A Comparative Analysis of Software Process Models

Ashish K. Sharma
Asstt. Prof.
Computer Technology
M.I.E.T, Gondia.

Sangita A. Sharma
Asstt. Prof. and Head
Computer Technology
M.I.E.T, Gondia.

I. C. Mehta
Asstt. Prof. and Head
Computer Technology
M.I.E.T, Gondia.

ABSTRACT
In today's fast-paced business environment and with floods of data involved in the business, the proper management of data is highly asked for successful running of business. Thus, the companies involved in the business have to use software for the proper management of data. Using software not only removes unnecessary onus but also helps in proper management of data besides saving time and effort. However, the development of software is a complex task. The successful development of software requires a systematic and disciplined approach. Software Engineering (SE) is a systematic and disciplined process to produce software. Software Engineering offers a pool of Software Development Life Cycle (SDLC) models to develop software products. A process model is selected keeping in view of business application to be software automated. Selection of a suitable process model is of utmost importance as each of these models has its own style, approach and applicability. This requires proper understanding of the models. To this end, this paper discuses the various SDLC models used for software development and presents a comparative analysis of the same to show the merits and demerits of each model.

Keywords
Software, Data, Software Engineering (SE); Software Development Life Cycle (SDLC); Process Model;

1. INTRODUCTION
In this competitive environment and with floods of data involved in the business, the proper management of data is highly asked for successful running of business. Thus, the companies involved in the business have to use software for the proper management of data. Using software not only removes unnecessary onus but also helps in proper management of data besides saving time and effort. However, development of software is a complex task. Moreover, the task gets more strenuous if the business to automate happens to be too lengthy as it may involves enormous data, numerous calculations and more time-consuming processes. Automation of such task enforces meticulous examination of the processes involved, gathering of data and then transcripting these data into the technical slabs for effective utilization. Besides, it is also concerned with the tools selected for the automation purpose [1] Also, developing a software requires to undergo a series of tests and revisions before it would become functional. The successful development of a software product requires a systematic and disciplined approach.

SE is a disciplined approach for developing and maintaining software systems that behave reliably and efficiently, are affordable to develop and maintain, and satisfy all the requirements that customers have defined for them [2]. SE is a process to produce software product. SE encompasses software engineering process models, project planning, management and SDLC models which enable the projects to deliver products with expected quality and efficiency, on time and on budget [3]. To develop a software product and to satisfy the customer a set of process activities must be defined and follow in due course, these set of activities are called as software process activities [4]. From the beginning of the idea of software system development, to the completion of process, even after it's delivery to the customer, the system undergoes gradual development changes and evolution. This system building, delivery and changes can be referred to as the software lifecycle, which is composed of several phases [5]. The SDLC models include Waterfall, Increment, Rapid Application Development **(**RAD), Prototyping, Spiral, etc. A process model has to be selected keeping in view of business application to be software automated. Selection of a suitable process model is of utmost importance as each of these models has its own style, approach and applicability. This requires proper understanding of the models. A proper selection of process model not only saves the developer from unnecessary onus but also assures the satisfactory software development in time and budget leading to higher customer satisfaction. To this end, this paper discuses the various SDLC models used for software development and presents a comparative analysis of the same to show the merits and demerits of each model.

2. SOFTWARE ENGINEERING (SE)
Development of software system is a complex task. Also, developing a software system requires to undergo a series of tests and revisions before it would become functional. SE is the application of a systematic, disciplined, quantifiable approach to the development, operation, and maintenance of software, and the study of these approaches; that is, the application of engineering to software [6]. The term software engineering first appeared in the 1968 NATO Software Engineering Conference, and was meant to provoke thought regarding the perceived "software crisis" at that time [7]. SE incorporates a development strategy that encompasses the process, methods, and tools. This strategy is often referred to as a software process model or a software engineering paradigm. The current definition of SE is still being debated by practitioners today as they struggle to come up with ways to produce software that is "cheaper, better, faster" [8].

SE can be divided into ten sub-disciplines. They are: [6]

Software requirements: The elicitation, analysis, specification, and validation of requirements for software.

Software design: The process of defining the architecture, components, interfaces, and other characteristics of a system or component. It is also defined as the result of that process.

Software construction: The detailed creation of working, meaningful software through a combination of coding, verification, unit testing, integration testing, and debugging.

Software testing: The dynamic verification of the behavior of a program on a finite set of test cases, suitably selected from the usually infinite executions domain, against the expected behavior.

Software maintenance: The totality of activities required to provide cost-effective support to software.

Software configuration management: The identification of the configuration of a system at distinct points in time for the purpose of systematically controlling changes to the configuration and maintaining the integrity and traceability of the configuration throughout the system life cycle.

Software engineering management: The application of management activities planning, coordinating, measuring, monitoring, controlling, and reporting to ensure that the development and maintenance of software is systematic, disciplined, and quantified.

Software engineering process: The definition, implementation, assessment, measurement, management, change, and improvement of the software life cycle process itself.

Software engineering tools and methods: The computer-based tools that are intended to assist the software life cycle processes and the methods which impose structure on the software engineering activity with the goal of making theactivity systematic and ultimately more likely to be successful.

Software quality: The degree to which a set of characteristics fulfils the requirements.

A software development process or SDLC, is a structure imposed on the development of a software product. Similar terms include software life cycle and software process. It is often considered a subset of systems development life cycle. There are several models for such processes, each describing approaches to a variety of tasks or activities that take place during the process. [9]. A process framework establishes the foundation for a complete software process by identifying a small number of framework activities that are applicable to all software projects, regardless of their size or complexity [10]. The main activities are applicable to the vast majority of software projects and they are:

- Requirements acquisition
- Planning
- Modeling
- Construction
- Deployment

[10] divide the process models into two types: prescriptive and agile software development. Prescriptive software process models prescribe a set of process elements: framework activities, software engineering actions, tasks, work products and quality assurance and change control mechanisms for each project. Each process model also prescribes a workflow, that is, the manner in which the process elements are interrelated to one another. On the other hand, agile software development aims to flexibility and adjustability [11]. Commonly used agile process model types are Extreme Programming (XP), Scrum, Adaptive Software Development, Feature Driven Development etc.

3. SOFTWARE PROCESS MODELS

3.1 The Waterfall Model

The waterfall model is proposed by Winston Royce and is called the Classic life or linear sequential model. It is a prescriptive model means it prescribes how a new software system should be developed. It consists of phases – Requirement Analysis, Design, Implementation, Testing and Deployment that are completed sequentially before proceeding to the next phase as shown in Figure 1. It suggests a systematic, sequential approach to software development that begins with customer specification of requirements and progresses through planning, modelling, construction and deployment, culminating with on-going support of the completed software [12].

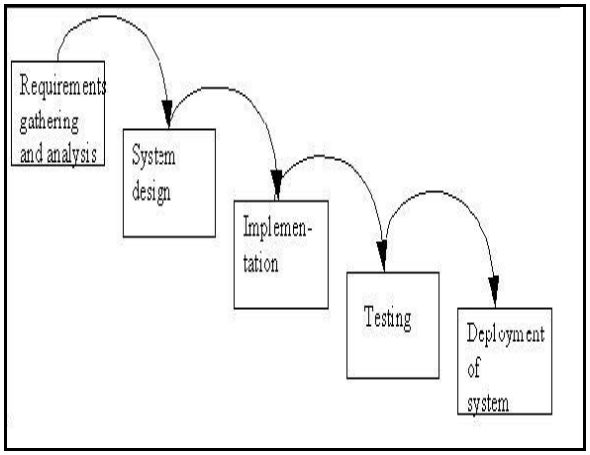

Fig. 1 The Waterfall model (Source Kivinen, 2008)

It is a formal method that follows a top-down approach. The advantages and disadvantages of this model as depicted by [13] are shown below.

Advantages

- Easy to understand and implement
- Widely used and known
- Reinforces good habits: define-before- design, design-before-code
- Identifies deliverables and milestones
- Works well on mature products and weak teams

Disadvantages

- Doesn't reflect iterative nature
- Unrealistic to expect accurate requirements so early in project
- Software is delivered late in project, delays discovery of serious errors
- Difficult to integrate risk management

3.2 Prototyping Model

The need of such a model originates from the fact that the customer is non-technical person and they are not able to specify detailed input, processing, or output requirements. They just define a set of general objectives for software. In other cases, the developer may be unsure of the efficiency of

an algorithm, the adaptability of an operating system, or the form that human/machine interaction should take. In these, and many other situations, a prototyping paradigm may offer the best approach. The prototyping model is shown below in Figure 2.

This model begins with the requirement gathering, then production of a quick design according to this requirement, construction of a prototype, customer evaluation of prototype and then production of final product. Prototyping model is iteration-based model as iterations are made on prototype to satisfy the needs of customer and development of better one [14].

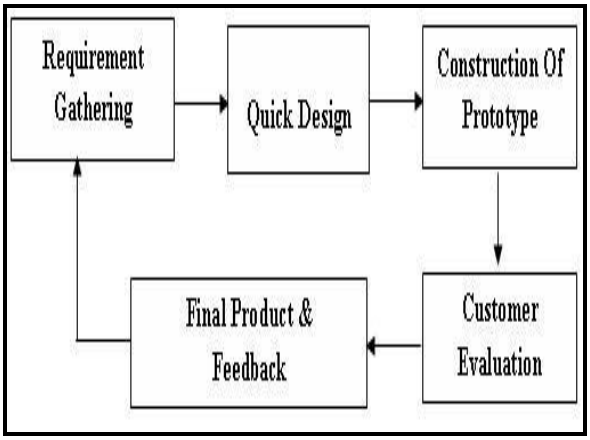

Fig. 2: Prototyping Model

The advantages and disadvantages of this model as depicted by [15] are as under.

Advantages

- Reduced time and costs
- Improved and increased user involvement

Disadvantages

- Insufficient analysis
- User confusion of prototype and finished system
- Developer misunderstanding of user objectives
- Developer attachment to prototype
- Excessive development time of the prototype
- Expense of implementing prototyping

3.3 The Spiral Model

The spiral model is an iterative model which includes risk analysis and risk management. The spiral model is proposed by Boehm and is shown in Figure 3. The basic idea behind spiral model is that it is hard to understand the end-user requirements. So software is developed on an experimental basis and check whether it fulfils the customer needs, if not rebuild it. This shows the iterative nature. The main characteristic of the spiral is that it is cyclic and not linear like waterfall model [5]. This model clearly shows the evolutionary nature of software development on task with not specified requirements. It combines the merits of linear sequential model and prototyping methods. In the spiral model, the angular component represents progress, and the radius of the spiral represents cost.

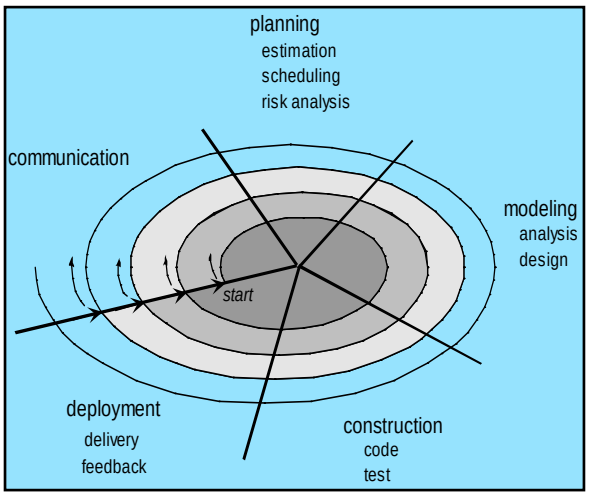

Fig. 3: Spiral model

The merits and demerits of this model as mentioned by [13] are as under.

Advantages

- High amount of risk analysis
- Good for large and mission-critical projects
- Software is produced early in the software life cycle

Disadvantages

- Can be a costly model to use
- Risk analysis requires highly specific expertise
- Project's success is highly dependent on the risk analysis phase
- Doesn't work well for smaller projects

3.4 Rapid Application Development Model

Rapid application development (RAD) is an adaptation of the linear sequential model. It processes in such a manner that if the requirements are collected and understood, it enables a development team to create a "fully functional system" within a very short time period (e.g. 60 to 90 days) [14]. The RAD model is shown in Figure 4. It uses component-based construction, but is high speed linear sequential. This approach consumes less time for software formation and also reduces the effort and money because of reuse of components. If a system cannot be properly modularized, building the components necessary for RAD will be problematic [16].

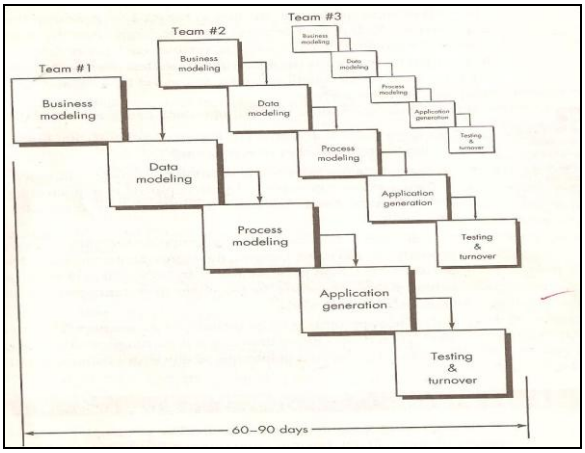

Fig. 4: RAD Model [Source: Pressman, 2000]

RAD is not appropriate when technical team is not so able to manage risk analysis. The merits and demerits of RAD model as depicted by [17] are as under:

Advantages

- Software development time is highly reduced due to a reduced requirement analysis
- Reusability of components is possible which results in speedy software development
- If a component is being picked for the repository, it is already tested and hence need not be tested again. This helps in saving time required for testing
- It promotes better documentation through written test cases
- Due to strong participation of project sponsor, the end user satisfaction level is higher

Disadvantages

- RAD may not be appropriate when technical risks are high
- This method may not be useful for large, unique or highly complex projects
- This method cannot be a success if the team is not sufficiently motivated and nor is unable to work cohesively together
- Success depends on the extremely high technical skills of the developers

3.5 Incremental Model

This model is the minimum departure from the linear sequential model in the direction of an evolutionary process [5]. The basic idea behind this model is to create an increment by the same software production processes namely analysis, design, coding and testing according to customer requirement and then increment of this is delivered to the customer. The first increment is called a core product. As the customer uses the core product a plan is developed for next increment. These plans suggest the modification of core product to meet the needs of the customer and thus delivery of additional features and functionality. This process is repeated until the complete product is produced. Figure 5 shows incremental model.

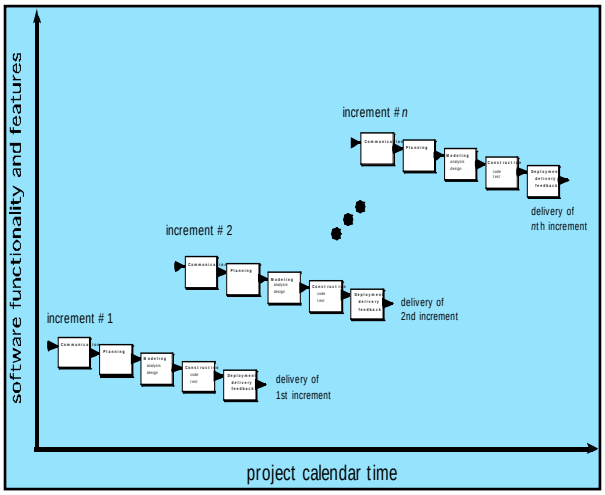

Fig. 5: Incremental Model

The merits and demerits of this model as depicted by [18] are as under.

Advantages

- The versions are provided after each iteration of the incremental model
- Incremental model includes use of the software by user to for changes
- This model dose not effect anyone's business values because they provides core of the software which customer needs, to the customer first, which will help him/her to keep running his/her business.
- It is flexible and easy to manage more manageable process and better software making and better software structure
- Better risk management because you can confirm the outcome by the customer after every version that whatever u are making is according to plan or not and fix it in next version etc.
- Testing is easy

Disadvantages

- Each phase of an iteration is rigid and do not overlap each other
- Problems may arise pertaining to system architecture because not all requirements are gathered up front for the entire software life cycle

4. SOFTWARE PROCESS MODEL FOR eStore WEBSITE DEVELOPMENT

The eStore is basically an online shopping site which caters to the wants and needs of customers by providing products online for shopping. A website eStore, in a way, is a software, which is developed for users on the World Wide Web. Therefore, for the website to come to its final stage, there are various stages through which the website has to go through. The website development process can be compared to software development [19]. Thus, like any other software application, a process model has to be selected for a website development. A suitable process model is very important as it plays a significant role in the design and development of website. In order to select a particular process model for

eStore it is needed to analyze the development plan used in website development. Also the website development plan for an eStore comprises of following steps:

- Information Gathering
- Requirement Analysis
- Planning
- Design
- Development
- Testing
- Delivery
- Maintenance and Updating

The steps above strongly resemble the linear or sequential process where each step is to be completed first before proceeding to next step. Thus, amongst the various models available, the waterfall model fits well for the website development of eStore.

5. CONCLUSIONS

Through this paper an attempt has been made to deal with a vital and important issue of software development, the software process model. The software process models act as an implementation vehicle in the software development process and thus the proper selection of a software process model is highly recognized in software development. With the pool of models available, selection of a suitable process model from the available ones is of utmost importance as each of these models has its own style, approach and applicability. This requires proper understanding of the models.

To this end, this paper has discussed the various SDLC models. It also presents the merits and demerits of these models which highlights their scope and applicability. The study carried out in this paper suggest that for applications where sequential flow is required, waterfall model seem to be the right choice while for some applications where implementation is carried out in iteration, the prototyping model is good. The study also reveals that if risk analysis and reusability are the issues then RAD is the best choice. It is also realized that if the software has to be developed in parts, incremental model is most suitable. The study presented here will strongly assist the software developers in the proper selection of process model for the development of particular software which in turn will lead to the development of successful and efficient software. This study concludes with the evaluation of process model for an eStore website development and it is realized that waterfall is the suitable choice.

6. REFERENCES

[1] A. K. Sharma, I. C. Mehta, J. R. Sharma, "Analyzing Programming Tools For The Development Of Quality Function Deployment Software", International Journal of Information and Decision Sciences, Inderscience Publishers, 2010, Vol. 2, No. 2, pp. 132-146.

[2] http://computingcareers.acm.org/?page_id=12

[3] A.R.M . Reddy, P. Govindarajulu, M. M. Naidu, "A Process Model for Software Architecture", International Journal of Computer Science and Network Security, 2007, VOL.7, No. 4.

[4] Sommerville, I., Sawyer, P. 1997 Requirements Engineering: A Good Practice Guide. John Wiley and Sons.

[5] Ghezzi, C., Jazayeri, M., Mandrioli, D. 1996 Fundamental of Software Engineering. Prentice Hall of India Pvt. Ltd.

[6] Bourque, P. and Dupuis, R. Guide to the Software Engineering Body of Knowledge. (2004), 1-1.

[7] http://en.wikipedia.org/wiki/Software_engineering

[8] http://en.wikipedia.org/wiki/Software_engineering# History

[9] http://en.wikipedia.org/wiki/Software_engineering_ process

[10] Thayer, R. H. and Christensen, M. J. 2005 Software Engineering, Volume 1: The development Process, Third Edition. John Wiley and Sons.

[11] Koch, A. S. 2005 Agile Software Development: Evaluating the Methods for your Organization. Artech House.

[12] Kivinen, T. 2008 Applying QFD to Improve The Requirements and Project Management in Small-Scale Project. Masters Thesis. University of Tampere.

[13] N. M. A. Munassar, A. Govardhan, "Comparison Between Five Models Of Software Engineering", IJCSI International Journal of Computer Science Issues, 2010, Vol. 7, No. 5.

[14] Pressman, R. 2000 Software Engineering: A Practitioner Approach, 5th Ed., Mcgraw-Hill.

[15] http://www.iotap.com/Blogs/tabid/277/EntryId/152/Advantages-Disadvantage-of-Prototyping-process-model.aspx

[16] J. Butler, "Rapid Application Development in Action", Managing System Development, Applied Computer Research, 1994, Vol. 14, No. 5.

[17] http://www.my-project-management-expert.com/the-advantages-and-disadvantages-of-rad-software-development.html

[18] http://www.blurtit.com/q9036463.html

[19] http://www.buzzle.com/articles/website-development-process.html