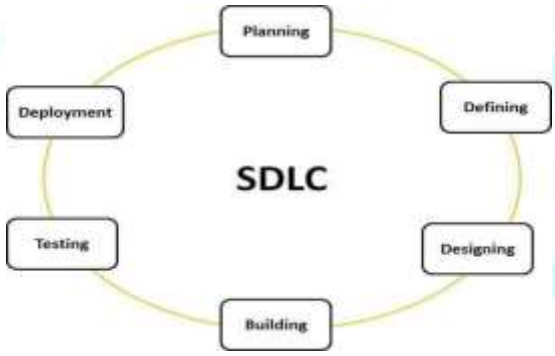
Various Software Development Life Cycle Models

Sahil Jindal, Puneet Gulati, Praveen Rohilla
Dronacharya College of Engineering, India

Abstract: An SDLC model is a conceptual framework describing different activities in a software development project from planning to maintenance. This process is associated with several models, each including a variety of tasks and activities. Software development is a cumbersome activity requiring proper identification of requirements, their implementation, and software deployment. However, the activities do not end there. After the distribution of the software, proper maintenance has to be provided in a timely manner. So to carry out the software development process efficiently we require certain models which help us to carry out our work properly.
Keyword: models, waterfall, RAD, spiral, etc**.**

INTRODUCTION

SDLC is a process followed for a software project, within a software organization. It consists of a detailed plan describing how to develop, maintain, replace and alter or enhance specific software. The life cycle defines a methodology for improving the quality of software and the overall development process



- Planning and Requirement Analysis : Requirement analysis is the most important and fundamental stage in SDLC. It is performed by the senior members of the team with inputs from the customer, the sales department, market surveys and domain experts in the industry. This information is then used to plan the basic project approach and to conduct product feasibility study in the economical, operational, and technical areas.

- Defining Requirements : Once the requirement analysis is done the next step is to clearly define and document the product requirements and get them approved from the customer or the market analysts. This is done through **'SRS' –** Software Requirement Specification document which consists of all the product requirements to be designed and developed during the project life cycle.

- Designing the product architecture : SRS is the reference for product architects to come out with the best architecture for the product to be developed. Based on the requirements specified in SRS, usually more than one design approach for the product architecture is proposed and documented in a DDS - Design Document Specification. This DDS is reviewed by all the important stakeholders and based on various parameters as risk assessment, product robustness, design modularity , budget and time constraints , the best design approach is selected for the product.

- Building or Developing the Product : In this stage of SDLC the actual development starts and the product is built. The programming code is generated as per DDS during this stage. If the design is performed in a detailed and organized manner, code generation can be accomplished without much hassle.

- Testing the Product : This stage is usually a subset of all the stages as in the modern SDLC models, the testing activities are mostly involved in all the stages of SDLC. However this stage refers to the testing only stage of the product where products defects are reported, tracked, fixed and retested, until the product reaches the quality standards defined in the SRS.

- Deployment in the Market and Maintenance : Once the product is tested and ready to be deployed it is released formally in the appropriate market. Sometime product deployment happens in stages as per the organizati**ons'** business strategy. The product may first be released in a limited segment and tested in the real business environment (UAT- User acceptance testing).

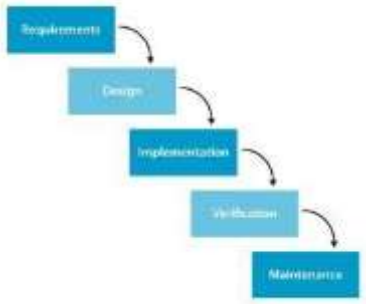
II. SDLC MODELS

There are various software development life cycle models defined and designed which are followed during software development process. These models are also referred as "Software Development Process Models". Each process model follows a Series of steps unique to its type, in order to ensure success in process of software development. Following are the most important and popular SDLC models

- Waterfall Model
- Prototype Model
- Iterative Model
- RAD
- Spiral Model

WATERFALL MODEL

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 before the next phase can begin and there is no overlapping in the phases. Waterfall model is the earliest SDLC approach that was used for software development .The waterfall Model illustrates the software development process in a linear sequential flow; hence it is also referred to as a linear-sequential life cycle model. This means that any phase in the development process begins only if the previous phase is complete. In waterfall model phases do not overlap.



Advantages:-

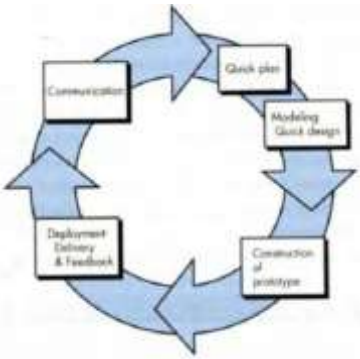
- Simple and easy to understand and use.
- Easy to manage due to the rigidity of the model – each phase has specific deliverables and a review process.
- Phases are processed and completed one at a time Works well for smaller projects where requirements are very well understood.
- Clearly defined stages.
- Well understood milestones.
- Easy to arrange tasks.
- Process and results are well documented.

Disadvantages:-

- 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.
- It is difficult to measure progress within stages.
- Cannot accommodate changing requirements.
- No working software is produced until late in the life cycle.
- Adjusting scope during the life cycle can end a project.

PROTOTYPE MODEL

A prototype is a working model that is functionally equivalent to a component of the product. In many instances the client only has a general view of what is expected from the software product . In such a scenario where there is an absence of detailed information regarding the input to the system, the processing needs and the output requirements, of the project are shown by the prototype model.



Advantages:-

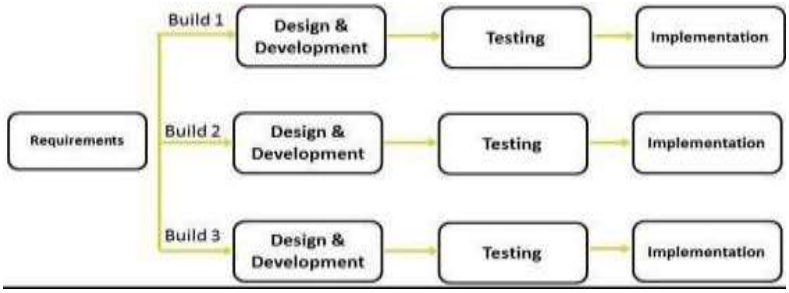
With reduced time and costs, Prototyping can improve the quality of requirements and specifications provided to developers. Prototyping requires user involvement and allows them to see and interact with a prototype allowing them to provide better and more complete feedback and specifications.

Disadvantages:-

The focus on a limited prototype can distract developers from properly analyzing the complete project. User can begin to think that a prototype, intended to be thrown away, it is actually a final system that merely needs to be finished or polished developer attachment to prototype.

ITERATIVE MODEL

Iterative process starts with a simple implementation of a subset of the software requirements and iteratively enhances the evolving versions until the full system is implemented. 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 (iterative) and in smaller portions at a time.



Advantages:-

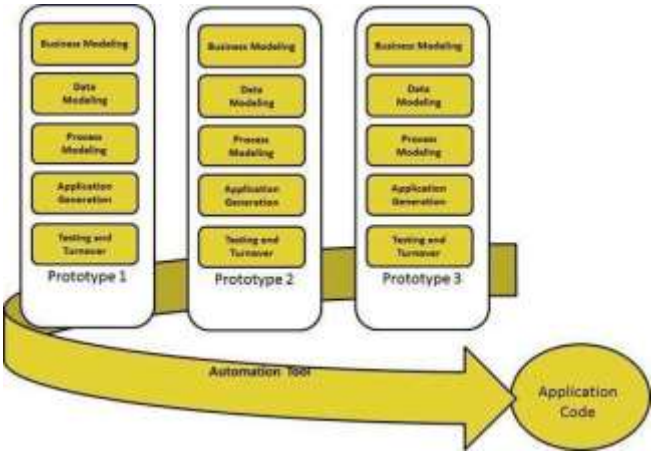
- Risk analysis is better.
- It supports changing requirements.
- Initial Operating time is less.
- Better suited for large and mission-critical projects.
- Some working functionality can be developed quickly and early in the life cycle.
- Results are obtained early and periodically.
- Parallel development can be planned.
- Progress can be measured.
- Less costly to change the scope/requirements.
- Testing and debugging during smaller iteration is easy.

Disadvantages:-

- More resources may be required.
- Although cost of change is lesser but it is not very suitable for changing requirements.
- More management attention is required.
- System architecture or design issues may arise because not all requirements are gathered in the beginning of the entire life cycle.
- Defining increments may require definition of the complete system.
- Not suitable for smaller projects.
- Management complexity is more.
- End of project may not be known which is a risk.
- Highly skilled resources are required for risk analysis**.**

RAD MODEL

The RAD (Rapid Application Development) model is based on prototyping and iterative development with no specific planning involved. The process of writing the software itself involves the planning required for developing the product. Rapid Application development focuses on gathering customer requirements through workshops or focus groups, early testing of the prototypes by the customer using iterative concept, reuse of the existing prototypes (components), continuous integration and rapid delivery.



Advantages:-

- Changing requirements can be accommodated.
- Progress can be measured.
- Iteration time can be short with use of powerful RAD tools.
- Productivity with fewer people in short time.
- Reduced development time.
- Increases reusability of components.
- Quick initial reviews occur.
- Encourages customer feedback.

Disadvantages:-

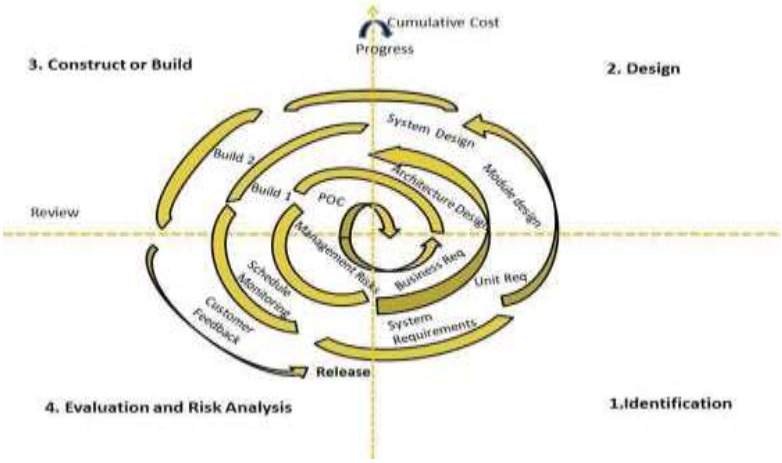
- Dependency on technically strong team members for identifying business requirements.
- Only system that can be modularized can be built using RAD.
- Requires highly skilled developers/designers.
- High dependency on modeling skills.
- Management complexity is more.
- Suitable for systems that are component based and scalable.
- Requires user involvement throughout the life cycle.

PIRAL MODEL

Spiral model is a combination of iterative development process model and sequential linear development model i.e. waterfall model with very high emphasis on risk analysis. It allows for incremental releases of the product, or incremental refinement through each iteration around the spiral.

The spiral model has 4 phases:

- Identification
- Design
- Construct or Build
- Evaluation and Risk Analysis



Advantages:-

- Changing requirements can be accommodated.
- Allows for extensive use of prototypes
- Requirements can be captured more accurately.
- Users see the system early.

- Development can be divided into smaller parts and more risky parts can be developed earlier which helps better risk management.

Disadvantages:-

- Management is more complex.
- End of project may not be known early.
- Not suitable for small or low risk projects and could be expensive for small projects.
- Process is complex
- Spiral may go indefinitely.
- Large number of intermediate stages requires excessive documentation.

III. CONCLUSION

The above study gives a clear understanding that various SDLC models when employed for developing different software then they may generate successful results owing to the fact that circumstances, resources, requirements, etc do vary for developer side as well as for client side. Employing a specified SDLC model for certain type software could not be determined in exact terms. Employing of any SDLC model is entirely a matter of choice which is dependent on the developer side. Thus comparing any of the SDLC models on some mathematical basis is almost impossible; they could be compared only on theoretical basis.

IV. References

- Software Methodologies Advantages & disadvantages of various SDLC models.mht
- Roger S. Pressman, Software Engineering: A Practitioner's Approach http://www.selectbs.com/analysis-and-design/what-is-the-waterfall-model
- http://www.softmart.ru/pdf/Tracker_Release_Highlights_8.pdf
- http://www.qpmg.com/main_pdts.html
- http://www.qpmg.com/main_pdts.html#FPworkbench