Z-SDLC Model: A New Model For Software Development Life Cycle (SDLC)

Syed Zaffar Iqbal, Muhammad Idrees
Alhamd Islamic University, Quetta-Pakistan

Abstract— **Software Development Life Cycle (SDLC) Models are the frameworks used to design, develop and test the software project. The SDLC models are set of procedures which are to be followed during the software development process. These SDLC models make sure that the software development is according to the needs of the client/customer and insure software will design within the given timeframe and budget.**

There are many SDLC models used during software development process. These models are also referred as Software Development Process Models (SDPM). Each process model follows a sequence of steps, in order to ensure success in the process of software development. We have different types of SDLC models. The SDLC models are waterfall model, iterative model, spiral model, V-model, agile model, RAD model and prototype model. Each of these models has its own weaknesses and strengths. In this paper I develop a new model called Z-SDLC model for software development that lays special emphasis on client/customer satisfaction and also tries to fulfil the objective of the Software Engineering for the development of high quality software product within timeframe/schedule and budget. The new proposed model is designed in such a way that it allows client/customer and software company to interact freely with each other in order to understand and implement requirements in a healthier way.

***Index Terms*— Client Satisfaction, SDLC Phases and Models, Software Development Process Model (SDPM), Z-SDLC Model.**

I. INTRODUCTION

Software Development Life Cycle (SDLC) is a procedure by which quality software project can be developed within timeframe/schedule and budget and also according to the client's expectations and prospects. SDLC ensures quality of software project. All software development processes models include various activities like requirements gathering, system feasibility, system analysis, system design, coding, testing, implementation and maintenance. The software company or the team of software developers have choice to select the SDLC model. Each of these models has its own weaknesses and strengths in different situations and circumstances. The challenge is to select which model should be good under certain conditions. Most of the present SDLC models have little attention towards client/customer satisfaction. Yes client/customer satisfaction matters. It matters not only to the client/customer but even more to the software company because it costs far less to retain a happy client/customer than it does to find a new client/customer. Satisfying client/customer is vital for staying in the market and in modern world of global competition.

Client/customer satisfaction is very necessary for the acceptance and delivery of the software project. Software projects fails in the absence of client/customer satisfaction. SDLC model must fulfill the requirements of the client/customer as per there expectations and even delight with the value of quality services.

II. SOFTWARE DEVELOPMENT LIFE CYCLE

Software Development Life Cycle (SDLC) is a process followed for a software project, within a software organization. It consists of a detailed plan describing how to develop, maintain, replace and alter or enhance specific software. The life cycle defines a methodology for improving the quality of software and the overall development process. SDLC is a process used by software industry to design, develop and test high quality software. The SDLC aims to produce a high quality software that meets or exceeds client/customer expectations, reaches completion within times and cost estimates. It is also called as Software development process. The SDLC is a framework defining tasks performed at each step in the software development process. ISO/IEC 12207 is an international standard for software life-cycle processes. It aims to be the standard that defines all the tasks required for developing and maintaining software [1], [12].

The following figure is a graphical representation of the various stages of a typical SDLC.

A. Planning and Requirement Analysis
B. Defining Requirements
C. Designing the Product Architecture
D. Building or Developing the Product
E. Testing
F. Deployment
G. Maintenance

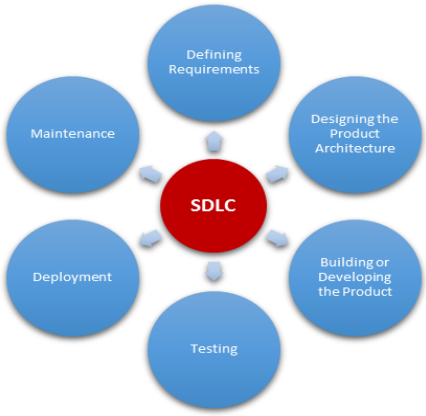

Fig 1: Software Development Life Cycle

A. Planning and Requirement Analysis

Requirement analysis is the most important and fundamental stage in SDLC. It is performed by the senior members of the team with inputs from the customer, the sales department, market surveys and domain experts in the industry. This

information is then used to plan the basic project approach and to conduct product feasibility study in the economical, operational and technical areas. Planning for the quality assurance requirements and identification of the risks associated with the project is also done in the planning stage. The outcome of the technical feasibility study is to define the various technical approaches that can be followed to implement the project successfully with minimum risks [12].

B. Defining Requirements

Once the requirement analysis is done the next step is to clearly define and document the product requirements and get them approved from the customer or the market analysts. This is done through Software Requirement Specification (SRS). SRS document which consists of all the product requirements to be designed and developed during the project life cycle [12].

C. Designing the Product Architecture

SRS is the reference for product architects to come out with the best architecture for the product to be developed. Based on the requirements specified in SRS, usually more than one design approach for the product architecture is proposed and documented in a Design Document Specification (DDS). This DDS is reviewed by all the important stakeholders and based on various parameters as risk assessment, product robustness, design modularity, budget and time constraints, the best design approach is selected for the product [12].

D. Building or Developing the Product

In this stage of SDLC the actual development starts and the product is built. The programming code is generated as per DDS. If the design is performed in a detailed and organized manner, code generation can be accomplished without much hassle. Developers have to follow the coding guidelines defined by their organization and programming tools like compilers, interpreters, debuggers etc are used to generate the code. Different high level programming languages such as C, C++, Pascal, Java and PHP are used for coding.

The programming language is chosen with respect to the type of software being developed.

E. Testing the Product

This stage is usually a subset of all the stages as in the modern SDLC models, the testing activities are mostly involved in all the stages of SDLC. However this stage refers to the testing only stage of the product where products defects are reported, tracked, fixed and retested, until the product reaches the quality standards defined in the SRS [12].

F. Deployment in the Market

Once the product is tested and ready to be deployed it is released formally in the appropriate market. Sometime product deployment happens in stages as per the organizations business strategy. The product may first be released in a limited segment and tested in the real business environment User acceptance testing (UAT).

G. Maintenance

Then based on the feedback, the product may be released as it is or with suggested enhancements in the targeting market segment. After the product is released in the market, its maintenance is done for the existing customer base [12].

III. SDLC Models

There are various software development life cycle models defined and designed which are followed during software development process. These models are also referred as Software Development Process Models (SDPM). Each process model follows a series of steps unique to its type, in order to ensure success in process of software development. Following are the most important and popular SDLC models followed in the industry:

A. Waterfall Model
B. RAD Model
C. Prototype Model

A. Waterfall Model

The Waterfall Model was first process model to be introduced provided by Winston W. Royce in 1970. It is also referred to as a linear-sequential life cycle model. It is very simple to understand and use. In a waterfall model, each phase must be completed before the next phase can begin and there is no overlapping in the phases. The waterfall model illustrates the software development process in a linear sequential flow; hence it is also referred to as a linear-sequential life cycle model. This means that any phase in the development process begins only if the previous phase is complete. In waterfall model phases do not overlap.

In the waterfall approach, the whole process of software development is divided into separate phases. In waterfall model, typically, the outcome of one phase acts as the input for the next phase sequentially. Following is a diagrammatic representation of different phases of waterfall model.

1. Requirement Analysis
2. System Analysis
3. Implementation
4. Verification
5. Maintenance

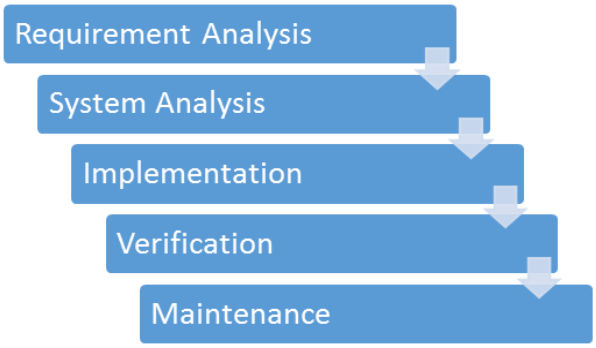

Fig 2: Waterfall Model

Every software developed is different and requires a suitable SDLC approach to be followed based on the internal and external factors. Some situations where the use of Waterfall model is most appropriate are:

- Requirements are very well documented, clear and fixed
- Product definition is stable
- Technology is understood and is not dynamic
- There are no ambiguous requirements
- Ample resources with required expertise are available to support the product

Pros

1) Simple and easy to understand
2) Easy to manage
3) Clearly defined stages
4) Well understood milestones
5) Easy to arrange tasks

Cons

1) High amounts of risk and uncertainty
2) It is difficult to measure progress within stages
3) Cannot accommodate changing requirements
4) Adjusting scope during the life cycle can end a project
5) No working software is produced until late during the life cycle [13].

B. Rapid Application Development (RAD) Model

The Rapid Application Development (RAD) model is based on prototyping and iterative development with no specific planning involved. The process of writing the software itself involves the planning required for developing the product. RAD focuses on gathering customer requirements through workshops or focus groups, early testing of the prototypes by the customer using iterative concept, reuse of the existing prototypes (components), continuous integration and rapid delivery.

Following image illustrates the RAD Model:

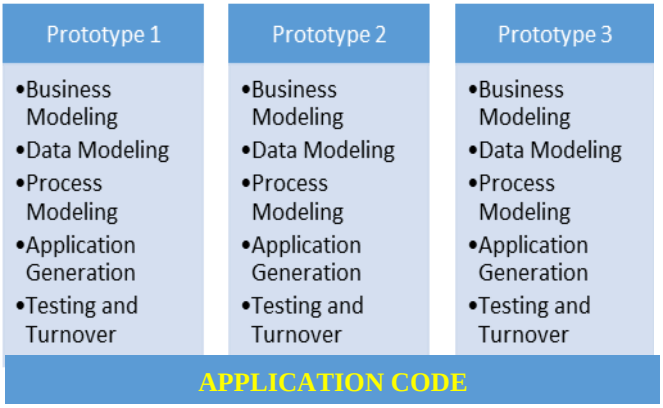

Fig 3: RAD Model

RAD model can be applied successfully to the projects in which clear modularization is possible. If the project cannot be broken into modules, RAD may fail.

Following are the typical scenarios where RAD can be used:

- RAD should be used only when a system can be modularized to be delivered in incremental manner.
- It should be used if there's high availability of designers for modeling
- It should be used only if the budget permits use of automated code generating tools
- RAD SDLC model should be chosen only if domain experts are available with relevant business knowledge
- Should be used where the requirements change during the course of the project and working prototypes are to be presented to customer in small iterations of 2-3 months

Pros

1) Progress can be measured
2) Reduced development time
3) Increases reusability of components
4) Quick initial reviews occur
5) Encourages customer feedback

Cons

1) Only system that can be modularized can be built using RAD.
2) Requires highly skilled developers/designers
3) High dependency on modeling skills
4) Inapplicable to cheaper projects as cost of modeling and automated code generation is very high
5) Requires user involvement throughout the life cycle [14].

C. Prototype Model

The Software Prototyping refers to building software application prototypes which display the functionality of the product under development but may not actually hold the exact logic of the original software. Software prototyping is becoming very popular as a software development model, as it enables to understand customer requirements at an early stage of development. It helps get valuable feedback from the customer and helps software designers and developers understand about what exactly is expected from the product under development.

Prototype is a working model of software with some limited functionality. The prototype does not always hold the exact logic used in the actual software application and is an extra effort to be considered under effort estimation. Prototyping is used to allow the users evaluate developer proposals and try them out before implementation. It also helps understand the requirements which are user specific and may not have been considered by the developer during product design.

Following is the stepwise approach to design a software prototype:

- Basic Requirement Identification
- Developing the initial Prototype
- Review of the Prototype
- Revise and enhance the Prototype

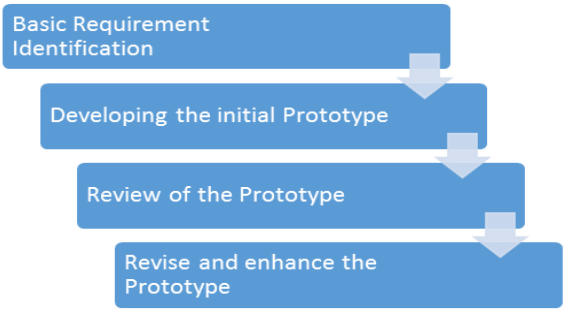

Fig 4: Prototype Model

Software Prototyping is most useful in development of

systems having high level of user interactions such as online systems. Systems which need users to fill out forms or go through various screens before data is processed can use prototyping very effectively to give the exact look and feel even before the actual software is developed. Software that involves too much of data processing and most of the functionality is internal with very little user interface does not usually benefit from prototyping. Prototype development could be an extra overhead in such projects and may need lot of extra efforts.

Pros

1) Increased user involvement in the product even before implementation
2) Reduces time and cost as the defects can be detected much earlier.
3) Quicker user feedback is available leading to better solutions.
4) Missing functionality can be identified easily
5) Confusing or difficult functions can be identified

Cons

1) Risk of insufficient requirement analysis owing to too much dependency on prototype
2) Users may get confused in the prototypes and actual systems
3) Practically, this methodology may increase the complexity of the system as scope of the system may expand beyond original plans
4) Developers may try to reuse the existing prototypes to build the actual system, even when it's not technically feasible
5) The effort invested in building prototypes may be too much if not monitored properly [15].

IV. New Proposed Z-SDLC Model

The new Z-SDLC model is planned in such a way that it allows software company and client to freely interact with each other in order to understand the requirements of software project in a good way to develop a good quality software within a given timeframe and budget.

SDLC process model start with the client's requirements so the proposed model tries to find every requirements like functional requirements, non-functional requirements and user requirements of the client/customer. It helps in developing a good quality of software product that satisfies the client/customer needs. The scope of computer based system products, client satisfaction is very much dependent on how system development process works to build operational product that satisfy the client's need and also related with the expected requirements.

Finally, client satisfaction depends upon the good understanding about the client needs and associated user requirements for a better software product and the capability to connect those requirements to the software company. In addition, client satisfaction and confidence depends upon the level of product guarantee offered throughout the SDLC. Understanding to the requirements problems inevitably leads to poor client/customer and software company relationship, unnecessary re-work and exceed the budget and timeframe. The client satisfaction is totally depended on client needs for this reason Z-SDLC model focus on the initial phases.

My proposed Z-SDLC model include the following:

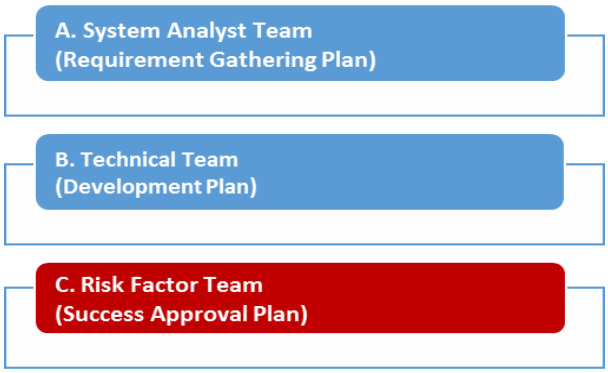

Fig 5: Z-SDLC Model

A. SYSTEM ANALYST TEAM (Requirement Gathering Plan)

The System Analyst team have a sufficient knowledge of computer science, software engineering, software development processes, software applications, operating system, as well as domain knowledge like various business functions to be performed. The system analyst team coordinates with the risk factor team and technical team. System Analyst team deals with the client for Identify Problem, Breakdown Requirements, Make a Prototype, Finalize the Requirements, Feasibility Study, Approval of SRS and any ambiguity of client is also discuss and solved by the system analyst team.

1. Identify the Problem
2. Identify the Requirements
3. Breakdown Requirements
4. Finalize the Requirements
5. Feasibility Study
6. Approval of SRS Document
7. Make a Prototype

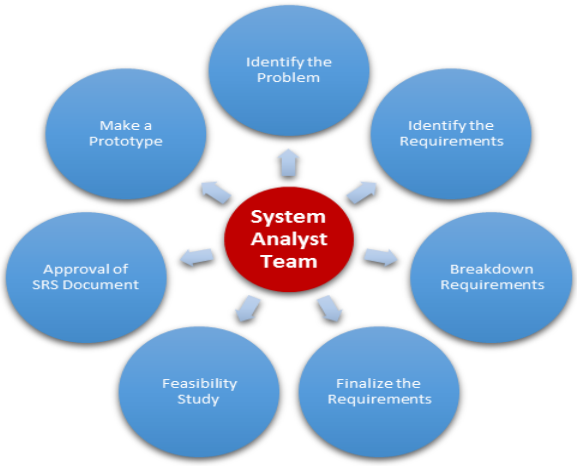

Fig 6: System Analyst Team

The system analyst team identifies the requirements and divide all the requirements into different features and then

gets the existing software whose requirements match with the current proposed software requirements.

The system analyst team breakdown the requirements into two parts:

1. Developed Requirement
2. Non-Developed Requirements

Developed requirements are those requirements which are already implemented in some existing software. Non-developed requirements are those requirements which are not implemented by any of the existing software, mean these are the fresh features and need to develop.

After finalizing requirements the system analyst team is now going quickly start work on Software Requirement Specification (SRS) Document and feasibility study report like estimates the budget, timeframe and effort which are mandatory for the development of the software product, as the SRS document is complete the system analyst team passes this SRS document (the final requirements document) to the technical team as well as to the risk factor team.

The system analyst team again with the collaboration of technical team can make a software prototype and show this prototype to the client/customer for approval. Client/customer also gave his/her reviews, suggestions and feedback to the system analyst team in order to change of the requirement(s). The result of breaking requirements and showing existing software as dummy to the client/customer is that the client/customer gets the feel of graphics, functionality and features of product. It helps both client/customer and software company to identify, discovers and implements the requirements efficiently. Now these requirement are passed to the technical team for final design and development of the software product.

If client wants any change(s) in the final requirements during the process, then system analyst firstly checks whether it can be implemented or not and what are impacts of change on the whole process in terms of cost, timeframe and effort. If change(s) is possible and its impact is little or very less then change(s) will be accommodated.

B. TECHNICAL TEAM (Development Plan)

Technical team is an expert team and its team members are updated with new technologies and new software products. It is a technically expert team. This team interacts with system analyst team during its working. Technical team studies the SRS document (the requirements document) received from the system analyst team which in turn get these requirements from the client/customer.

The member of technical team is full of skills and interacts with system analyst team. Technical team works on non-developed requirements. This team studies the feasibility of requirements to check whether these are technically possible or not. This team also identifies and resolves the various risk associated with the implementation of non-developed requirements with the collaboration of risk factor team. After feasibility study and risk analysis the technical team finally verify the final SRS document, check the prototype provided to the client/customer and start work on the following phases, i.e. Designing, Coding, Testing, Implementation, Maintenance each of these phase also followed by validation process.

1. Designing
2. Coding
3. Testing
4. Implementation
5. Feedback and Maintenance

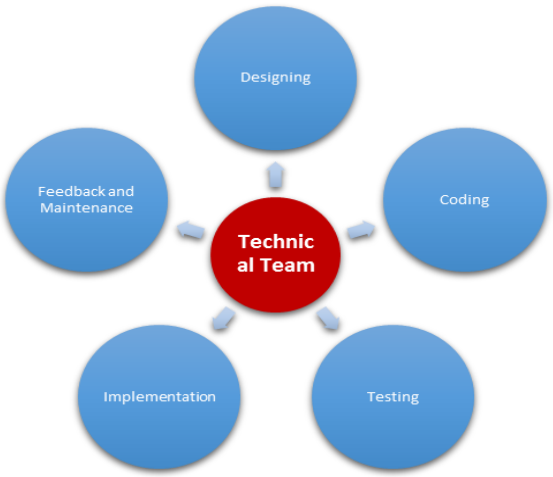

Fig 7: Technical Team

C. RISK FACTOR TEAM (Success Approval Plan)

What is Risk?

Risk are future uncertain events with a probability of occurrence and a potential for loss. Risk identification and management are the main concerns in every software project. Effective analysis of software risks will help to effective planning and assignments of work [16].

Risks are identified, classified and managed before actual execution of program. These risks are classified in different categories.

- Schedule Risk
- Budget Risk
- Operational Risks
- Technical risks
- Programmatic Risks

Fig 8: Risk Factor Team

The risk management in SDLC model is started as per the client's requirements. In the beginning, these requirements are in the mind of the client. The system analyst team by using

a software development model has to identify, discover, understand and fulfill the requirements of the client in order to satisfy the client. The requirement phase of the Software Development Life Cycle transforms the idea in the mind of the client into a formal document known as Software Requirement Specification (SRS). The quality of the SRS impacts client satisfaction, system validation, quality of final software, software development cost and schedule. A high quality SRS is necessary to produce the high quality software. A system analyst team and technical team are fail to satisfy the client because of the three reasons: If software fail to discover requirements, If software fail to implement these requirements and If software changes rapidly.

The client usually does not understand Software or the Software Development Life Cycle (SDLC) and the software company often does not understand the clients problem and application area. But the Z-SDLC model permits the client and software company to interact freely with each other for the better understating of the problem and in identifying the requirements feature. Moreover, In Z-SDLC model the requirements are breakdowns into two parts i.e. developed requirements and non-developed requirements. For developed requirements the present most similar most software modules with matching the client's requirements are shown to the client so that the client can easily identify and express the requirement to the system analyst team. If the identified requirements are not fulfilled or implemented accurately then it leads to disappointment and dissatisfaction of client. As, In Z-SDLC model requirements are breakdown into developed and non-developed requirements. In the base of non-developed requirements the system analyst team can make a feasibility study report and SRS document. This risk factor team also identifies the various risk associated with the requirements, so that the decision can be taken about the deployment/implementation of requirements. If some non-developed requirements are not technical possible then the client is informed about this during the early stage so that the client does accept the system with satisfaction and have no objection.

We know that the software requirements are frequently changes. Some of the changes are expected due to changing requests. But many changes come because the requirements are not properly taken and analyzed and not enough effort was used to validate the requirements. But Z-SDLC model is designed in such a way that the software company can focus on proper requirement gathering. It is estimated 20% to 40% of total development effort [17], [18] in a software project is due to rework much of which occurs due to change in requirements. The cost of the requirement phase is normally about 6% of the total project cost [19].

Consider the complete software project the total effort requirement is estimated to be 40 person-months. For this project, the requirement phase consumed 3 person months. If by spending a 50% effort in the requirement phase, we reduce the total requirement change required by 33% then the total effort due to rework will reduce from 10 to 20 person months to 6 to 12 person months, resulting in total saving of 5 to 11 person months, i.e. a saving of 10 to 20% of total cost [4].

The following details explain how the new proposed Z-SDLC model is applicable. The details given below explain how the new suggested Z-SDLC model has the command of satisfying the client/customer.

V. Deployment Software

Software is developed for the clinical laboratory named as Logical LIMS (Laboratory Information and Management System). There are various SDLC models for the software development but I choose Waterfall model, Prototype Model, Incremental Model and my new designed Z-SDLC model for software development and comparing the working of existing models with the Z-SDLC.

Let's check software developed by traditional SDLC models.

A. Software Development by Waterfall model

The waterfall model is a linear sequential model. We considered the requirements, check them and moved towards the designing phase followed by the coding and testing phases for software development named as LIMS-1, but the LIMS-1 was not accepted by Mr. Haji Asif, Technologist Lab In-charge (the client) because they were not satisfied. As the client want to change it in terms of graphics, functionality and features the waterfall model does not allow to change after completing the requirements so, it fails to convey client about the software product.

Fig 9: Software development by waterfall model

B. Software Development by Prototype model

The prototype model build prototype to give feel of the proposed software to the client. As we already have Logical LIMS requirement so, we build prototype and showed it to the client. After client's feedback, we changed it and again showed it to the client. After building and showing three prototypes, client finalized the requirements and we passed these final requirements to next phases for software development and named it as LIMS-2. Finally LIMS-2 was delivered to the client. But building prototype affects cost, schedule and effort which get exceeded.

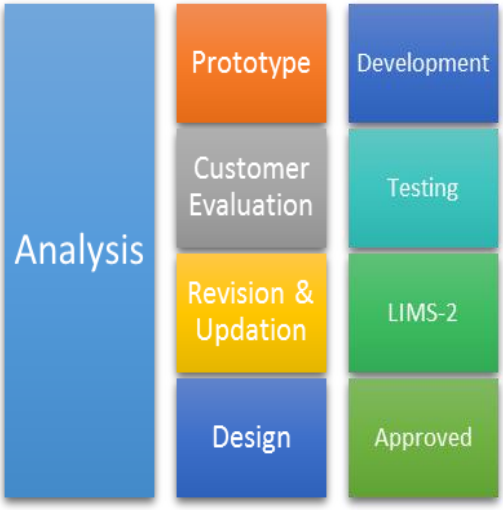

Fig 10: Software development by prototype model

C. Software Development by Incremental model

The Incremental model is an evolution of waterfall model which has number of iterations and after each iteration, we get a working product. Initially we analyzed the requirements and go through the designing, coding and testing phases and released the first iteration. The first iteration working product was given to the client and after getting client's feedback we changed it and released the product of the second iteration. With each iteration functionality and feature of the product get enhanced and after three iterations we got LIMS-3 which was finally delivered to the client. Incremental model reduce the cost of building prototype because instead of building prototype it accommodate the changes into the working product but due to iterations, schedule get exceeded which in turn effect the cost and effort.

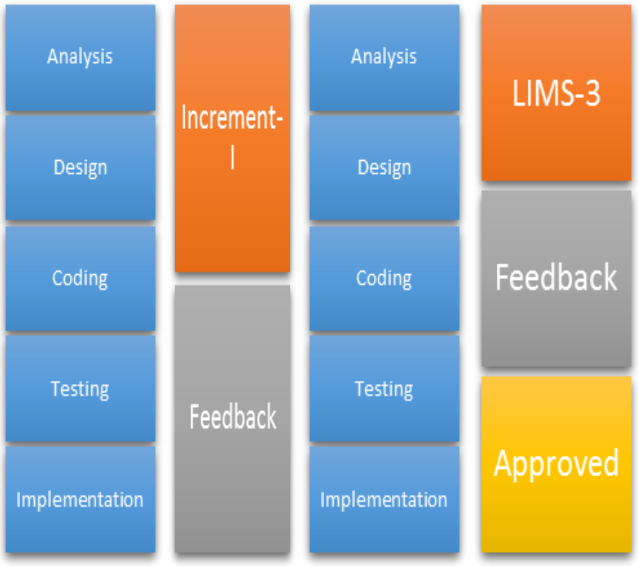

Fig 11: Software development by incremental model

D. Software Development by Z-SDLC model

Z-SDLC model is a new model for the software development. The striking feature of this model is the client satisfaction. Firstly, system analyst team deal with Mr. Haji Asif, Technologist Lab In-charge (the client) to discover the problem and requirements. After discovering the problem the system analyst team breakdown this requirement into developed requirements and non-developed requirements.

Accordingly, the system analyst team have now a breakdown of available requirements i.e. developed and non-developed requirements but in this case there was no any non-developed requirements. The system analyst team finalize the requirement and also start work on the feasibility study and SRS document.

The system analyst team with the collaboration of technical team analyzed the available requirements provided by the system analyst that is present in SRS document for the proposed system and searched the most matching software for them. He found three such software whose requirements matched with the proposed software's requirements previously build for laboratories.

Now the system analyst team showed the software to the client so that the client got the feel of proposed software and also identifies the undiscovered requirements and gave his suggestion and feedback to the system analyst team. The system analyst team again passed these suggestions to the technical team and the process goes on until the client finalized the requirements. System analyst team passed final requirements to the risk factor team for the risk analysis and the requirement validation.

After validation and resolving various risk associated with the final requirements, these final requirements were passed to technical team for final software product. The technical team start work on the following phases i.e. designing, coding, testing, implementation and maintenance followed by the validation process to develop the final product named as Logical LIMS. Logical LIMS was approved by the client because it satisfied the client's requirements within budget and given timeframe because budget and timeframe were not disturbed or affected due to various increments or by building prototype. Finally, User Acceptance Test is signed with the client.

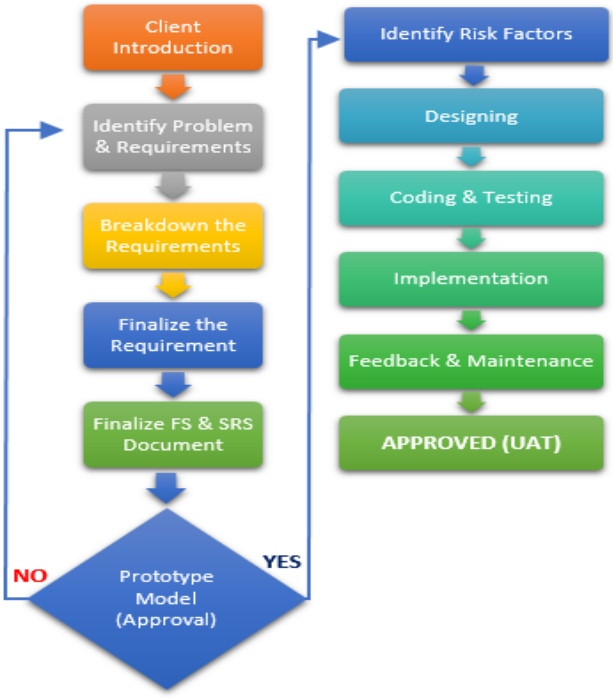

Fig 12: Software development by Z-SDLC model

VI. Comparison of Z-SDLC with other Models

Let's check the comparison of other models with new Z-SDLC model.

Table 1A: Comparison of Z-SDLC model with other models

Features	Waterfall	Prototype
Requirement	Cleared	Not Cleared
Software Cost	Low	High
Schedule	On time	Not on time
Risk Factor	High	Low
User Involvement	Low	High
Initial Product Feel	No	Yes
Client Satisfaction	Low	Medium
Guarantee of Success	Low	Medium

Table 1B: Comparison of Z-SDLC model with other models

Features	Incremental	Z-SDLC
Requirement	Cleared	Cleared
Software Cost	Medium	Low
Schedule	Not on time	On time
Risk Factor	Medium	Low
User Involvement	High	High
Initial Product Feel	No	Yes
Client Satisfaction	Medium	High
Guarantee of Success	Medium	High

VII. CONCLUSIONS

In this research paper various models like a waterfall, RAD and prototype models have been considered and various topographies like requirement specification, cost, risk factor, user involvement, success rate, simplicity is analyzed. Each model has its own pros and cons. In the requirement gathering phase the software developer can select the suitable software development life cycle model according to the needs. My suggested work can be concise as the construction of the new Z-SDLC model for more efficient software development. The goal of the Software Engineers are to grow the software industry at bigger stage and want to make software with high quality within budget and schedule. My suggested plan tries to accomplish the objective of Software Engineers by showing existing matching software as a prototype to the client/customer for discovering the requirements efficiently from the client/customer in order to approximation of cost, time frame, schedule, work and effort more accurately and precisely.

References

[1] Roger S. Pressman, Ph.D. "Software Engineering a Practitioner's Approach, Fifth Edition: McGraw-Hill Higher Education, 2001, pp. 53-193

[2] Roger S. Pressman, Ph.D. "Software Engineering a Practitioner's Approach, Fifth Edition: McGraw-Hill Higher Education, 2001, pp. 245-507

[3] Ian Sommerville, "Software Engineering", Addison-Wesley, 2007

[4] The Software Development Life Cycle (SDLC) for Database Applications, First Edition, Digital Publication LLC 2005, pp. 14-21.

[5] Rod Stephens, "Beginning Software Engineering", John Wiley & Sons, 23-Mar-2015

[6] Elvis Foster, "Software Engineering: A Methodical Approach", Apress, 16-Dec-2014

[7] Shari Lawrence Pfleeger, Joanne M. Atlee, "Software Engineering: Theory and Practice", Prentice Hall, 2010

[8] RAJIB MALL, "FUNDAMENTALS OF SOFTWARE ENGINEERING", PHI Learning Pvt. Ltd., 18-May-2009

[9] Claes Wohlin, Per Runeson, Martin Höst, Magnus C. Ohlsson, Björn Regnell, Anders Wesslén, "Experimentation in Software Engineering", Springer Science & Business Media, 16-Jun-2012

[10] Hossein Hassani, How to do the Final Year Projects A Practical Guideline for Computer Science and IT Students, 2012. Available: www.bookboon.com

[11] Shari Lawrence Pfleeger and Joanne M. Atle, "Software Engineering theory and practice" Fourth Edition Pearson Publishing

[12] SDLC Overview Tutorials Point Simple Easy Learning. Available: https://www.tutorialspoint.com/sdlc/sdlc_overview.htm

[13] SDLC Overview Tutorials Point Simple Easy Learning. Available: https://www.tutorialspoint.com/sdlc/sdlc_waterfall_model.htm

[14] SDLC Overview Tutorials Point Simple Easy Learning. Available: https://www.tutorialspoint.com/sdlc/sdlc_rad_model.htm

[15] SDLC Overview Tutorials Point Simple Easy Learning. Available: https://www.tutorialspoint.com/sdlc/sdlc_software_prototyping.htm

[16] Types of Risks in Software Projects, Basics of Software testing, Quality assurance, Test strategy | Last Updated: "December 14, 2016". Available: http://www.softwaretestinghelp.com/types-of-risks-in-software-projects/

[17] A Brief Summary of the Original COCOMO Model. Available: http://www.mhhe.com/engcs/compsci/pressman/information/olc/COCOMO.html

[18] M. A. Parthasarathy, "Practical Software Estimation: Size, Effort, and Scheduling of Projects" InformIT, Jan 25, 2008. Available: http://www.informit.com/articles/article.aspx?p=705249&seqNum=4

[19] Sarah Afzal Safavi and Maqbool Uddin Shaikh COMSATS Institute Pakistan, "Effort Estimation Model for each Phase of Software Development Life Cycle", Source Title: Handbook of Research on E-Services 2011 Available: http://www.igi-global.com/chapter/effort-estimation-model-each-phase/46269

[20] PK.Ragunath, S.Velmourougan (1, January 2010) "Evolving A New Model (SDLC Model-2010) For Software Development Life Cycle (SDLC)", IJCSNS International Journal of Computer Science and Network Security, VOL.10 No.1.

[21] Shubham Dwivedi (2, February 2016) Software Development Life Cycle Models - A Comparative analysis International Journal of Advanced Research in Computer and Communication Engineering, Vol. 5, Issue 2.

[22] Naresh Kumar, A. S. Zadgaonkar, Abhinav Shukla (March 2013) Evolving a New Software Development Life Cycle Model SDLC-2013 with Client Satisfaction, International Journal of Soft Computing and Engineering (IJSCE) ISSN: 2231-2307, Volume-3, Issue-1.

[23] Clifford J. Berg "Software Development Life Cycle", Prentice Hall PTR New Jersey 1998 Available: http://www.shazsoftware.com/bookextracts/software_development_life_cycle.php

[24] Vishwas Massey, Prof. K. J Satao, Evolving a new Software Development Life Cycle Model (SDLC) incorporated with release management, International Journal of Engineering and Advanced Technology (IJEAT), volume-I, Aril 2012,

[25] 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,

[26] Seema , SonaMalhotra , "comparative analysis of popular SDLC models ", International Journal of advances in computing and information technology, July 2012.

Syed Zaffar Iqbal works as a Lecturer at Alhmad Islamic University, Quetta-Pakistan. He received his MCS degree from University of Balochistan in 2010. His research interests include are software engineering, database management system, programming and development. He is a supervisor of final year projects at his university and ICT R&D Funded projects. He is the author of computer science books. He is also running a software company named as Logical Creations as Managing Director. Contact: 0092 343 8244401

Muhammad Idrees works as a Lecturer in Mathematics at Government Boys Degree College, Nushki Balochistan. He received his MSc Mathematics from University of the Punjab, MCS from Virtual University of Pakistan and M.Phil Scholar in Mathematics at University of Balochistan, Quetta. His research interests include are Fuzzy Logic, Fuzzy Algebras and Theoretical Computer Science. He is also an advisor in Logical Creations.
Contact: 0092 321 8040866