Which model is best for the Software Project? "A Comparative Analysis of Software Engineering Models"

Anand Kr. Shukla
IICA Dept Invertis University, Bareilly

Archana Saxena
IICA Dept Invertis University, Bareilly

ABSTRACT

Software development methodology (SDLC) in software engineering is a framework that is used to structure plan and control the process of developing an important system. Choosing the right SDLC methodology for your project is as important to the success of the project. Today all live in the era of computer technology. This paper also identity the basic problems in the spiral, waterfall, and iterative models. These models has own advantages and disadvantages. This research paper compare all four models on the basis of some key points which will be helpful to develop a successful software project with the help of comparison any one can select his/her own type of model for his/her project development.

***Keywords*: Software Development Process, four models, Comparative analysis of model.**

1. INTORDUCTION

1.1 Spiral Model

Spiral Model can also be consider as incremental Model, spiral model consist four phases.

- Planning
- Risk Analysis
- Engineering/Development
- Evaluation

These four phases involves in iteration of any Software project which is just like a spiral so in this model, at the initial level Requirement Analysis or Planning has comes in which information has gathered and risk is estimated, in second phase i.e. **risk analysis phase** there is a process by which Risk and its alternative solution is identify and at the end of this phase a prototype is produced.

Software is developed in the third phase i.e. **engineering phase**, and also for performance checking software testing also be done at the end of the phase. At the last now time to evaluate output it can be allow the customer to evaluate the output of the project in **Evaluation Phase** before the pro ject continues to the next spiral.

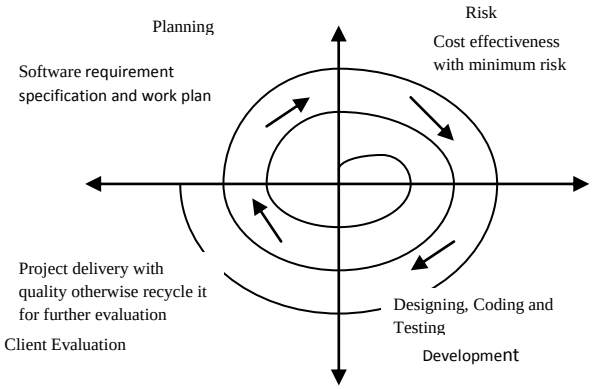

Fig.1 Spiral Model

1.1.1 Model description on following key points

- **Requirement specification**: because of its first phase Requirement Analysis, gathering requirement is easy and well understood.
- **Cost Estimation**: Cost and time estimations are also not very easy. Because in this model cost of Risk Analysis is greater than cost of the entire project.
- **Risk Analysis:** This model requires risk identification, its projection, risk assessment and risk management which is not an easy task.
- **Software Compatibility**: It is compatible with large projects.
- **Complexity**: The software projects which are developed by spiral model commonly are Big Project so the complexity is there.
- **Flexibility**: yes it is Flexible.
- **User involvement**: as because it is spiral so user involvement exist at every spiral Phase so we can say user involvement commonly very High.
- **Success-Rate**: as it is clear that this model is for big project not for smaller so the success-rate is very good, one more thing that because it have well define step and phases so performance checking , testing is involved at every spiral so there is less chance of failure which indicates it high Success Rate.

Features	Spiral Model
Requirement specification	*Well understood*
Cost Estimation	*High cost*
Risk Analysis	*Very high*
Software Compatibility	*For big project*
Complexity	*Complex*
Flexibility	*Flexible*
User involvement	*User involvement exist*
Success-Rate	*High success rate.*

1.2 Prototype Model

Prototyping model requi\re before development of actual Software so a well working prototype of the software system be built.

Prototyping model is just like system development method, in this method or model a Prototype of a system is created e.g. Have you seen a **Model** of any building/multi-stories building, Hotels, Apartments….etc, which gives an exact idea about that particular Building you can easily understand how will be the building will be so same in software development the concept of prototyping is works after testing. Prototype model is attractive and supportive for trial, finding errors and it helps to understand the condition and for future aspects development of the project

Prototyping Model categories as: 1. Proof-of-Principle Prototype (Model) 2. Form Study Prototype (Model) 3. User Experience Prototype (Model) 4. Visual Prototype (Model) 5. Functional Prototype (Model).

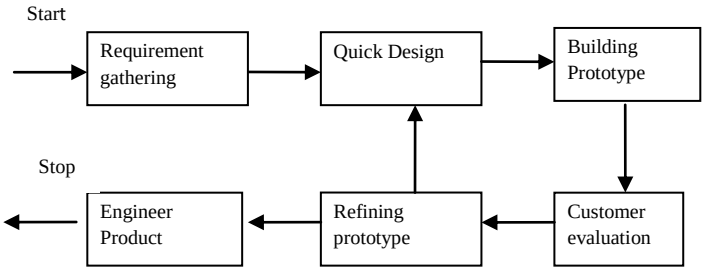

Fig.2 Prototype Model

1.2.1 Model description on following key points

- **Requirement specification**: in this model Requirements changes time to time
- **Cost Estimation**: as compare to waterfall and Iterative model thee cost of Prototype model is high but it is not as high as Spiral Model.
- **Risk Analysis:** Because this is based on prototype so Model has to be create before actual Model so all the testing has been done before launching the Model so Minimum Risk involve in this model.
- **Software Compatibility:** Prototype Model is suitable for big project as well as small project but small projects are much suitable.
- **Complexi**ty: The software projects which are developed by Prototype model is consider as complex as Complexity is higher.
- **Flexibility**: Because of prototype (in the form of Model) if any extension/modification/Updating require can easily be seen and understand so this is the quit Flexible Model.
- **User involvement**: Because of Model a user involvement is always higher because it is a Prototype so before launching Actual Model a prototype model has made and a user can also be interacts during the development and also after completion of this Model.

  For example if anyone want to buy a Flat in a Colony there are some sample flats for the customer and once customer agree to buy this then during the development customer involvement is approx necessary.
- **Success-Rate**: Because it is based on prototype so all the aspect and component of the original software project has been cover up it provide error free and successful environment which indicate less chance of failure so success Rate will be higher.

Key Points	Prototype Model
Requirement specification	*Not well understood*
Cost Estimation	*High cost*
Risk Analysis	*Minimum Risk*
Software Compatibility	*For both large and small project much suitable for small project.*
Complexity	*Complex*
Flexibility	*Flexible*
User involvement	*High User involvement*
Success-Rate	*High success rate.*

2. WATERFALL MODEL

The waterfall model also called the sequential model. In this model need to complete the previous phase to move to the

next phase. Once the phase is completed it's difficult to move to the previous phase.

The linear sequential model is the oldest and the most widely used paradigm for software engineering. At the end of each stage need determine the project move to the next stage of the development. This model has its own weaknesses like it's difficult to know exactly what is needed in each phase. Some clients are not confirming with their own requirements. Frequently they change their requirements and it's difficult in this model to make changes in the previous phase once the previous phase is completed. The software project must be adaptable, and spending considerable effort in design and implementation based on the idea that requirements will never change is neither adaptable nor realistic in these cases.

When to use this model? When the requirements are clear, technology is well known and the project is a small scale project.

In Royce's original waterfall model, the following phases are followed in order:

- Requirements specification
- Design
- Construction (implementation or coding)
- Integration and testing
- Installation
- Maintenance

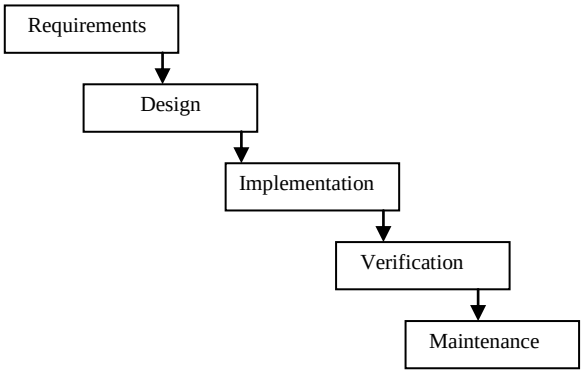

Fig.3 Waterfall Model

2.1 Model description on following key points

- **Requirements specification:** requirements cannot be managed well and has not been identified as the main reason for failure. When the system is put to use the customer discovers problems of early phases very late and system does not reflect current requirements. Not suitable for the projects where requirements are at a moderate to high risk of changing. During the development process, changes to requirements will be small enough that we can manage them without substantially rethinking or revising our plans.
- **Cost Estimation:** The waterfall model is connected to high costs and efforts. That is, it requires approval of many documents, changes are costly to implement, iterations take a lot of effort and rework, and problems are usually pushed to later phases
- **Risk analysis**: High amounts of risk and uncertainty.
- **Software compatibility: N**ot suitable for large projects because cost is high.
- **Complexity:** Easy to manage due to the rigidity of the model each phase has specific deliverables and a review process.
- **Flexibility:** Extremely hard to respond to changes.
- **User involvement**: Lack of opportunity for customer to provide feedback on the system.
- **Success Rate:** This model is so rigid and unrealistic. Business requires continues changes but in this model these changes were impossible, the development cycle was too long, system were over engineered and ended up with high cost.

Features	Waterfall Model
Requirement specification	*Not well understood*
Cost Estimation	*Difficult to estimate cost*
Risk Analysis	*Risk not cover*
Software Compatibility	*For big project*
Complexity	*Complex*
Flexibility	*Poor Flexible*
User involvement	*No user involvement exist*
Success-Rate	*No high success rate.*

3. ITERATIVE MODEL

The incremental model has same phases that are in waterfall model. But it is iterative in nature. In iterative model, iterative process starts with a simple implementation of a small set of the software requirements and iteratively enhances the evolving versions until the complete system is implemented. This model is repeated, producing a new version of the software at the end of each iteration of the model. At each iteration, design modifications are made and new functional capabilities are added. The basic idea behind this method is to develop a system through repeated cycles and in smaller portion at a time.

The iterative process adopts the logical sequence of events of the Waterfall model but it follows the full lifecycle several times within a single project. Each iteration goes through the activities of Requirements, Analysis, Design, Coding, Integration and Testing and produces production-quality software at the end of the iteration.

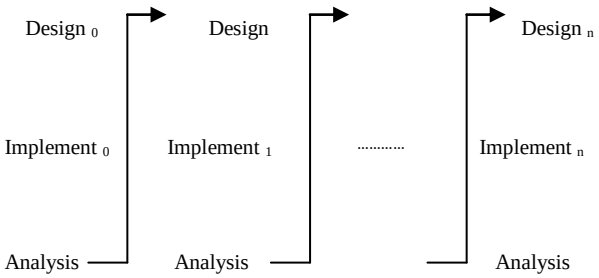

Fig.4 Iterative Model

3.1 Model description on following key points

- **Requirements Specification:** System architecture or design issues may arise because not all requirements are gathered in the beginning of the entire life cycle. Not suitable for smaller projects.
- **Cost Estimation:** Less costly to change the scope/requirements
- **Risk analysis:** Risks are identified and resolved during iteration. The risk to the client is limited because after each iteration some working software is delivered to client.
- **Software compatibility:** more resources are required to complete the project.
- **Complexity:** less complex
- **Flexibility:** The businesses are changing rapidly today, they never really know the "complete" requirements for the software, and there is a need to constantly add new capabilities to the software to adapt the business to changing situations. Iterative process allows this.
- **User involvement:** More users friendly because after iteration we are getting the one working model.
- **Success Rate:** constantly we can add new features in the working software.

Features	Iterative Model
Requirement specification	*Well understood*
Cost Estimation	*less costly*
Risk Analysis	*less risk*
Software Compatibility	*For big project*
Complexity	*less complex*
Flexibility	*Flexible*
User involvement	*User involvement exist*
Success-Rate	*High success rate.*

3.2 Comparative Analysis:

Features	Spiral Model	Prototype Model	Waterfall Model	Iterative Model
Requirement specification	Well understood	not well understood	Not well understood	Well understood
Cost Estimation	High cost	High cost	difficult to estimate cost	less costly
Risk Analysis	Very high	Minimum Risk	risk not cover	less risk
Software Compatibility	For big project	Complex	For big project	For big project
Complexity	Complex	Flexible	Complex	less complex
Flexibility	Flexible	For both large and small project much suitable for small project.	Poor flexible	Flexible
User involvement	User involvement exist	High User involvement	No user involvement exist	User involvement exist
Success-Rate	High success rate.	High success rate.	no high success rate.	High success rate.

4. CONCLUSION

After analysis on various parameter (Features), it has found that the big companies used water fall model for their project. Iterative water fall model overcome the drawback of original waterfall model. It allows feedback to proceeding stage. Online system develop by Prototype model for transaction processing.. Spiral model is used for development of large, complicated and expensive projects like scientific Projects .Since spiral model approach enables the project term to address the highest risk at the lowest total cost.

5. References

[1] Bowman, M., Debray, S. K., and Peterson, L. L. 1993. Reasoning about naming systems. .

[2] Dyer, Mike, *The Cleanroom Approach to Quality Software Development*, 1993

[3] *Summary of Changes to DoD-STD-2167A and DoD-STD-7935A resulting in MIL-STD-SDD*, Executive Summary, p. 1, December 1992.

[4] Guidelines for Successful Acquisition and Management of Software Intensive Systems: Weapons Systems, Command and Control Systems, Management Information Systems, September 1994.

[5] Morris, Randy, and Boyd Tidwell, "Object-Oriented Ada Using State Controlled Implementation," *CrossTalk*, June 1994, p. 17.

[6] *Software Management Guide, Vol. I*, Software Technology Support Center, October 1993, p. 23.

[7] Blum, Bruce I., Software Engineering: A Holistic View, 1992.

[8] Boehm, Barry, "A Spiral Model of Software Development and Enhancement," from Proceedings of an International Workshop on the Software Process and Software Environments, 1985.

[9] Booch, Grady, *Software Engineering with Ada*, 1994,

[10] Cobb, Richard, Ara Kouchakdjian, and Harlan D. Mills, *The Cleanroom Engineering Software Development Process*, Software Technology for Adaptable, Reliable Systems (STARS) Program,

[11] Coad, Peter, and Edward Yourdon, *Object-Oriented Analysis*, 1991