Popularity of Agile over Software Models

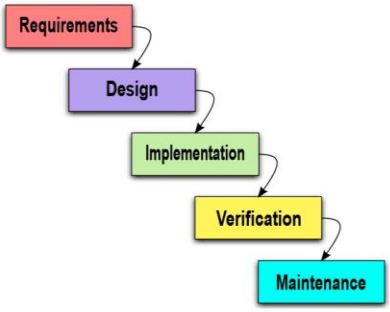
Dr. Joginder Singh Cheema
HOD, Department of Computer Applications,
Baba Budha College, Bir Sahib, Tarntaran

Abstract: In this paper, we can study the standard life cycle models namely waterfall model, prototype model, V-shaped model, spiral model, incremental model, rapid application development and Agile Development Model. We can also discuss the advantages and disadvantages of all software models and situations where we can use these models. In this paper, we can compare the agile method to traditional methods.

Keywords: **SDLC, Agile model, Extreme programming**

I. INTRODUCTION

[1] In order to achieve a structured and controllable software development effort, several software development models are being used. There are many accepted models in development process, for example, the Spiral Model, the Waterfall-model, the V-Model and the Agile Model, which are popular today. All this models define a systematic way to accomplish an orderly way of working during the project. Testing appears in each of these life cycle models, but with very different meanings and different extend. Some methods work better for specific type of projects, but in the final result, the most important factor for the project success may be how closely particular plan was followed. According to my experience, in some projects we can see some models work in parallel; the reason is a very dynamic project aids. For example: a project chooses the Spiral Model in advanced, but during the developing process, it becomes clear that the project should update some of its requirements because of customer change of design. In this example we can see that Spiral Model has interaction with some indications of Agile Model.



II. RELATED WORKS

A. Waterfall-Model

[2] [3] The original SDLC model was the Waterfall-model. This model is very simple to understand and very well known in development process. The Waterfall Model was first Process Model to be introduced. It is also referred to as a **linear-sequential life cycle model**. It is very simple to understand and use. In a waterfall model, each phase must be completed fully before the next phase can begin.

When to use the waterfall

- Project is short.
- Technology is understood.
- When the requirements are very well known, clear and fixed.
- Product definition is stable.
- There are no ambiguous requirements.

Advantages of waterfall model:

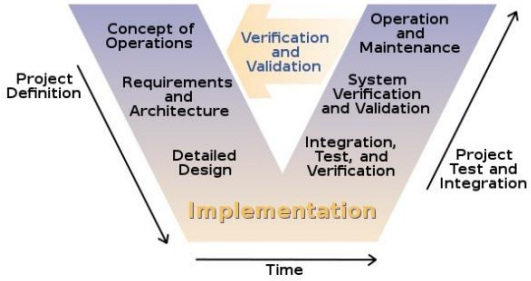
- This model is simple and easy to understand and use.
- It is easy to manage due to the rigidity of the model – each phase has specific deliverables and a review process.
- In this model phases are processed and completed one at a time. Phases do not overlap.
- Waterfall model works well for smaller projects where requirements are very well understood.

Disadvantages of waterfall model:

- Once an application is in the **testing** stage, it is very difficult to go back and change something that was not well-thought out in the concept stage.
- No working software is produced until late during the life cycle.
- High amounts of risk and uncertainty.
- Not a good model for complex and object-oriented projects.
- Poor model for long and ongoing projects.
- Not suitable for the projects where requirements are at a moderate to high risk of changing.

B. V-Model

[2] V- Model means Verification and Validation model. Just like the **waterfall model**, the V-Shaped life cycle is a sequential path of execution of processes. Each phase must be completed before the next phase begins. **V-Model** is one of the **many software development models**. Testing of the product is planned in parallel with a corresponding phase of development in **V-model**.

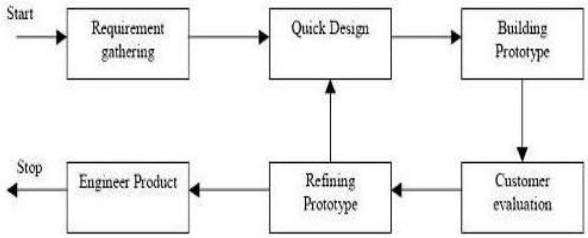


Advantages of V-model:

- Simple and easy to use.
- Testing activities like planning, **test designing** happens well before coding. This saves a lot of time. Hence higher chance of success over the waterfall model.
- Proactive defect tracking – that is defects are found at early stage.
- Avoids the downward flow of the defects.
- Works well for small projects where requirements are easily understood.

Disadvantages of V-model:

- Very rigid and least flexible.
- Software is developed during the implementation phase, so no early prototypes of the software are produced.



When to use Prototype model

Advantages of Prototype model:

- Users are actively involved in the development
- Since in this methodology a working model of the system is provided, the users get a better understanding of the system being developed.
- Errors can be detected much earlier.
- Quicker user feedback is available leading to better solutions.
- Missing functionality can be identified easily
- Confusing or difficult functions can be identified Requirements validation, Quick implementation of, incomplete, but functional, application.

Disadvantages of Prototype model:

- Leads to implementing and then repairing way of building systems.

When to use the V-model:

- Medium sized projects where requirements are clearly defined and fixed.
- Ample technical resources are available with needed technical expertise.
- If any changes happen in midway, then the test documents along with requirement documents has to be updated.

C. Prototype model

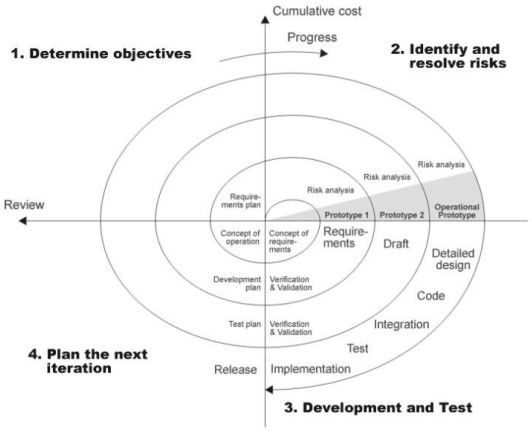
[2] [4] The basic idea in **Prototype model** is that instead of freezing the requirements before a design or coding can proceed, a throwaway prototype is built to understand the requirements. This prototype is developed based on the currently known requirements. Prototype model is a **software development model**.

The prototype is usually not complete systems and many of the details are not built in the prototype. The goal is to provide a system with overall functionality.

- When the desired system needs to have a lot of interaction with the end users.
- Typically, online systems, web interfaces have a very high amount of interaction with end users, are best suited for Prototype model.
- They are excellent for designing good human computer interface systems.
- Practically, this methodology may increase the complexity of the system as scope of the system may expand beyond original plans.
- Incomplete application may cause application not to be used as the full system was designed Incomplete or inadequate problem analysis.

D. Spiral Model

[2] [5] The spiral model is similar to the **incremental model**, with more emphasis placed on risk analysis. The spiral model has four phases: Planning, Risk Analysis, Engineering and Evaluation. A software project repeatedly passes through these phases in iterations (called Spirals in this model). The baseline spiral, starting in the planning phase, requirements are gathered and risk is assessed. Each subsequent spiral builds on the baseline spiral. Its one of the **software development models** like **Waterfall**, **Agile**, **V-Model**.

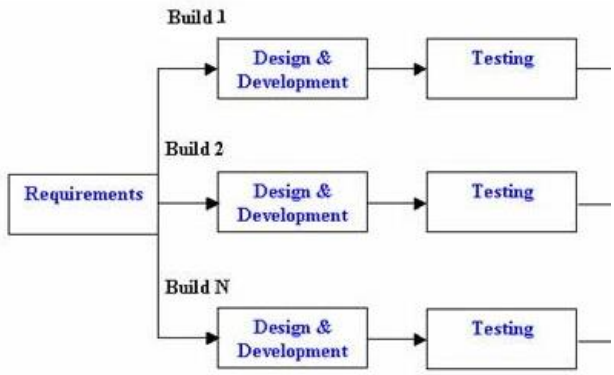


Advantages of Spiral model

- High amount of risk analysis hence, avoidance of Risk is enhanced.
- Good for large and mission-critical projects.
- Strong approval and documentation control.
- Additional Functionality can be added at a later date.
- Software is produced early in the **software life cycle**.

Disadvantages of Spiral model

- Can be a costly model to use.
- Risk analysis requires highly specific expertise.



Advantages of Incremental model

- Generates working software quickly and early during the software life cycle.
- This model is more flexible – less costly to change scope and requirements.
- It is easier to test and debug during a smaller iteration.
- In this model customer can respond to each built.
- Lowers initial delivery cost.
- Easier to manage risk because risky pieces are identified and handled during it'd iteration.

Disadvantages of Incremental model

- Needs good planning and design.
- Needs a clear and complete definition of the whole system before it can be broken down and built incrementally.
- Total cost is higher than **waterfall**.

F. RAD model

[2] [6] RAD model is Rapid Application Development model. It is a type of **incremental model**. In RAD model the components or functions are developed in parallel as if they were mini projects. The developments are time boxed,

When to use Spiral model

- When costs and risk evaluation is important
- For medium to high-risk projects
- Long-term project commitment unwise because of potential changes to economic priorities
- Users are unsure of their needs
- Requirements are complex
- New product line
- Significant changes are expected (research and exploration)

- Project's success is highly dependent on the risk analysis phase.
- Doesn't work well for smaller projects.

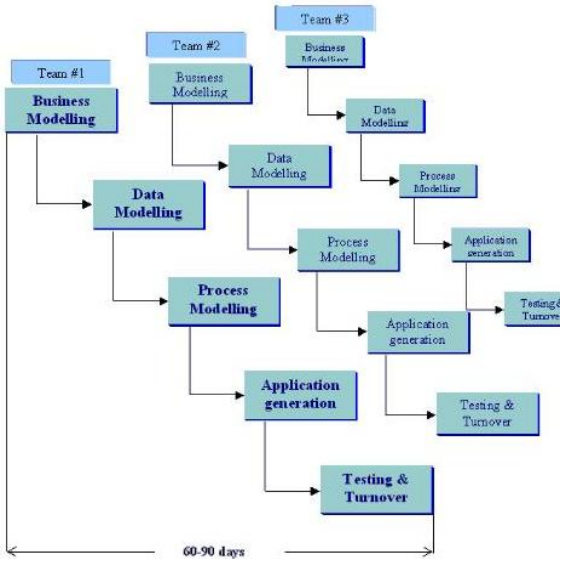
E. Incremental model

[2] In incremental model the whole requirement is divided into various builds. Multiple development cycles take place here, making the life cycle a **"multi-waterfall" cycle**. Cycles are divided up into smaller, more easily managed modules. Incremental model is a type of software development model like **V-model**, **Agile model** etc.

When to use the Incremental model:

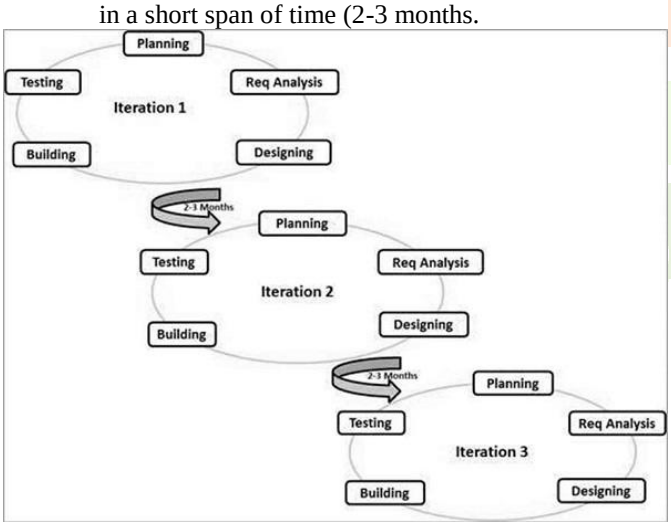
- This model can be used when the requirements of the complete system are clearly defined and understood.
- Major requirements must be defined; however, some details can evolve with time.
- There is a need to get a product to the market early.
- A new technology is being used
- Resources with needed skill set are not available
- There are some high risk features and goals.

delivered and then assembled into a working prototype. This can quickly give the customer something to see and use and to provide feedback regarding the delivery and their requirements.



When to use RAD model

- RAD should be used when there is a need to create a system that can be modularized in 2-3 months of time.
- It should be used if there's high availability of designers for modeling and the budget is high enough to afford their cost along with the cost of automated code generating tools.
- RAD **SDLC model** should be chosen only if resources with high business knowledge are available and there is a need to produce the system in a short span of time (2-3 months.



When to use Agile model

- New changes can be implemented at very little cost because of the frequency of new increments that are produced.

The Agile thought process had started early in the software development and started becoming popular with time due to its flexibility and adaptability.

Following are the Agile Manifesto principles −

- **Individuals and interactions** − In Agile development, self-organization and motivation are important, as are interactions like co-location and pair programming.
- **Working software** − Demo working software is considered the best means of communication with the customers to understand their requirements, instead of just depending on documentation.
- **Customer collaboration** − As the requirements cannot be gathered completely in the beginning of the project due to various factors, continuous customer interaction is very important to get proper product requirements.
- **Responding to change** − Agile Development is focused on quick responses to change and continuous development.

Agile development model is also a type of **Incremental model**. Software is developed in incremental, rapid cycles.

- To implement a new feature the developers need to lose only the work of a few days, or even only hours, to roll back and implement it.
- Unlike the **waterfall model** in agile model very limited **planning** is required to get started with the project. Agile assumes that the end users' needs are ever changing in a dynamic business and IT world. Changes can be discussed and features can be newly effected or removed based on feedback. This effectively gives the customer the finished system they want or need.
- Both system developers and stakeholders alike, find they also get more freedom of time and options than if the software was developed in a more rigid sequential way. Having options gives them the ability to leave important decisions until more or better data or even entire hosting programs are available; meaning the project can continue to move forward without fear of reaching a sudden standstill.

Advantages of the RAD model

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

Disadvantages of RAD model

- 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.

III. AGILE DEVELOPMENT

[2][7] Agile model believes that every project needs to be handled differently and the existing methods need to be tailored to best suit the project requirements. In Agile, the tasks are divided to time boxes (small time frames) to deliver specific features for a release.

Iterative approach is taken and working software build is delivered after each iteration. Each build is incremental in terms of features; the final build holds all the features required by the customer.

This results in small incremental releases with each release building on previous functionality. Each release is thoroughly **tested** to ensure **software quality** is maintained. It is used for time critical applications. Extreme Programming (XP) is currently one of the most well known agile **development life cycle model**.

The advantages of the Agile Model are as follows:

- Is a very realistic approach to software development.
- Promotes teamwork and cross training.
- Functionality can be developed rapidly and demonstrated.
- Resource requirements are minimum.
- Suitable for fixed or changing requirements
- Delivers early partial working solutions.
- Good model for environments that change steadily.
- Minimal rules, documentation easily employed.
- Enables concurrent development and delivery within an overall planned context.
- Little or no planning required.
- Easy to manage.
- Gives flexibility to developers.

The disadvantages of the Agile Model are as follows:

- Not suitable for handling complex dependencies.
- More risk of sustainability, maintainability and extensibility.
- An overall plan, an agile leader and agile PM practice is a must without which it will not work.
- Strict delivery management dictates the scope, functionality to be delivered, and adjustments to meet the deadlines.
- Depends heavily on customer interaction, so if customer is not clear, team can be driven in the wrong direction.
- There is a very high individual dependency, since there is minimum documentation generated.
- Transfer of technology to new team members may be quite challenging due to lack of documentation.

IV. EXTREME PROGRAMMING IN AGILE DEVELOPMENT

[2] The primary driver of any software project should be the problem the software is aimed at solving. If the design of the program is too large and expensive it may become the driver of the project, either because it is too unwieldy to change or too much has been invested in it. Similarly, a requirements gathering process may become the main driver of the project. **Extreme Programming** insists on the fundamental importance of keeping the software problem to be solved as the focus of the development effort. **Extreme Programming** is also designed with that fundamental observation in mind.

A. Extreme Programming

Extreme Programming maintains that tests should be created as the beginning of the code and that the code is written to pass those tests. As well, the client who has a problem to be solved with software defines criteria to create Acceptance Tests. The Extreme Programming aim of maintaining tight feedback and iteration cycles among test, code, and design offer a viewpoint on which to dispense with requirements entirely. The framers of the software product simply have to create a set of tests that the software must satisfy sets of tests to code to and a set of tests to define acceptance criteria (this can even go so far as to include integration, system, and performance tests).

The classical view of **Software Testing** maintains that tests are designed to verify that the software satisfies (or fails to satisfy) given requirements. In fact, some Automated Test Management Suites are built on this premise (e.g. Mercury Test Director). This easy to understand and implement the concept of a one to one relationship between requirements and tests are unfortunately invalid. Software tests are designed around models of the actual software. Similarly, requirements define a model that the software is supposed to adhere to.

V. AGILE VERSUS TRADITIONAL SDLC MODELS

[8][9] [10]Agile is based on the **adaptive software development methods**, whereas the traditional SDLC models like the waterfall model is based on a predictive approach. Predictive teams in the traditional SDLC models usually work with detailed planning and have a complete forecast of the exact tasks and features to be delivered in the next few months or during the product life cycle.

Predictive methods entirely depend on the **requirement analysis and planning** done in the beginning of cycle. Any changes to be incorporated go through a strict change control management and prioritization.

Agile uses an **adaptive approach** where there is no detailed planning and there is clarity on future tasks only in respect of what features need to be developed. There is feature driven development and the team adapts to the changing product requirements dynamically. The product is tested very frequently, through the release iterations, minimizing the risk of any major failures in future.

Customer Interaction is the backbone of this Agile methodology, and open communication with minimum documentation are the typical features of Agile development environment. The agile teams work in close collaboration with each other and are most often located in the same geographical location.

VI. CONCLUSION AND FUTURE SCOPE

We can study number of software models, their advantages, disadvantages and where to use which software model according to its features. In this paper, we can explain the agile development process in a very efficient manner. On the basis of such capabilities for a specific software project one can choose which of such software development life cycle models ought to be selected for the specific project. This research work can make the strategy of picking the SDLC model easy& therefore will prove to be extremely effective for software industry.

REFERENCES

[1] Sharma, Anubha, Manoj Kumar, and Sonali Agarwal. "A Complete Survey on Software Architectural Styles and Patterns." Procedia Computer Science 70 (2015): 16-28.

[2] Ian Sommerville,"Software Engineering", Addison Wesley,7th edition, 2004.

[3] Steve Easterbrook, "Software lifecycles", University of Toronto Department of Computer Science, 2001.

[4] Karlm, "Software Lifecycle Models", KTH, 2006.

[5] Langer, Arthur M. "System Development Life Cycle (SDLC)." In Analysis and Design of Information Systems, pp. 10-20. Springer London, 2008.

[6] Mexican International Conference on Computer Science IEEE Computer Society Washington, DC, USA, 2009.

[7] In Confluence The Next Generation Information Technology Summit (Confluence), 2014 5th International Conference-, pp. 221-226. IEEE, 2014.

[8] A. M. Davis, H. Bersoff, E. R. Comer, "A Strategy for Comparing Alternative Software Development Life Cycle Models", Journal IEEE Transactions on Software Engineering ,Vol. 14, Issue 10, 1988.

[9] Niazi, Mahmood, Muhammad Ali Babar, and June M. Verner. "Software Process Improvement barriers: A cross-cultural comparison." Information and software technology 52, no. 11 (2010): 1204-1216.

[10] Dybå, Tore. "An empirical investigation of the key factors for success in software process improvement." Software Engineering, IEEE Transactions on31, no. 5 (2005): 410-424.