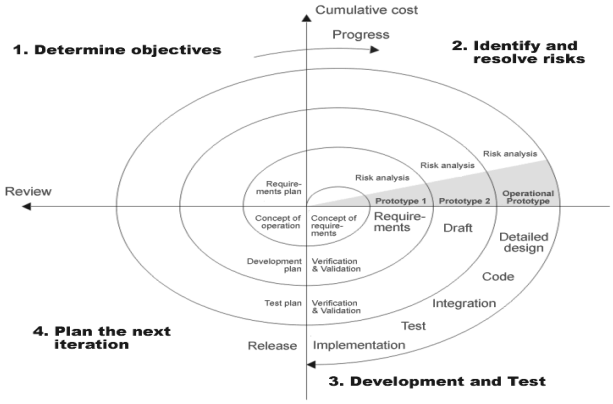

Fig. 4 Spiral model

D. V-Shaped Model

This model is like as the waterfall model, the V-shaped lifecycle is a sequential path of an execution of processes. Each phase must be completed before the next phase begins. Testing is emphasized in this model more than the waterfall model. The testing procedures are developed early in the lifecycle before any coding is done, during each of the phases preceding implementation. The test plan focuses on meeting the functionality specified in requirements gathering. The high-level design phase focuses on the system architecture and design. An integration test plan is created in this phase in order to test the pieces of the software systems ability to work together. However, the low-level design phase lies where the actual software components are designed, and unit tests are created in this phase as well. The implementation phase is again, where all coding takes place. Once coding is complete, the path of execution continues up to the right side of the V, where the test plans developed earlier are now put to use (see Fig. 5).

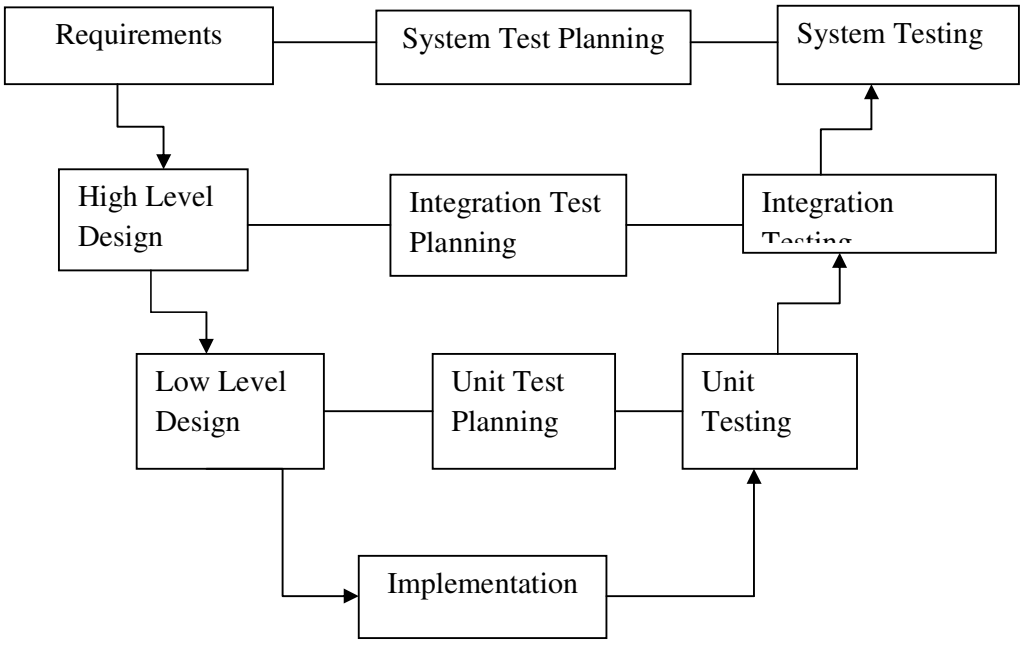

Fig. 5 V-shaped model

IV. WORKING OF ULTRA-FLOW MODEL

This model is used to overcome the limitation of the waterfall model. In this model, instead of freezing the requirements before coding or testing or design, a prototype is built to clearly understand the requirements. This prototype is built based on the current requirements. Through examining this prototype, the clients get better understanding of the features of the final product. The prototype may be a usable program, but is not suitable as the final software product, as shown in Fig. 6.

The following processes are involved in this approach:

- First, it stores the initial requirements for the customer's specification. Then, the next process it start the design of the product.
- After completing the design of the process, it will check the design using the ultra-flow module and also check the customer evaluation. If any changes are occurring in will change or modify using review and updating process.
- After the customer satisfaction, the document will be developed. After the development of coding, it must check the coding and evaluating from the customer. If any changes will modifying by the developers of customer needs.
- Then, the product is tested. Finally, it will maintain the product using an ultra-flow model.

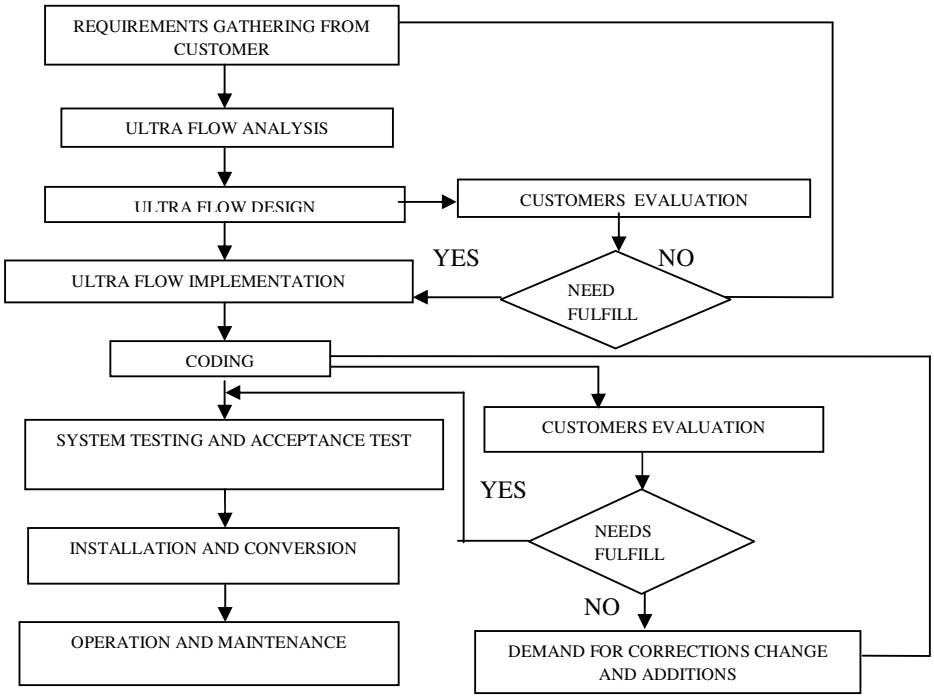

Fig. 6 Architecture of ultra-flow SDLC model

V. DIFFERENCE BETWEEN ULTRA and OTHER MODELS

This section tells how ultra-model is different from its base models.

A. Ultra Versus Spiral Model

- The main difference between the proposed model and spiral model is the size of products. The spiral model concentrates only on the large size products but the proposed model can develop product of any size.
- There were times when the spiral model was not able to provide a clear-cut picture of how software looks to the user. Ultra-model gives a picture for the user, and creates an interface that allows the user to work with.
- In the spiral model, prototyping was used only for risk management, but here it plays equally important role as the spiral model.

B. Ultra Versus Prototype Model

In the prototype model, we create a prototype, which is shown to the user if the user is satisfied the prototype, is thrown away and the software is created. Thus, the time taken for creating the prototype becomes a prime factor. But it is not the case in the proposed model.

C. Ultra

Ultra is the name of the model that here create a model that takes all the advantages of these models and leaves the negatives by doing this landed into a model, which we call as ultra. Ultra, if needed, considered as a solution for the flaws of the model. The basic definition of the spiral model said that it is suitable for creating large-sized products. Therefore, we decided to create a model that looked like the ultra-model but can be used to create even small-sized product. Ultra also concentrates mainly on changing desires of the user, so it uses the prototype technique to communicate with the user after each and every stage or phase of the software development.

VI. COMPARISON of DIFFERENT SDLC MODELS WITH NEW ULTRA-FLOW SDLC MODEL

As there are various models of the SDLC, each has its own advantages and disadvantages depending upon

which have to decide, which model should have to choose, as shown in Fig. 7. For instance, if the requirements are known before hand and well-understood and want a full control over the project at all time, we can use the waterfall model. The V-shaped model has a higher chance of success over the waterfall model due to the development of test plans during the lifecycle. It works well for small projects where requirements are easily understood. An incremental model is the heart of a cyclic software development process. It is easy to test and debug during a smaller iteration. The spiral model is good for large and mission critical projects where a high amount of risk analysis is required like launching of satellite.

RAD model is flexible and adaptable to changes as it incorporates the short development cycles, i.e., users see the RAD product quickly. It also involves user participation, thereby increasing the chances of early user community acceptance and realizes an overall reduction in project risk. Each model has different stages of planning and development. Finally, here, the proposed ultra-flow SDLC model is works good for a large number of projects and high risk analysed. . And it is easier to manage risk, because risky pieces are identified and handled during a process. It tries to fulfill the objective of the software engineering of developing a high-quality product within schedule and budget. The comparison of the different models is represented in Table 1 on the basis of certain features (see Fig. 7).

TABLE I

COMPARISON OF SDLC WITH ULTRA-FLOW SDLC

Features	Waterfall	Prototyping	Spiral	V-Shaped	Ultra-Flow
Requirements Specifications	Beginning	Frequently Changed	Beginning	Beginning	At Beginning
Cost	Low	High	Expensive	Expensive	Very Low
Guarantee Of Success	Less	Good	High	Good	High
Simplicity	Simple	Simple	Intermediate	Intermediate	Very Simple
Risk Involvement	High	Low	Medium	Low	Easily manage
Changes Incorporated	Difficult	Easy	Easy	Difficult	Very Easy
User Involvement	At Beginning	High	High	At the Beginning	At Beginning and Intermediate
Flexibility	Rigid	Flexible	Flexible	Little Flexible	Highly Flexible
Maintenance	Least Glamorous	Routine Maintenance	Typical	Least	Easily Maintained
Integrity & Security	Least	Weak	High	Robust	High
Documentation &Training Required	Vital	Weak	Yes	Yes	Well Documented
Time Frame	Long	Short	Long	According to project size	Very short
Reusability	Limited	Weak	High	Yes	High

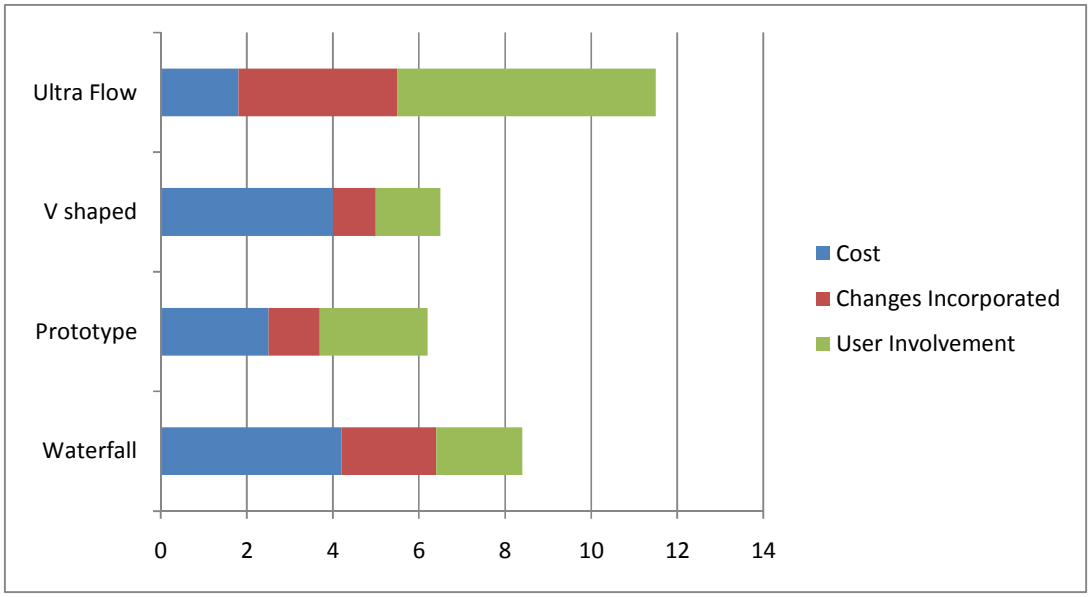

Fig. 7 Comparison graph of SDLC and ultra-flow SDLC

VII. CONCLUSION

SDLC is a methodology that governs the entire development process. In this paper, the various SDLC models are studied, such as waterfall, spiral, incremental, prototyping, and V-shaped models. The waterfall model provides the base for other development models. Here, comparing the top most five SDLC models and new ultra-flow SDLC model. In this paper, we show that the comparison between both the SDLC model and the ultra-flow SDLC model. The proposed work can be summarized as the creation of an approach, ultra-flow SDLC, to develop software more efficiently. The aim of software engineering is to develop software with a high quality within budget and schedule. The proposed plan tries to fulfil the objective of software engineering by showing the existing matching software as the prototype to the client for discovering the requirements efficiently from the client in order to estimate cost, schedule, and effort more accurately. Applying the suggested model to many projects is to ensure of its suitability and documentation to explain its mechanical work.

ACKNOWLEDGMENT

The authors would like to thank the anonymous referees for their valuable comments, which greatly improved the readability of this paper. G. R. Divya would also like to thank S. Kavitha, Assistant Professor, Department of Computer Science, who guided for my work, and also express my whole hearted thanks to my parents and friends for their encouragements to bring this work to a successful completion.

REFERENCES

[1] Sema, SonaMalhotra. "*Analysis and tabular comparison of popular SDLC models*", International Journal of Advance in Computer and Information Technology (IJACIT), July 2012.

[2] Vishwas Massey, Prof. K. J Satao. *"Comparing various SDLC models and the new proposed model on the basis of available methodology*", International Journal of Advanced Research in Computer Science and Software Engineering (IJARCSSE), volume 2, April 2012.

[3] Daniel Galin, *"Software Quality Assurance From Theory to Implementation"*.

[4] Laura C. Rodriguez Martinez, Manuel Mora, Francisco, J.Alvarez, "*A Descriptive/Comparative Study of the Evolution of Process Models of Software Development Life Cycles*", Proceedings of the 2009 Mexican International Conference on Computer Science IEEE Computer Society Washington,DC, USA, 2009.

[5] Sanjana Taya, Shaveta Gupta, "*Comparative Analysis of Software Development Life Cycle Models."*

[6] Kushwaha ety.al, "*Software Development Process and Associated Metrics - A Framework*", IEEE CNF.

[7] K. K. Aggarwal, Yogesh Singh, "*Software Engineering*", 3rd Edition.
[8] *"Software Development Life Cycle (SDLC) "*– the five common principles.htm
[9] *"Software Methodologies Advantages & disadvantages of various SDLC models"*.mht
[10] Naresh Kumar, A. S. Zadgaonkar, Abhinav Shukla "*Evolving a New Software Development Life Cycle Model SDLC-2013 with Client Satisfaction*" International Journal of Soft Computing and Engineering (IJSCE).