Volume 6, Issue 9, September 2016 ISSN: 2277 128X
International Journal of Advanced Research in Computer Science and Software Engineering
Research Paper
Available online at: www.ijarcsse.com

Comparative Study and Analysis of Software Process Models on Various Merits

Harminder Pal Singh Dhami
Department Computer Science, OPJS University, Churu,
Rajasthan, India

Abstract: ***Day by day I.T industry is increasingly relying on a growing quantity of ever-larger software, and software development one of the most powerful, vital and important issue. The success of a software project greatly depends upon which process model is used for development, as IT solutions & projects have become essential in organizations of all sizes. At the time an organization builds / implements an IT solution, it involves considerable time, money & resources to the project with an expectation of valuable outcome. System development life cycle (SDLC) is all about the minimization of risk and failure and maximization of quality software product. It is the systematic and structural method of software developing process. The SDLC defines the software development structure that comprises of various activities and tasks to be carried out during the software development process. There are various software development life cycle models that are used in the software development process having their own advantages and disadvantages. The objective of this research paper is to represent various SDLC models and make a comparative analysis on various parameters.***

Keywords: Software Development Life Cycle (SDLC), Models and Comparative Analysis

I. INTRODUCTION

The need and importance of computers are growing exponentially day by day. There are hardly any areas where computers are not being used. Specialized software's are required according to the application areas. IT product development has its own challenges – hurdles because it required an entirely different approach. Different approach is required because of its inherent quality of being intangible in nature [10]. Hardware alone is not adequate to do some useful work. Hardware and Software complements each other. Software engineering [6] is an engineering discipline whose aim is development of quality product, a product which is reliable, within estimated budget and within a given time framework. We can say the process defines what different activities and tasks are to be carried out during software development.

As we know that most of the time software problems are the main causes of system failures. There are many well-known cases of the tragic consequences of software failures. In systems, high reliability is naturally expected. Software packages need to be highly reliable, because the enormous investment of the software developer is at stake. Studies have shown that reliability is regarded as the most important attribute by potential customers.

Jintao Zeng et al.[12], Investigated that Software reliability models are used for the prediction and estimation of software reliability. In their research paper, "A Prototype System of Software Reliability Prediction and Estimation", IITSI they had proposed an approach for software reliability model selection based on experiences from history software projects. All software developed will have a significant number of defects. All programs must be tested and debugged, until sufficiently high reliability is achieved.

In order to estimate as well as to predict the reliability of software systems, development of good software solution requires a proper process to be followed. This software process which is required to produce software differs in IT industry. As IT Product Development requires a different approach than the conventional projects because of exploiting the Knowledge Management Approach for managing IT Product Development is to decrease the uncertainty and increase the rate of success. Use of managerial knowledge and experience of the existing enterprise will enable organizations to build successful and reliable IT products [10].

Software development life cycle -SDLC is the systematic approach for developing a software product within the time and maintain quality of the software. A proper software life cycle model provides the set of activities to be carried out during the system development. Software development is divided into set of activities that allow any software development company to control the software product easily. The software development life cycle models use the step by step approach to complete the software development process. It can help IT industry not only in building a software product but it also serves as a basis for planning, designing, organizing, staffing, coordinating and directing various other software development activities. If the process is strong, the end product will also be strong and project can be reliable. While developing software product all developers (involved directly or indirectly) should focus on quality, Process, Methods, and Tools.

The software development cycle is all about.

- Problem domain i.e. Understanding the problem.
- Solution domain i.e. Decide a plan for solution.
- Designing and Coding of planned solution.
- Test the actual program.
- Maintain the product.

Software Engineering Terminology in IEEE standard Glossary, the software lifecycle is "The period of time that starts when a software product is conceived and ends when the product is no longer available for use". The software life cycle typically include the following activities which are as follows:

- Planning
- Requirements Analysis
- Design
- Software architecture
- Development or coding
- Testing
- Documentation
- Training and Support
- Maintenance

The above activities form part of SDLC framework activities and are performed in every software development project.

The generic stages that characterize the software development process - plan ning, analysis, designing, development, testing, deployment or implementation, maintenance & support – are applicable to all the models of software product development as illustrated in the figure (Fig.1).

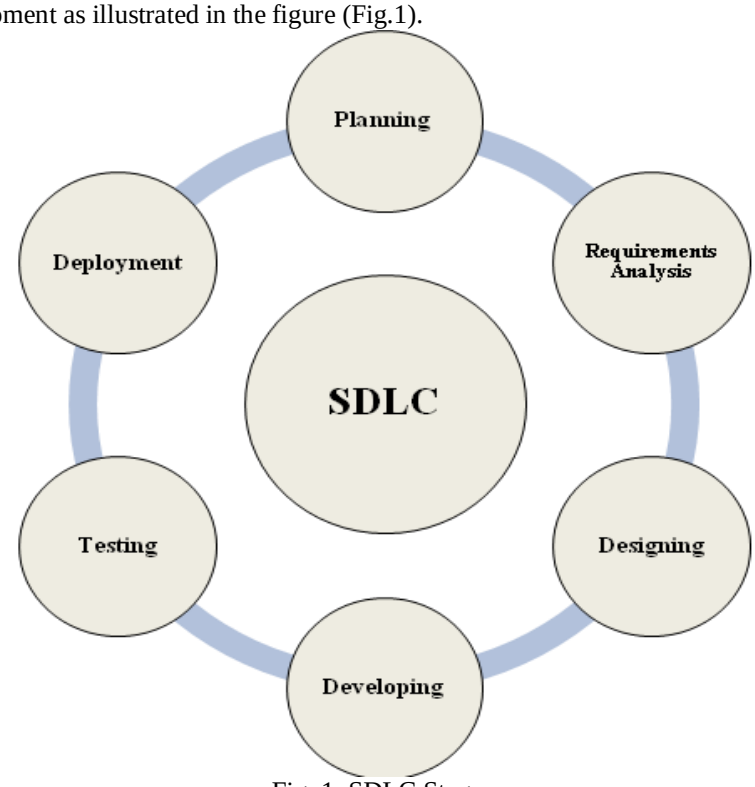

Fig. 1: SDLC Stages

II. SOFTWARE PROCESS MODELS

The software process model is the representation of process which presents the description of a process as – Specification, Design, Validation and Evolution. In this paper we are going to present a comparative study of the following general software process models:

1. Waterfall model
2. Iterative Waterfall model
3. Prototype model
4. Rapid application development model (RAD)
5. Incremental model
6. Spiral model
7. Build and fix model
8. V-shaped model

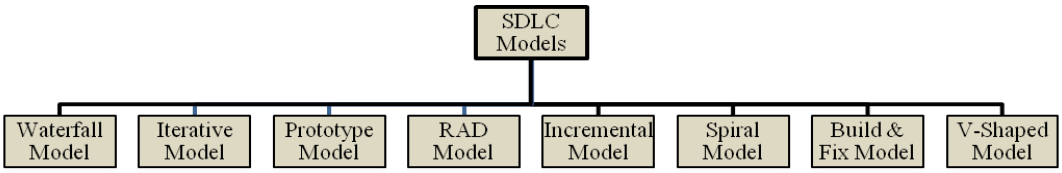

Fig.2: Various Software Process Models

III. SDLC MODELS

3.1 Waterfall Model

In 1970 Rocye[9] proposed this classical model of software engineering. The model is one of the oldest models used in many of the major companies. It is also known as classical lifecycle model or linear sequential model. It is named "Waterfall" because its diagrammatic representation looks like a cascade (flow or drop) of Waterfall. The model begins with requirement analysis and continues with design, coding, testing and maintenance [11]. The phases are placed in such a manner that the phase executed once can't be repeated again. All the phases of waterfall model are independent of each other and developer must complete each phase before the next phase could begin.

This model is suitable for projects in which requirements are specified before the start of the project and they are well defined in earlier stages. This model is simple to understand and use. It follows a sequential approach. The product cannot be delivered to the client until the final stage is over.

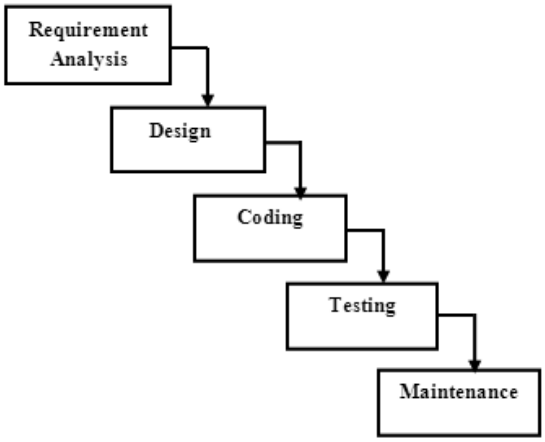

Fig.3: Classical Waterfall Model

As the water fall model is the classical model so it serves as the baseline of all other models. The waterfall model consists of several non overlapping stages as shown in the figure below. It is a one way channel as waterfall i.e. once a phase is executed there is no provision of going back.

Advantages:

1. Easy to understand and use.
2. Provides well defined structure.
3. Objective and requirements are clear.
4. Define before design.
5. It is used for project with simple and strict deadlines.
6. Good for management control (plan, staff, and track) works well when quality is more important than time or cost.

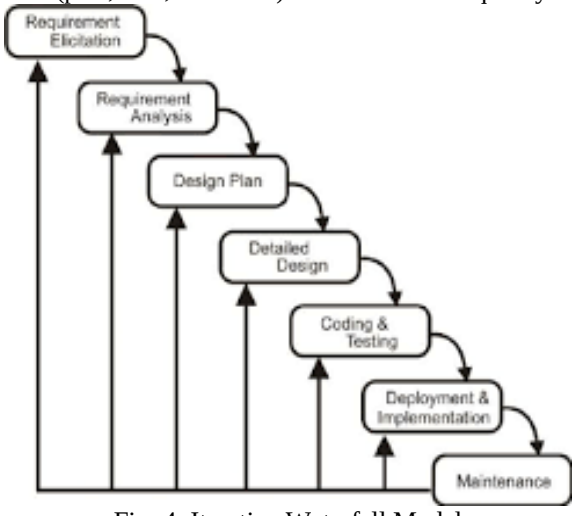

Fig. 4: Iterative Waterfall Model

Disadvantages:

1. Requirements must be clearly defined upfront
2. Outcomes created for each phase are considered to be fixed.
3. Problems remain uncovered until testing.
4. Late delivery.
5. Not easy to integrate risk Management.
6. Difficult to incorporate change during development.
7. Little opportunity for customer to preview the system during development.

3.2 The Iterative Waterfall Model

A need of a new model was felt due to the limitations of the classical waterfall model. The iterative model came into existence to handle the problems of the original waterfall model. The iterative waterfall model is an enhanced version of classical waterfall model which could provide faster results require less time and packed with good flexibility. New model is created by adding an "iterative" loop at the end of the cycle which allows a return to previous stages and the changes can be done whenever required. In this model problem is divided into small parts and this allows developer team to go easily and quickly for their goal and obtain their valuable feedback from users. The project that is divided into small parts and each part is taken as a mini waterfall process.

Advantages:

- Much better model of software process.
- Client can get Feedback.
- Used in that type of projects were requirements are not clear.
- Document driven process.
- Works well on week teams.

Disadvantages:

- Not easy to manage.
- Not clear mile stones.
- No stage is finished really.

3.3 The Prototyping Model

This model requires that before carrying out the development of the actual product, a working model or prototype of the system should be built. Prototype usually turns out to be a very crude version of the actual system, possibly a small working model. A prototype [5] is a toy implementation of a system. Prototyping can be evolutionary or throwaway. In prototyping first requirements of customer are gathered from them and then a working prototype is developed as per their requirements. After the prototype is developed it is provided to the customer and customer review it to see that whether it meets its expected requirements. Prototype is developed to determine the actual need of the customer. Evolutionary Process models are the iterative type models using this model the developer can develop increasingly more complete version of software.

After finalization of software requirement specification developer attempts to use existing program segments from prototype and actual system is then developed using waterfall approach to produce good quality software product. It is generally used when detailed information related to input and output requirements of the system is not available. In prototype model here we have the quick design phase through which the developer has to quickly design the software after requirement gathering as shown in figure 5.

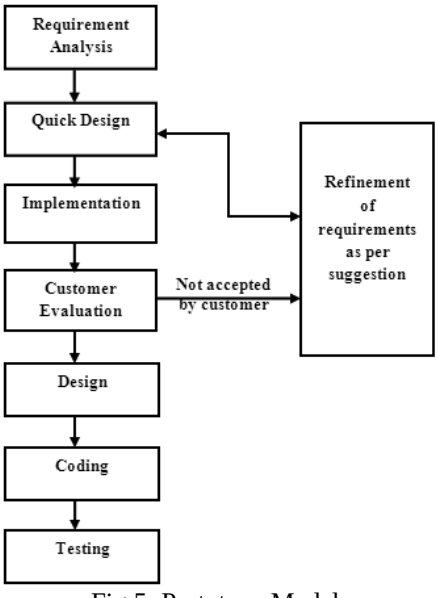

Fig.5: Prototype Model

Advantages:

- Applied when detailed information related to input & output requirements is not available.
- Regular visibility progress aids management
- Reduces risk of incorrect user requirements
- Good where requirements are changing frequently
- Higher outputs.
- Cost reductive.
- Client can actually feel with the system, i.e. feedback.
- Early design supports early product marketing

Disadvantages:

- Possibility of unfinished systems. As an unstable/badly implemented prototype often becomes the final product.
- Possibility of inadequate system so requires extensive customer collaboration.
- Lack of flexibility.
- Not suitable for large systems.
- Management is very complicated and becomes difficult to finish if customer is not committed.

3.4 Rapid Application Development (RAD) Model

The Rapid Application Development generally referred as RAD. RAD model is a high speed adaption of waterfall model. Rapid application development is a model based on the concept that higher-quality products can be developed faster through more expedient processes, such as early prototyping, reusing software components and less formality in team communications. This model can be implemented if a developer knows the requirements of customer in advance and its development cycle is extremely small. Customer involvement is there and in every stage of RAD model to gathering requirements using workshops or focus groups, Prototyping and early, reiterative user testing of designs, the re-use of software components, a rigidly paced schedule that defers design improvements to the next product version and less formality in reviews and other team communication. RAD should be used when there is a need to create a system that can be modularized in 2-3 months of time.

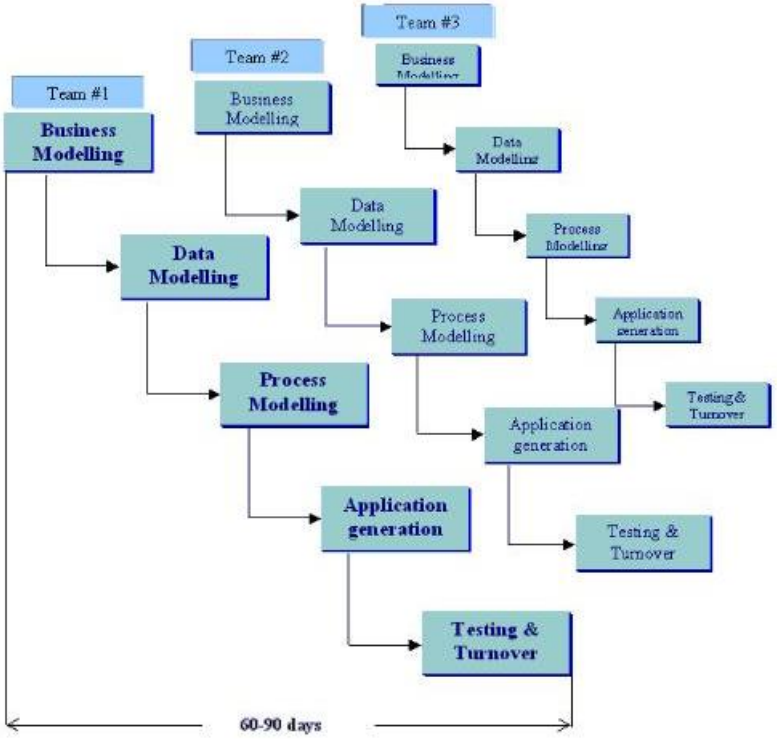

Fig.6: RAD Model

This model has four phase business modeling, Data modeling, Process modeling, Application generation and Testing and turnover. A number of teams work on a single function and then it is integrated to form whole software.

- **Business modeling:** The information flow is identified between various business functions. **Data modeling:** Information gathered from business modeling is used to define data objects that are required for the business.
- **Process modeling:** Data objects defined in data modeling are converted to achieve the business information flow to achieve some specific business objective.
- **Application generation:** Automated case tools are used to convert process models into code and the actual system.
- **Testing and turnover:** Test new components and its all interfaces.

Advantages:

- Reduces development time.
- Increases reusability of components.
- Quick initial reviews occur
- Encourages customer feedback
- Integration from very beginning solves a lot of integration issues.

Disadvantages:

- Depends on strong team and individual performances for identifying business requirements.
- Only system that can be modularized can be built using RAD
- Requires highly skilled developers / designers.
- High dependency on modeling skills
- Inapplicable to cheaper projects as cost of modeling and automated code generation is very high.

3.5 Incremental Model

The incremental model is a method where the product is designed, implemented and tested incrementally (each time a little more is added) until the product is finished. It involves both development and maintenance. If a customer requires changes in its product, then incremental model [8] accommodate changes as required by the customer. The previous models discussed earlier do not take into consideration changes in product. This model is iterative in nature. It construct a partial implementation of a total system. A reusable product is released at the end of each cycle, with each release providing additional functionality. Then slowly add increased functionality. The incremental model prioritizes requirements of the system and then implements them in groups. After each release customer can do some useful work and thereby he can accommodate changes in the product. This model combines the features of the waterfall model with the iterative nature of prototyping. The waterfall and prototype model gives complete operational software while iterative model delivers an operational quality product at each release. In incremental model the product is divided into number of components, each of which are designed and built separately. Each component is delivered to the client when it is complete. This allows partial utilisation of product and avoids a long development time. Each subsequent release of the system adds function to the previous release, until all designed functionality has been implemented. The product is defined as finished when it satisfies all of its requirements.

Such models are used when requirements are clearly defined and can be implemented phase wise. Mostly such model is used in product based companies and web applications.

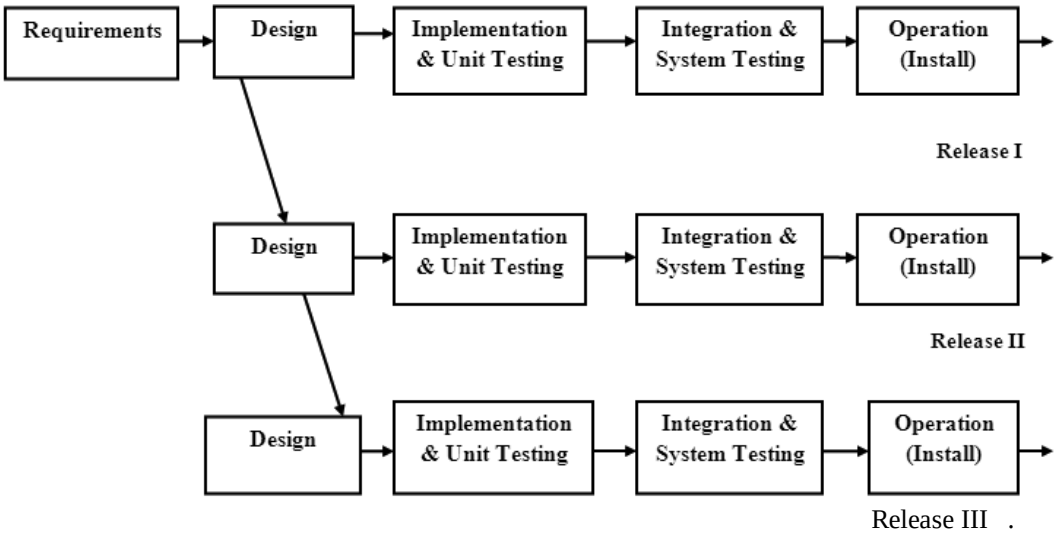

Fig.7: Incremental Model

Advantages:

- Develop high-risk or major functions first
- Each release delivers an operational product
- Customer can respond to each build
- Uses “divide and conquer” breakdown of tasks
- Initial product delivery is faster during the software life cycle.
- Customers get important functionality early
- Risk of changing requirements (at end) is reduced
- More flexible – less costly to change scope and requirements.
- Easier to test and debug during a smaller iteration.
- Easier to manage risk as risky modules are identified and handled during its iteration.
- Each iteration is an easily managed routine.

Disadvantages

- Requires very thoughtful planning and design
- Requires early definition of a complete and fully functional system to allow for the definition of increments
- Well-defined module interfaces are required (some will be developed long before others) – to allow proper increment
- Each new build must be integrated with previous builds and any existing systems.
- Each phase of an iteration do not overlap each other and is fixed.
- Problems may arise pertaining to system architecture as all requirements are not gathered up front for the entire software life cycle

3.6 The Spiral Model

The spiral model was developed in late 1980s, it was proposed by Barry Boehm, and introduces a factor of risk analysis that other models did not take into account [1]. It is cyclic in nature and project undergoes each phase repeatedly called spiral. It consists of four phases and each phase is represented by one quadrant. The four phases are Requirement planning, Risk analysis, Engineering (development & test) and Customer evaluation & Feedback. The planning phase is the base line spiral and during this phase requirements are gathered.

The objective of planning phase is to determine resources for the project as well as what functionality we want to incorporate in the project. Project Cost and scheduling is also considered in it. In the risk analysis phase focus is on identification of risks and at the same time providing alternate solutions for them. Software is developed in the engineering phase and during customer evaluation it undergoes testing by the customer and feedback is collected. The radius of the spiral represents the cost and angular dimension represents the progress in process. In this model aim is to identify high risks related to project and resolve it before it threatens the software operation or cost. Risks associated with over budget, delay, unmanageable and unsatisfactory product quality must be taken into account and resolved before processing to next phase.

Basically, the spiral model attempts to bring together key aspects of some other prominent models (like the waterfall, incremental, and evolutionary prototyping), in an attempt to gather the most appropriate qualities from each one, because specific projects might be more or less adaptable to specific models[3]. This type of model is used in risk analysis project for ex. Space Crafts. It has properties of all above said models and so it is also referred as Meta model.

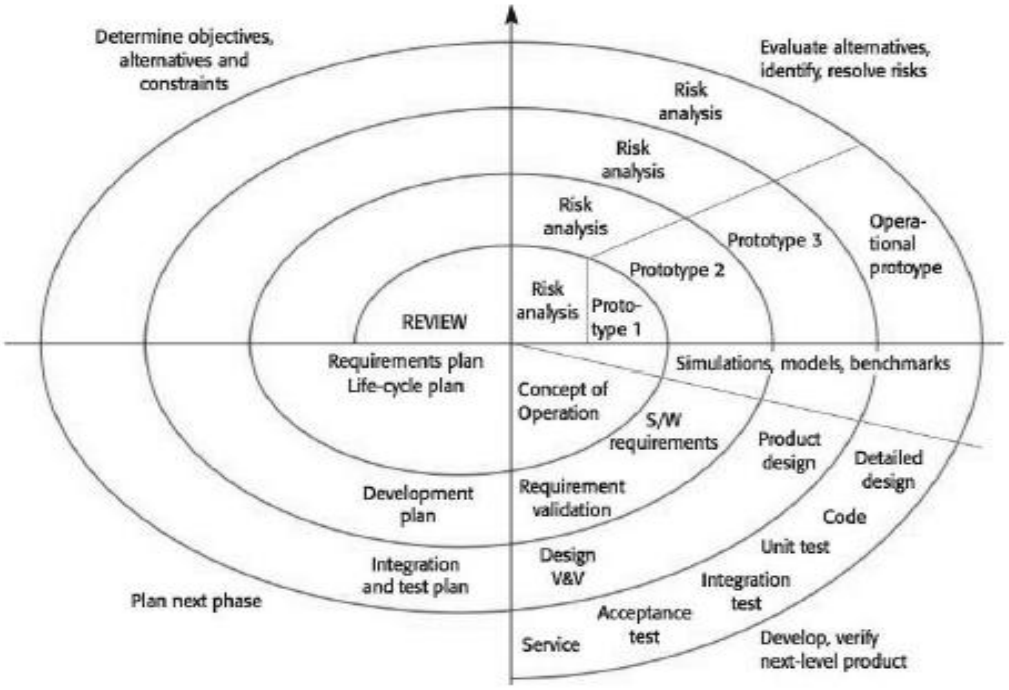

Fig.8: Spiral Model

Advantages:

- New prototype obtained after each cycle/iteration.
- Reuse capabilities.
- Better productivity.
- It has systematic stepwise approach.
- Elimination of errors in early stage.

Disadvantages:

- No proper cost and time estimation.
- Cost of risk analysis is high on large projects.

- Time consuming and Complex model.
- Need expertise for this model.

3.7 Build and Fix Model

In the *build and fix model*, also referred to as an ad hoc model. In this model a software product is built without any specification and without applying any kind of design. Developer in this model adopts an adhoc [4] approach which is not well defined. Using this model the project resulted in number of failures because the product was not constructed using proper specification and design. Developer built the product as many times as possible until it satisfies the customer. Build and Fix are two phases of the model.

Build phase deals with writing the software code and passed on to the next phase. Fix phase deals with correcting the code developed in the build phase in correspondence to user requirements. Build and Fix model requires less experience to execute or manage other than the ability to program. Less project planning is required and is suitable for smaller products.

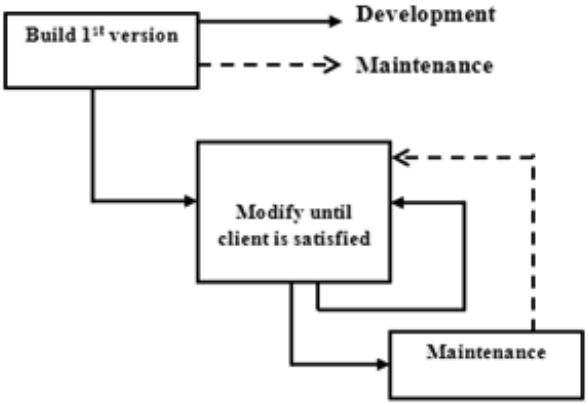

Fig. 9: Build and Fix model

Advantages:

- The model is useful only for small size projects.
- Requires less experience to execute or manage other than the ability to program.
- Requires less project planning.

Disadvantages:

- Informal design of the product as it involves unplanned procedure.
- Non specification of requirements leads to product full of errors
- Reworking results in cost escalation.
- No methodology is available for assessing the progress, quality, and risks.
- Maintenance is quite problematic.
- Not suitable for large and complex projects

3.8 V-Shaped Model or Vee-Model

The V-Model, also known as the Vee-Model, is a product-development process, which was originally developed in Germany for government defense projects. It gets its name from the fact that the process is often mapped out as a flowchart that takes the form of the letter V. V-Model can be considered as an extension of waterfall model as both the models are sequential. In waterfall model we move in a linear way while in V model process steps are bent upwards the coding phases to form typical V-Shape [7].

The relationship between each phase of the development process /verification and its associated phase of testing /validation is shown in V model. As testing is an important part of Software development process so the V shaped model emphasis more on testing.

Requirements have to be very clear before work on the project starts because it is usually expensive to go back and make changes for the existing projects. This model is generally used in the medical development field, as it is strictly disciplined domain. Following are the suitable scenarios where V-Model can be used:

- Requirements are well defined, clearly documented and fixed.
- Product definition is stable.
- Technology is not dynamic and is well understood by the project team.
- There are no ambiguous or undefined requirements.

As shown in figure (Fig.10) the development process proceeds from upper left point of the V toward the right, ending at the upper right point. In downward-sloping branch (left-hand) of the V, development personnel define business requirements, application design parameters and design processes. At the base point of the V, the code is written. In upward-sloping branch (right-hand) of the V, testing and debugging is done. The unit testing is carried out first, followed by bottom-up integration testing. The extreme upper right point of the V represents product release and ongoing support.

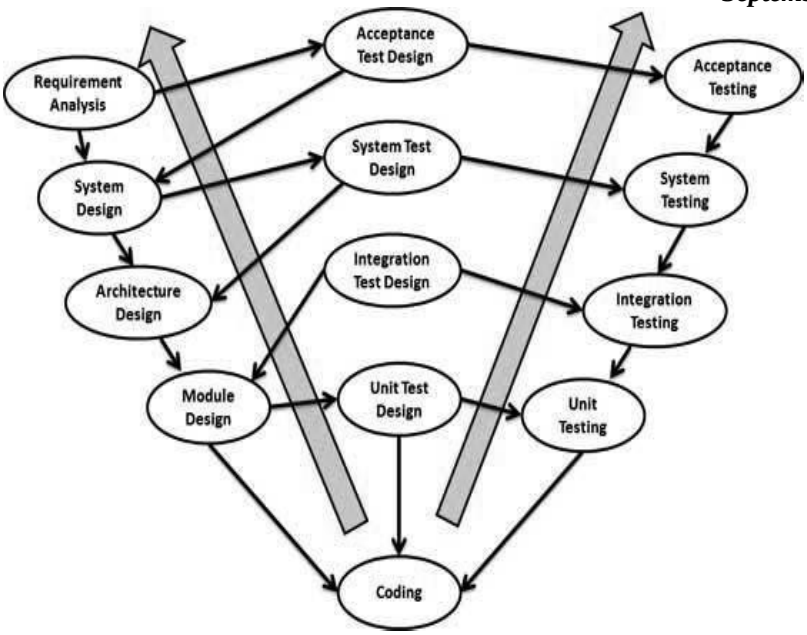

Fig. 10: V-Shaped Model

The V-Model has gained acceptance because of its simplicity and straightforwardness. Whereas, some developers believe it is too rigid for the evolving nature of information technology business environments.

Advantages:

- Well defined requirement specifications
- High amount of risk analysis involved at beginning.
- Good for critical projects.
- Early production.
- Easy to manage due to rigidity of model.
- Easy to understand.

Disadvantages:

- Not good model for object oriented projects.
- Not good for long and ongoing projects.
- Not suitable were requirements have high risk of changing.
- Costly model.
- Not suitable for small projects.
- Long implementation time

IV. CONCLUSION

After studying and completing the analyzing and comparison among various popular SDLC models available in the market. We conclude that a software development team will have to decide which model they use for their project as per customer requirements. All these models have their own pros and cons. However, in the existing commercial software development world, the fusion of all these methodologies is incorporated. Now a day's waterfall and spiral are the most commonly used in the software development process and the other models are used according to the requirements of the software products. An organization which needs to develop a project in a linear sequential way will use waterfall model. All software developers use the models according to the size and complexity of the software that is to be developed. All the developers and customers look at the low cost, risk, high quality and small cycle of time, so that the productivity and quality of the software product can be optimized. Timing is very crucial in software development. If a delay happens in the development phase, the market could be taken over by the competitor. So, there should be a tradeoff between the development time and the quality of the product. Every customer expect a bug free and user-friendly product to be developed in a shorter span of time. It is concluded that a blend of more than one model may be required in some software development product.

V. COMPARISON OF SDLC MODELS ON VARIOUS MERITS

	SDLC MODELS							
PARAMETERS/ MERITS	**WATERFALL**	**ITERATIVE**	**PROTOTYPE**	**RAD**	**INCREMENTAL**	**SPIRAL**	**BUILD & FIX**	**V-SHAPED**
Requirement	Yes	Yes	No	Yes	No	No	No	Yes

specifications								
User involvement in phases	Only at requirement analysis	Yes	High	Yes	Yes (intermediate)	High	Yes	No
Flexibility	No	Good	No	Yes	Low	Yes	Yes	Low
Overlapping phases	No	Yes	Yes	No	No	Yes	Yes	No
Risk Analysis	Only at beginning	Low	Yes	Low	No	Yes	No	Only at beginning
Expertise required	High	High	Medium	High	Medium	High	Medium	Medium
Reusabilty	Limited	Yes	Yes	Yes	Yes	Yes	Weak	Low
Simplicity	Simple	Simple	Low	Simple	Medium	Medium	Simple	Medium
Development time	Long	Medium	Long	Quick	Long	Long	Depends upon projects	Medium
Incorporation of changes	Difficult	Easy	Easy	Easy	Easy	Easy	Difficult	Difficult
Success rate	Low	High	Good	Good	Good	High	Medium	High
Cost	High	Low	High	Low	Low	High	Low	High
Cost control	Yes	No	No	No	No	Yes	Yes	Yes
Interface	Minimum	Critical	Critical			Critical		Minimum
Security	Vital	Limited	Weak			High		Limited
Resource control	Yes	Yes	No			Yes		Yes

REFERENCES

[1] Boehm B. W.(1988), A spiral Model of software development and enhancement. ISSN:0018-9162. Volume: 21, Issue: 5, on page(s): 61-72, May 1988.

[2] Youssef Bassil (2012), A Simulation Model for the Waterfall Software Development Life cycle Engineering &Technology, ISSN: 2049-3444, Vol.2, No.5, 2012.

[3] Barry Boehm, "Spiral Development: Experience, Principles, and Refinements", edited by Wilfred J.Hansen, 2000.

[4] A. M. Davis, H. Bersoff, E.R. Comer (1988), A Strategy for comparing Alternative Software Development Life Cycle Models, Journal IEEE Transactions on Software Engineering. Vol:14, Issue 10, 1988.

[5] Roger Pressman, Software engineering: A practitioner approach 0-07-365578-3,5th ed., TMH, 2001.

[6] Ian Somerville, Software Engineering ,Addison Wesley,9th ed,2010.

[7] Nabil Mohammed Ali Munassar, A.Govardhan (2010), A Comparison between Five Models of Software Engineering, IJCSI- International Journal of Computer Science Issues, Vol.: 7, Issue: 5, Sep, 2010.

[8] Craig Layman and Victor Basili (2003), Iterative and Incremental Development: A Brief History, IEEE Computer 2003.

[9] W. Royce (1970), Managing the Development of Large Software Systems, presented at the Proceedings of IEEE WESCON, 1970.

[10] A. Kaur, H. P. S. Dhami and J. Kaur (2009): Knowledge Management Approach For "I.T" Product Development, *Modern Management Practices & Information Technology Trends*, MMPITT-2009.

[11] Barry Boehm, "Spiral Development: Experience, Principles, and Refinements", edited by Wilfred J. Hansen, 2000.

[12] Jintao zeng, Jinzhong Li, Xiaohui Zeng, Wenlang Luo (2010), A Prototype System of Software Reliability Prediction and Estimation. *IITSI* 2010.

