Evolving a New Software Development Life Cycle Model SDLC-2013 with Client Satisfaction

Naresh Kumar, A. S. Zadgaonkar, Abhinav Shukla

Abstract— In the era of 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 is capable to address the issues of client satisfaction. In this paper we develop a new model (SDLC-2013) for software development that lays special emphasis on client satisfaction and also tries to fulfil the objective of the Software Engineering of developing high quality product within schedule and budget. The new proposed model is designed in such a way that it allows client and developer to interact freely with each other in order to understand and implement requirements in a better way.

Index Terms— SDLC, Software Development, SDLC Phases, SDLC-2013 Model, Client Satisfaction

I. INTRODUCTION

Software Engineering is a discipline whose aim is the production of quality software, software that is delivered on time, within budget and that satisfies its requirements [1]. Software Engineering is the area which is constantly growing. It is very interesting subject to learn as all the software development industry based on this specified area. There exist various models to develop software. But most of the existing Software Development Models pay less or very little attention towards client satisfaction. Client satisfaction matters. It matters not only to the client, but even more to the developer because it costs far less to retain a happy client 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 very necessary for the acceptance and delivery of the software product. Software projects fails in the absence of client satisfaction. Software Development Model must satisfied and even delight client with the value of software products and services.

II. WHAT IS SDLC

Software Development Life Cycle is a process to develop software. This process is divided into some phases such as Requirement Analysis, Design, Coding, Testing, Installation and Maintenance. All these activities are carried out in different ways as per the client's need. Each way is known as a Software Development Life Cycle Model. Every system must go through these phases whether it is small scale or large scale [2],[3].

A. *Requirement Analysis:*

Requirement analysis is the initial phase of the Software Development Life Cycle. The goal of this phase is to understand the client's requirements and to document them properly. The emphasis in requirement analysis is an identifying what is needed from the system. It is most crucial phase in Software Development Life Cycle. The output of requirement analysis is Software Requirement Specification (SRS) [4],[5].

B. *Design:*

It is the first step to move from the problem domain towards the solution domain. It is the most creative phase in Software Development Life Cycle. The goal of this phase is to transform the requirement specification into structure [1]. The output of this phase is Software Design Document (SDD).

C. *Coding:*

In this phase Software Design Document (SDD) is converted into code by using some programming language. It is the logical phase of the Software Development Life Cycle. 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 [1],[5],[6].

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. *Waterfall Model*

The Waterfall model is provided by Winston W. Royce in 1970. In this model whole work is done in linear fashion. Entire work is divided into five different phases. All the phases are cascaded to each other so that second phase is started as and when defined set of goals are achieved for the first phase and it is signed off, so it is named as "Waterfall Model".

Manuscript received March 2013.

Naresh Kumar, M. Phil (IT) Scholar, Information Technology Department, Dr. C. V. Raman university, Bilaspur, India.
Dr. A. S. Zadgaonkar, Vice Chancellor, Dr. C. V. Raman University, Bilaspur, India.
Abhinav Shukla, Assistant Professors & HOD (IT) Department, Dr. C. V. Raman University, Bilaspur, India.

Requirements are very well understood before start working. In the un-modifiable waterfall model progress flows from the top to the bottom, like a waterfall. The waterfall development model has its origin in the manufacturing and construction industry; highly structured physical environment in which change in the requirements are prohibitively costly. Since no formal software development methodologies existed at the time, this hardware model was simply adapted for software development [2],[5],[7].

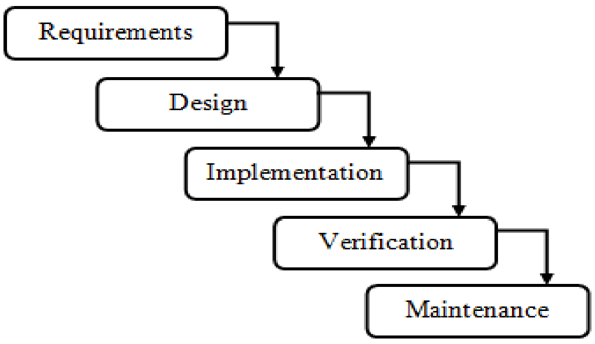

Figure1. Waterfall Model

Advantages
1) Simple and easy to use.
2) Easy to arrange tasks and clearly defined stages.
3) Requirement should be clear before going to next phases.
4) Each phase of development proceeds in linear order without any overlapping.
5) Works well for projects where requirements are well understood.

Disadvantages
1) Users can judge quality only at the end.
2) It does not allow changes as per client's requirement.
3) High amount of risk and uncertainty.
4) User doesn't get the feel of the product before delivery.
5) It follows the "Big bang" approach- the entire software is delivered in one shot at the end.

B. *Prototype Model*

In this model prototype is built as per the client requirements. Instead of freezing the requirement before a design or coding can proceed. The purpose of a prototype is to allow users of the software to evaluate proposals for the design of the eventual product by actually trying them out, rather than having to interpret and evaluate the design based on descriptions. Prototyping has several benefits: The software designer and developer can obtain feedback from the users early in the project. The client and the developer can compare if the software made matches the software specification, according to which the software program is built. It also allows the software engineer some insight into the accuracy of initial project estimates and whether the deadlines and milestones proposed can be successfully met. A prototype model is not a standalone, complete development methodology, but rather an approach to handle selected part of a larger, more traditional development methodology. It attempts to reduce inherent project risk by breaking a project into smaller segments and providing more ease-of-change during the development process. User is involved throughout the development process, which increases the likelihood of user acceptance of the final implementation. Small-scale mock-ups of the system are developed following an iterative modification process until the prototype evolves to meet the user's requirement. While most prototypes are developed with the expectation that they will be discarded, it is possible in some cases to evolve from prototype to working system. A basic understanding of the fundamental business problem is necessary to avoid solving the wrong problem [3],[8],[9].

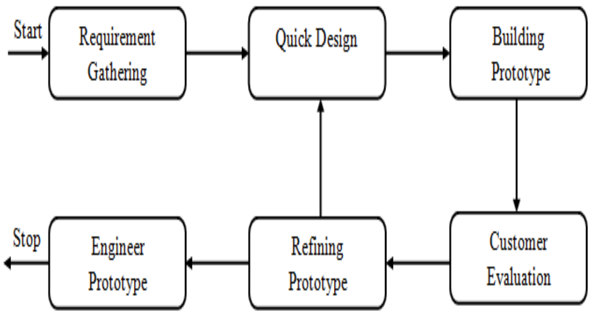

Figure2. Prototype Model

Advantages
1) Users are actively involved in the development
2) When prototype Model is shown to the user, he gets a proper clarity about his requirements. And feel the functionality of the software, so can suggest the changes and modifications.
3) It reduces risk of failure, as potential risks can be identified early and steps can be taken to remove that risk.
4) The customer does not need to wait long for working software.

Disadvantages
1) Wastage of Time and money to build prototype, if client not satisfied.
2) Too many changes can disturb the rhythm of the developer team.
3) Long term procedure.
4) It follows the "Quick and dirty" approach- the prototype is through away after showing to the client.

C. *Incremental Model*

It is the evolution of waterfall model. In this model all the activities are repeatable. Multiple activities run parallel. The phases of waterfall model are employed in such a manner that the result of the increment is used back as the input for the next increment. Thus with each increment there are some clients feedback that is used for getting the next incremental product. Thus with each ongoing increment the functionality of the core product gets enhanced. Incremental Model is an evolution of the waterfall model, where the waterfall model is incrementally applied. The series of releases is referred to as increments, with each increment providing more functionality to the client. After the first increment, a core product is delivered, which can already be used by the client. Based on client feedback, a plan is developed for the next increments, and modifications are made accordingly.

This process continues, with increments being delivered until the complete product is delivered [4], [10],[11].

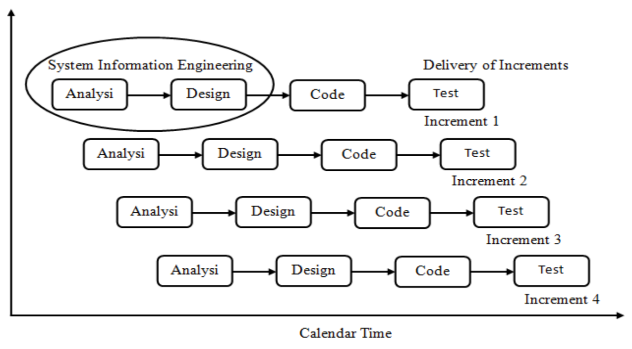

Figure3. Incremental Model

Advantages

1) After using first iteration model, user can give their suggestion and demand for change.
2) It is flexible to the customer's requirements and easy to manage model.
3) This model is used when requirements are clear to some extend but project scope requires pure linear approach.
4) Testing and debugging during smaller iteration is easy.

Disadvantages

1) Each phase of an iteration is very rigid and do not overlap each other.
2) Mapping requirements to increments may not be easy so managing documents are very difficult.
3) During development process changes are being done at first iteration. As if continuous to change and it never finished.
4) More management attention is required due to frequently changes in requirements.

IV. NEW PROPOSED SDLC-2013 MODEL

The New SDLC-2013 model is designed in such a way that it allows client and developer to interact freely with each other in order to understand and implement requirements in a better way to produce a high quality software within budget and schedule. As the Software Development process began with the client's need, so the proposed model tries to discover most of the requirements of the client. It helps in developing an efficient software product that satisfies client. In the sphere of computer based system products, client satisfaction is dependent on how system development process evolves to build operational product systems that satisfy the perceived and actual client's need and associated system requirements. Ultimately, client satisfaction depends upon the depth of 'through-life' understanding about the client needs and associated user requirements for a future system, and the ability to communicate those requirements to the system developer. In addition, client satisfaction and confidence depends upon the level of system assurance offered throughout the system development lifecycle. Requirements understanding problems inevitably lead to poor client-developer relationship, unnecessary re-work, and overrun cost and time. The client satisfaction is totally depended on client needs for this reason SDLC-2013 focus on the initial phases.

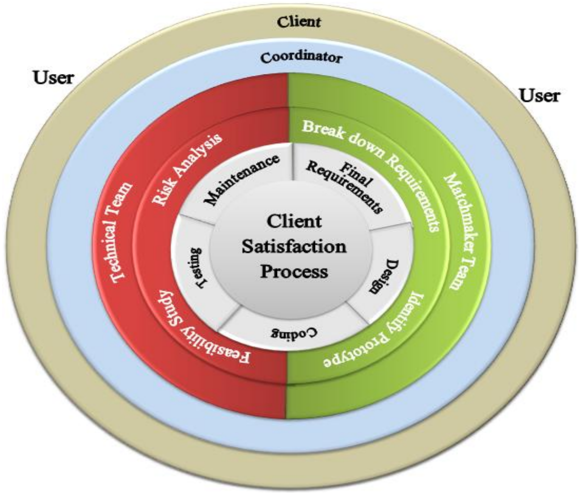

Figure4. New Proposed SDLC-2013 Model

A. *Coordinator*

Coordinator have a general knowledge of every aspect of software development process, software applications, various applicable operating systems or platforms as well as various business functions to be performed. He coordinates with all the phases of the software development process. Coordinator deals with the client for gathering the requirements and passes these requirements to the matchmaker team and any query of client is also solved by the coordinator. After finalizing requirements coordinator estimates the cost, time and effort required to develop the software product. Then he passes the final requirements to the technical team. If client wants any change in the final requirements during the process, then coordinator firstly checks whether it can be implemented or not and what are impacts of change on the whole process in terms of cost, schedule and effort. If change is possible and its impact is little or very less then change will be accommodated.

B. *Matchmaker Team*

Matchmaker team is an expert team and its team members are updated with new technologies and new software products. This team interacts with coordinator and technical team during its working. Matchmaker team studies the requirements received from the coordinator which in turn get these requirements from the client. This team identifies and gets the existing software whose requirements match with the current proposed software's requirements. And accordingly breakdown the requirements into two parts implemented requirements and non-implemented requirements. Implemented requirements are those requirements which are already implemented in some existing software.
Non-implemented are those requirements which are not implemented by any of the existing software, mean new fresh features, and are passed to the technical team. The Matchmaker team then passes the matching software to the coordinator, so that coordinator can show the software as a prototype to the client. Client also gave his suggestion and feedback to the coordinator in order to change of the requirement. The result of breaking requirements and showing existing software as dummy to the client is that the client gets the feel of graphics, functionality and features of product. It helps both client and developer to identify,

Published By:
Blue Eyes Intelligence Engineering
& Sciences Publication

discovers and implements the requirements efficiently.

C. Technical Team

It is a technically expert team. The member of this team is full of skills and interacts with coordinator and matchmaker team. Technical team works on non-implemented 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-implemented requirements. After feasibility study and risk analysis the technical team verify the final requirements and pass these to the next phases, i.e. designing, coding, testing, each of these phase also followed by validation process.

D. Client Satisfaction Process

The software development life cycle is initiated by the client's needs. In the beginning, these needs are in the mind of the client. The software developer by using a software development model has to identify, discover, understand and fulfill the requirement of the client in order to satisfy the client. The requirement phase of the Software Development Life Cycle translates 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 developer fails to satisfy the client because of the three reasons:

A. If fail to discover requirement:

The client usually does not understand Software or the Software Development Life Cycle, and the developer often does not understand the clients problem and application area. But the SDLC-2013 allows the client and developer to interact freely with each other for the better understating of the problem. Moreover, In SDLC-2013 the requirements are break downs into two parts i.e. implemented requirements and non-implemented requirement. For implemented requirements the existing most matching software with the client's requirements are shown to the client so that the client can easily identify and express the requirement to the developer.

B. If fail to implement the requirement:

If the discovered requirement are not fulfilled or implemented properly then it leads to dissatisfaction of client. As, In SDLC-2013 requirements are break down into implemented and non-implemented requirements. The non-implemented requirements are passed to the technical team for the feasibility study of the requirements. This team also identifies the various risk associated with the requirements so, that the conclusion can be drawn about the implementation of requirements. If some Non-implemented requirements are not technical possible or feasible then the client is informed about it during the initialed stages (first phase) so that the client does expect the system with fulfilled no feasible requirements.

C. If requirement Change:

We know that the requirements frequently changed. Some of the changes are inevitable due to changing needs and perceptions. But many changes come because the requirements are not properly analyzed and not enough effort was expended to validate the requirements. But SDLC-2013 is designed in such a way that the developer can focus on proper analyzation of requirements. It is estimated 20% to 40% of total development effort in a software project is due to rework much of which occurs due to change in requirements. According to COCOMO model the cost of the requirement phase is typically about 6% of the total project cost. Consider a project whose total effort requirement is estimated to be 50 person-months. For this project, the requirement phase consumed 3 person months. If by spending an edition 50% effort in the requirement phase. We reduce the total requirement change request 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 [1].

The following details explain the applicability of the new proposed SDLC-2013 Model. The details given below explain how the new Proposed SDLC-2013 Model has the strength of satisfying the client.

V. DEPLOYING SOFTWARE

Software is developed for automating the work of a doctor's clinic. There are various Software Development Models for developing software but we choose Waterfall model, Prototype Model, Incremental Model and New Proposed SDLC-2013 Model for developing software (DCA, stand for doctor's clinic automation) for comparing the working of existing Models with the SDLC-2013. Software developed by traditional SDLC Models.

A. Development of software by Waterfall model

As we know waterfall is a linear sequential flow model. We analyzed the requirements and freeze them and moved toward the designing phase followed by the Coding and testing phases for developing software named as DCA-I. But the DCA-1 was not accepted by the client (doctor) because client was not satisfied, as the client want to change it in terms of graphics, functionality and features. As, waterfall model does not allow changes after freezing the requirements so, it fails to deliver the software product.

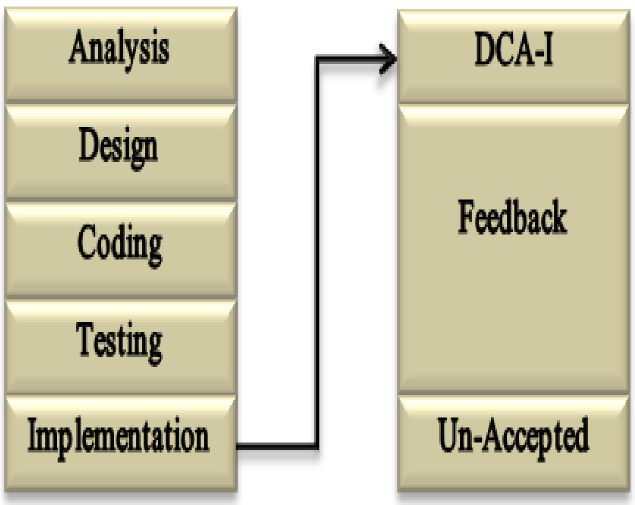

Figure5. Employing waterfall model for software development

B. Development of software by Prototype model

We know that prototype model build prototype to give feel of the proposed software to the client. As we already have doctor's 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, doctor finalized the requirements and we passed these final requirements to next phases to develop the software and named it as DCA-II. Finally DCA-II was delivered to the client. But building prototype affects cost, schedule and effort which get exceeded.

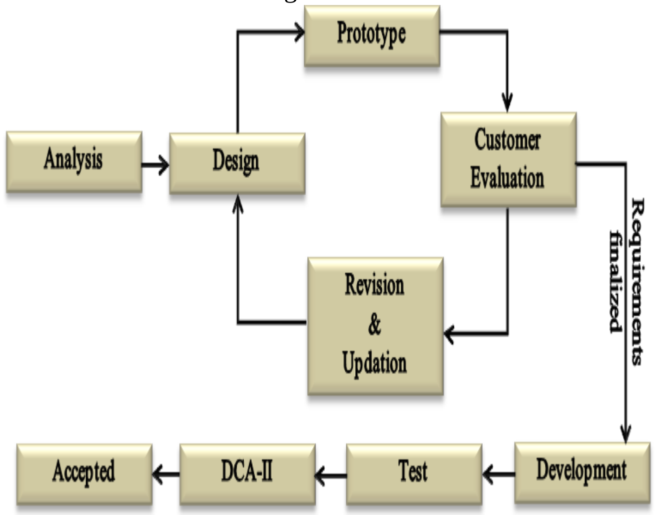

Figure6. Employing prototype model for software development

C. Development of software by Incremental model

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's 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 DCA-III 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.

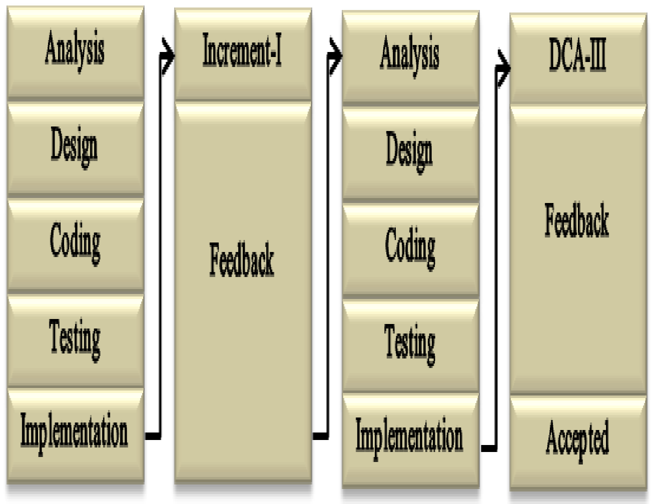

Figure7. Employing incremental model for software development

D. Development of software by SDLC-2013 model

SDLC-2013 is a new Advance Model for the software development. The striking feature of this model is the client satisfaction. Firstly, Coordinator deal with the client (doctor) to discover the requirements and then he passed these requirements to the matchmaker team. Matchmaker team analyzed the available requirements for the proposed system and searched the most matching software for them. He found two such software whose requirements matched with the proposed software's requirements. Accordingly, he has to breakdown the available requirements into implemented and non-implemented requirements but in this case there was no non implemented requirement. Implemented requirements along with their matching software were given back to the coordinator. Coordinator 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 coordinator. Coordinator again passed these suggestions to the matchmaker team and the process goes on until the client finalized the requirements. Coordinator passed final requirements to the technical team for the risk analysis and the requirement validation. After validation and resolving various risk associated with the final requirements, these requirements were passed to designing, coding and testing phases followed by the validation process to develop the final product named as DCA. DCA was accepted by the client because it satisfied the client's requirements within budget and schedule because budget and schedule were not disturbed or affected due to various increments or by building prototype.

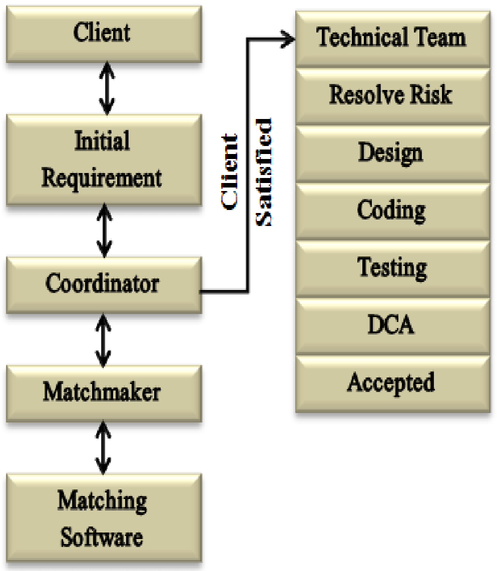

Figure8. Employing SDLC-2013 model for software development

TABLE I **Tabular comparison of SDLC- 2013 with other models**

FEATURES	WATERFALL	PROTOTYPE	INCREMENTAL	SDLC-2013
UNDERSTANDING REQUIREMENTS	WELL UNDERSTOOD AT BEGINNING	NOT WELL UNDERSTOOD AT BEGINNING	WELL UNDERSTOOD AT BEGINNING	WELL UNDERSTOOD AT BEGINNING
COST	LOW	HIGH	MEDIUM	LOW
SCHEDULE	WITHIN SCHEDULE	SCHEDULE MAY EXCEED	SCHEDULE MAY EXCEED	WITHIN SCHEDULE
RISK INVOLVEMENT	HIGH	LOW	MEDIUM	LOW
USER INVOLVEMENT	LOW	HIGH	HIGH	HIGH
GUARANTY OF SUCCESS	LOW	GOOD	HIGH	HIGH
CLIENT SATISFACTION	LOW	HIGH	HIGH	HIGH
FLEXIBILITY	RIGID	FLEXIBLE	FLEXIBLE	FLEXIBLE
TIME FRAME	MEDIUM	SHORT	VERY LONG	SHORT
INITIAL PRODUCT FEEL	NO	YES	NO	YES

VI. CONCLUSIONS

The proposed work can be summarized as the creation of the approach SDLC-2013 to develop software more efficiently. The aim of Software Engineering is to develop software of high quality within budget and schedule. The proposed plan tries to fulfill the objective of Software Engineering by showing existing matching software as prototype to the client for discovering the requirements efficiently from the client in order to estimate cost, schedule and effort more accurately.

REFERENCES

1. K. K. Aggarwal, Yogesh Singh Software Engineering 3rd Edition.
2. Software Development Life Cycle (SDLC) – the five common principles.htm
3. Software Methodologies Advantages & disadvantages of various SDLC models.mht
4. www.shazsoftware.com/software-development-life-cycle.html
5. www.waterfall-model.com/sdlc/
6. Roger Pressman titled Software Engineering - a practitioner's approach.
7. www.en.wikipedia.org/wiki/Systems_development_life-cycle
8. Analysis and tabular comparison of popular SDLC models, International Journal of Advance in Computer and Information Technology (IJACIT), July 2012, Sema, SonaMalhotra.
9. 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,Vishwas Massey, Prof. K. J Satao.
10. 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, Vishwas Massey, Prof. K. J Satao.
11. Comparative analysis of different types of models in Software Development Life Cycle, International Journal of Advanced Research in Computer Science and Software Engineering (IJARCSSE), Volume 2, May 2012, Ms. Shikhamaheshwari, Prof. Dinesh Ch. Jain.

AUTHORS PROFILE

Naresh Kumar dis his Msc-IT from Baba Ghulam Shah Badshah University, Rajouri, J&K., India and Currently pursing M.Phil-IT From Dr. C. V. Raman University, Bilaspur, Chhattisgarh, India.

Dr. A. S. Zadgaonkar, Ph.d (Istru.), Ph.d (Materials), D. Lit (Speech Recog.) is vice chancellor of Dr. C. V. Raman University, Bilaspur Chhattisgarh. He has fourty years of teaching and Adminstrative Experience. He has published more than 470 papers in International, National Journals/ Conferences. He has guided more than 10 Ph.d Candidates. He is author of 3 books. He has received more than 13 Award and 10 Research Grants. He is member of more than 15 socities.

Abhinav Shukla Assistant Professor and HOD (IT) in Dr. C. V. Raman Univrsity, Bilaspur, Chhattisgarh. He did his MSc-IT from Guru Gasidass Ceneteral University Bilaspur, Chhattisgarh. M. Tech (IT) from KSOU. and M.Phil from Dr. C. V. Raman University, Bilaspur, Chhattisgarh. He has more than 10 years of teaching experience. He has published more than 4 papers in National Journals.