**Software Development Life Cycle (SDLC)**

Software Development Life Cycle is the process followed for a software project which has detailed planning that describes what goes on from the start of a project to the deployment of software system. The process from the requirements engineering, analysis & design, implementation, testing and finally, deployment.

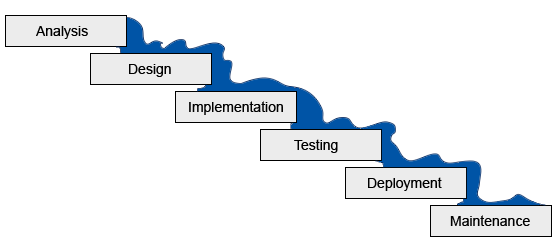
Firstly, we have to find out what the user wants the software to be able to do. Next, we need to analyze using a combination of text and diagrammatic forms to depict the requirements needed to be accomplished. Next, we need to do a visual prototype of the entity that will later be built, these includes the architectural design, user interface design and database design.

Afterwards, the implementation stage begins, in this stage of SDLC is where the actual development starts and the product is built. If the design was performed in a detailed and organized manner, code generation should not pose a problem for the developer.

Next, we have to test to ensure that the software is reliable and meets the requirements of the user’s needs. This is the stage where product defects are reported, and require actions to rectify the issue. It will then be retested until the product reaches the quality standards.

Finally, after the product is tested. The application may be distributed among a group of selected customers. Sometimes product deployment happens in stages as per the organization’s business strategy. The product might be released in a limited segment and tested in the real business environment. Based on the feedback of these users, the product may then be released as it is or with further enhancements in the targeted market segments.

**Waterfall Method**



The waterfall model is a sequential design approach, in which progress is seen as flowing steadily downwards through the phases of conception, initiation, analysis, design, construction, testing, production and maintenance. Time spent in the early stages of production can greatly reduce costs at later stages. For instance, bugs found in the early stage can be much easily rectified as opposed to a fully completed product, where a change is very complicated.

**Requirement Gathering and Analysis:** All possible requirements of the system to be developed and implemented are captured in this phase and documented in a requirement specification doc.

**System Design:** The requirement specifications from first phase are studied in this phase and system design will be prepared. System Design helps us in specifying hardware and system requirements and also helps in defining overall system architecture.

**Implementation:** With inputs from system design, the system is first developed in small programs called units, which are integrated in the next phase. Each unit is developed and tested for its functionality which is referred to as Unit Testing.

**Integration and Testing:** All the units developed in the implementation phase are integrated into a system after testing of each unit. Post integration the entire system is tested for any faults and failures.

**Deployment of system:** Once the functional and non-functional testing is done, the product is deployed in the customer environment or released into the market.

**Maintenance:** There are some issues which come up in the client environment. To fix those issues patches are released. Also to enhance the product some better versions are released. Maintenance is done to deliver these changes in the customer environment.

**Pros of waterfall method:**

* It divides complex tasks into smaller and hence more manageable tasks (maintain control)
* Each task produces a well-defined deliverable (documentation driven)
* Easy to control and monitor because you deal with one activity at a time
* Development moves from concept, through design, implementation, testing, installation, troubleshooting, and finally be in operation alongside with maintenance whenever required.

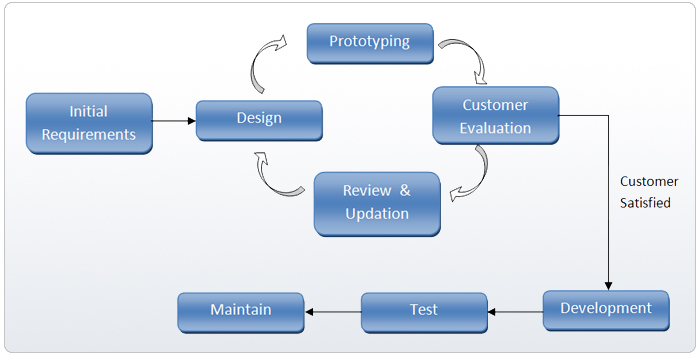
**Cons of waterfall method:**

* Can only see the product at the end, no opportunity to validate user requirements at early stages of development.
* If a problem is discovered at an earlier stage, nothing can be done about it (traditional waterfall)
* Does not allow for much reflection or revision
* Does not stress the need for anticipating changes

**When to use waterfall method:**

* The model is used only when the requirements are very well known, clear and fixed
* Problem definition is stable
* No ambiguous requirements

**Prototyping Method**



Instead of working on a functional system, start off with a prototype. The goal of a prototype is to see if it looks desirable and whether it meets the needs of the user. It can be used to design user interfaces, demonstrate the feasibility of the software, and verify that the technology will work on it. It serves to provide specifications for a real, working system rather than a theoretical one. It allows designers to understand the limitations of prototypes which simulates the characteristics of their intended design.

**Steps taken in prototyping:**

1. The system requirements are defined in detail as much as possible. This could include interviewing the clients and users representing the various departments of the existing system to determine their issues
2. Create a preliminary design for the new system
3. Construct a prototype of the new system, typically a scaled down version of the actual product. It represents an approximation of the characteristics of the final product.
4. Let the client evaluate the first prototype and allow for feedback to understand which areas are strong, which areas are weak.
5. Developer collects the feedback and analyze the remarks
6. Develop a second prototype based on the feedback of the client’s needs and have it evaluated again and repeated until the prototype has reached a desired outcome.
7. The final system will be constructed based on the final prototype
8. The final system is thoroughly evaluated and tested then fully deployed.
9. Maintenance may be carried out to prevent potential failures so as to minimize system downtime.

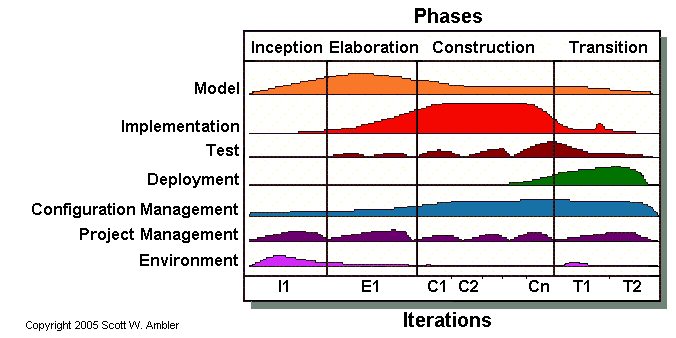
**Pros of prototyping method:**

* Clarify user requirements
* Specifications can be developed incrementally, giving users opportunity to change their mind

**Cons of prototyping method:**

* Estimating, planning and managing a prototype project can be difficult because there is no regular deliverables e.g. how to predict how many iterations?
* Continual changes tend to corrupt software structure. Changes become more costly and difficult.

**Unified Process Method**



Unified Process is an iterative and incremental software development process that provides an infrastructure for executing projects. It is a software development process framework, a lifecycle model involving context, collaborations, and interactions. The Unified Process is broadly applicable to different types of software systems, including small-scale and large-scale projects having various degrees of managerial and technical complexity, across different application domains and organizational cultures.

The aim of the analysis workflow is to analyze and refine the requirements to achieve detailed understanding of the requirements that are essential for developing the software product.

**Phases of UP:**

**Inception –** The aim of this phase determine whether the proposed software product is viable in the said business. Gain an understanding of the domain. Ask questions like “Is the proposed software product cost effective?”, “What are the risks involved in developing the software product, and how these risks can be mitigated?”

**Elaboration –** The aim of the elaboration phase is to refine the initial requirements, monitor the risks and refine their priorities, produce the management plan

**Construction –** The aim of this phase is to produce the first operation-quality version of the software product. It emphasizes on the implementation and product testing of the overall system.

**Transition –** The aim of this phase is to ensure the client’s requirements are met. Ensure faults are corrected, ensure that any loopholes are discovered before delivering the full product.

**Key characteristics of UP:**

* It is an iterative and incremental development framework
* It is architecture-centric with major work being done to define and validate an architectural design for most coding is done
* It is risk-focused and emphasizes that highest-risk factors be addressed in the earliest deliverables possible
* It is use-case and UML model driven with nearly all requirements being documented in one of those forms