**Software Development Life Cycle (SDLC) –Riduwan**

The Software Development Life Cycle (SDLC) is a process took after for a software project, inside a software organization. It comprises of a detailed plan showing developers how to develop, maintain, replace and alter or enhance a particular software. The life cycle characterizes an approach for improving the quality of software and the overall development process.

**Phases of Software Development Life Cycle**

**Requirements Engineering Phase**

* Discover what the user, which represents the client or the customer wants the software to do.

**Pitfalls**

* Fuzzy, hard to specify precisely
* Incomplete requirements
* Unstable requirements
* Misconception user goals

**Analysis and Design Phase**

Analysis: Uses a blend of text and diagrammatic structures to depict requirements.

Outline: Delivers a representation of an entity that will be built. An example includes architectural design, user interface design and database design.

**Pitfalls**

* Not meeting the requirements
* Compromised design because of cost and time
* Different designs
* Explosion of derived requirements
* Poor, ad-hoc design

**Implementation Phase**

* The process whereby detailed designs are converted into instructions written in the programming language.

**Pitfalls**

* Tied down to certain technology
* Future improvements not considered
* Distinctive developers and teams
* Bad compromises made due to deadline
* Clashing decision between cost and time

**Testing Phase**

* Retrieve as many errors as possible
* Amend the errors
* Track the errors to comprehend their causes and any patterns that may exist.
* Revalidate the stability of the solutions, including guaranteeing that the amendment of one error does not lead to another error elsewhere.

**Pitfalls**

* The test was done by people who are not developers.
* Insufficient testing which could lead to errors.
* Testing done late in the project which could make the project fail to run due to many errors.

**Deployment Phase**

The final phase of Software Development Life Cycle (SDLC) whereby the final software is implemented in the production environment. The application is distributed among a group of selected customers before the date of the official release. The application that is distributed to the customers may consist of training.

**Pitfalls**

* Inaccurate expectation of production environment
* Not trying out the production environment
* Version control as version is not up to date.

**Waterfall Model**

**Traditional Waterfall Approach**

****

**Advantages Of Waterfall**

The advantages of waterfall model is that it separates complex task into smaller and thus more reasonable task. For every task creates a very much deliverable documentation. Adding on, it is simple to control and monitor since users only need to manage one activity at once.

**Disadvantages Of Waterfall**

The disadvantage of waterfall model is that no working software is delivered until late amid the life cycle. The amount of risk and uncertainty is high and it is not suitable for projects where requirements are at a moderate to high risk of changing.

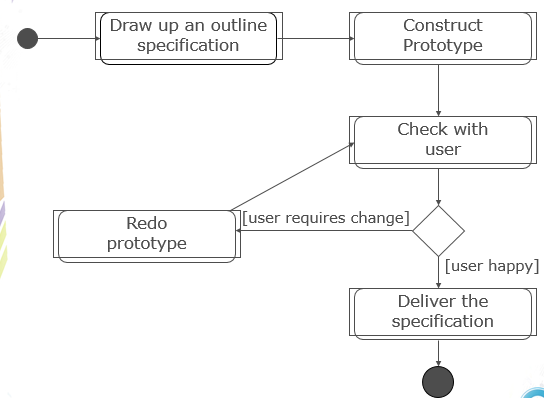
**Prototyping Model**

The prototyping model also known as the system development method (SDM) in which a prototype that represents an early approximation of a final product which is built, tested and afterward improved until an acceptable prototype is finally achieved from which the complete product can now be produced. This model works best in situations whereby the project requirements are still uncertain in detail ahead of time. This is a trial-and-error process between developers and users.

**Throwaway Prototyping**

Throwaway prototyping consists of a small of the system which is developed and after that given to the end user to try out and evaluate. The user will then provide feedback which can quickly be incorporated into the development of the main system. The prototype is then disposed or thrown away. The purpose of throw-away prototyping is to guarantee that the system requirements are validated and they are clearly understood.

**Stages of throwaway prototyping**



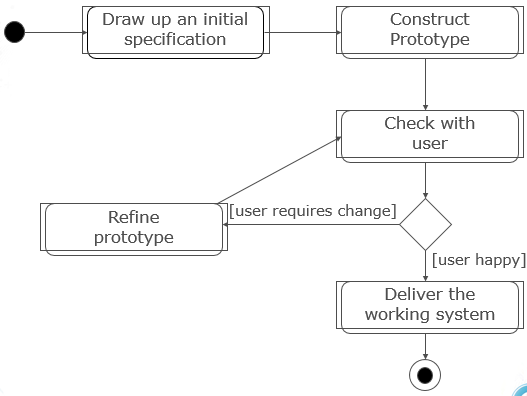
**Techniques for Throwaway Prototyping**

* Use a high level-language such as Visual Basic, drag and drop to create a Graphical User Interface (GUI)
* Reuse components from the existing systems.
* Ignore error handling such as “no validation input”, “no exception handling”.
* Omit features like security and logging
* Ignore functions of doing a mock-up

**Evolutionary Prototyping**

The initial prototype is presented to the user whereby the users will provide feedbacks and recommendations to improvements. These are actioned by the developer who then shows a more refined model. The user once again gives their feedback. The procedure is then repeated. Therefore, at every stage the model “evolves” towards the finalized system.

**Stages Of Evolutionary Prototyping**



**Advantages of prototyping**

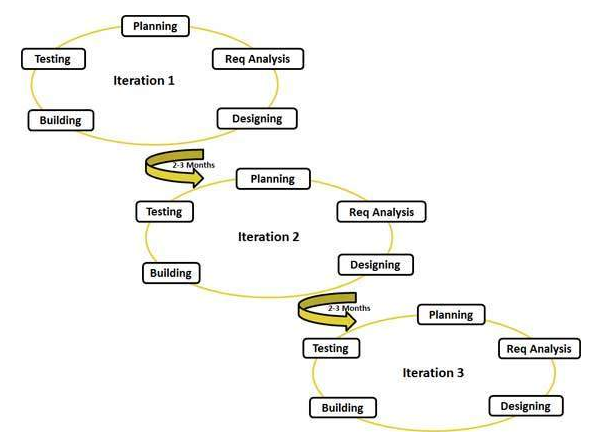
* Users are actively involved in the development
* Since in this methodology a working model of the system is provided, the users get a better understanding of the system being developed.
* Errors can be detected much earlier.
* Quicker user feedback is available leading to better solutions.

**Disadvantages of Prototyping**

* Leads to implementing and then repairing way of building systems.
* May increase the complexity of the system as scope of the system may expand beyond original plans.
* Changes become more costly and difficult.

**Agile Methods**

A mix of iterative and incremental process models which concentrates on process adaptability and customer satisfaction by fast delivery of working software product.



Agile uses 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 almost at all times, through the release iterations, minimizing the dangers of any major failures in future.

**Advantages of Agile**

* A very realistic approach to software development
* Promotes teamwork and cross training
* Suitable for fixed or changing requirements
* Resource requirements are minimum

**Disadvantages of Agile**

* Not suitable for handling complex dependencies
* An overall plan, an agile leader and agile PM practice is strictly required for it to work
* Involves customer interaction, therefore if customer is unclear, the team can be driven in the wrong direction
* Transferring of technology to new team members may be quite challenging due to lack of documentation

**References**

<http://www.tutorialspoint.com/sdlc/sdlc_waterfall_model.htm>

<http://searchcio.techtarget.com/definition/Prototyping-Model>

<http://www.teach-ict.com/as_a2_ict_new/ocr/A2_G063/331_systems_cycle/prototyping_RAD/miniweb/pg4.htm>

<http://www.teach-ict.com/as_a2_ict_new/ocr/A2_G063/331_systems_cycle/prototyping_RAD/miniweb/pg3.htm>

<http://istqbexamcertification.com/what-is-prototype-model-advantages-disadvantages-and-when-to-use-it/>

<http://www.lifecyclestep.com/open/440.0TESTPHASE.htm>

<http://www.tutorialspoint.com/sdlc/sdlc_agile_model.htm>