SDLC Assessment 1

**Introduction:**

In this report I will look be investigating various concepts of software development life cycles as well as discussing what is required to successfully employ a Software Development Life Cycle along with a look at documentation that is produced along the way.

To begin we should have a look at what a Software Development Life Cycle (referred from here on as SDLC) is and what its purposes are. As the name suggests a SDLC is the process of how software is created, from concept brief to full release.

There are many different SDLC models, each of these have their own idiosyncrasies however, they all share one important thing, their purpose, which is to deliver a high-quality product which meets all the client’s needs.

**Different Approaches:**

Here we will look at some of the various approaches to Software Development and give some examples of models that exemplify these approaches.

**Incremental:**

An incremental approach is one where software is designed, executed and tested in increments, i.e., a little is added each time, until the project is completed. It takes parts from the waterfall model with elements of prototyping usually found in more iterative design models. It’s usually made up of 4 processes: Analysis, Design, Code and Test, that make up a build, the process repeats with more functionality added each time until the project is complete.

Its main advantages are that the customer can give feedback during development, it’s easier to test and bug fix as elements are introduced in relatively small changes for each iteration and the cost and time to deliver on the project is relatively low. A major disadvantage is that due to the added requirements that may be requested by the customer, feature creep is a distinct possibility (mitigated by having a rigorous planning phase and clearly defined goals for the project agreed upon by both parties) and the addition of unplanned for features may break earlier completed elements in the code requiring redesign of already completed work.

An incremental model should be used if there is a need to get the product to market quickly, if there is a new technology being involved and when the requirements of the project is clearly defined and understood.

Figure 1 The Incremental Model



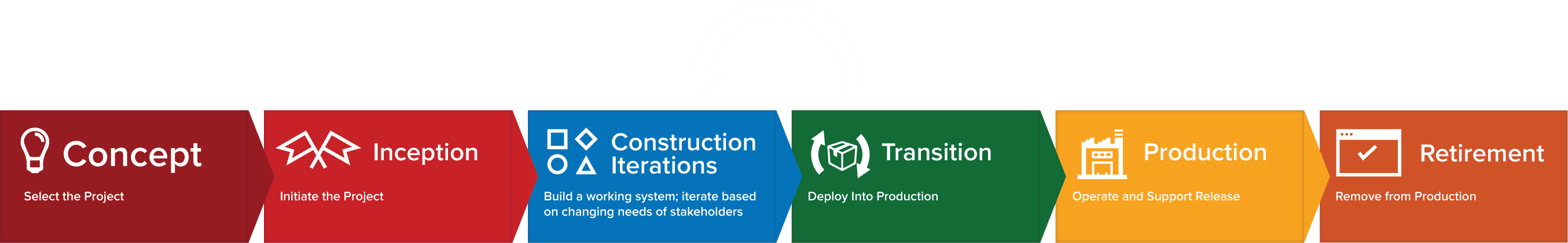
**Agile:**

An agile approach focuses on iterative development that focuses on delivering results over strictly adhering to established processes. At its core, the model believes that every project is unique and therefore has different requirements that need to be met necessitating the need to tailor methods to achieve these needs. It usually is broken up into 6 stages: Concept, Inception, Iteration/Construction, Release, Production and Retirement.

Advantages of using the model are functionality is developed quickly and can be shown to client, resource requirements are minimal, it is flexible meaning if the requirements of the project change it is easy for the project to adapt, it’s easy to manage giving flexibility to the developers and can promote teamworking and cross training which will benefit and strengthen the team moving forward. Some disadvantages are that it relies so heavily on customer interaction, if the customer is not clear in their needs the project can go in the wrong direction, due to the lack of documentation it is highly individual dependant meaning if there is turnover in the team it can be hard to get a new member of staff up to pick up where the previous member left off.

This approach is best used when a projects vision is not well defined as it allows the customer to adjust requirements as the project progresses.

Figure 2 The Agile Model



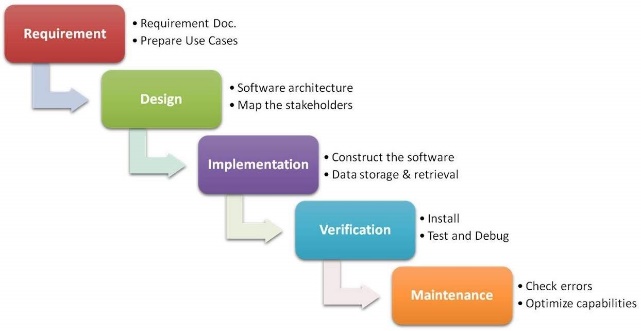
**Linear:**

A linear approach focuses on a very rigid structure in which a task cannot be worked on until the previous task is completed 100%. It was adopted early on in Software Development but is not used often currently. The Waterfall model exemplifies this approach.

Some advantages of this model are its easy to manage, fast delivery on product, there is minimal client interaction, meaning that the process is entirely dependant on the team making any unforeseen additions or feature creep impossible. The disadvantages include the lack of flexibility, the late implementation of testing and that small errors in the completed software can cause a lot of problems and be hard to fix.

The linear approach is most suited to a small-scale project that has clearly defined requirements and will not need features or extra functionality added later. It is totally unsuitable for large-scale projects that will need to be updated often.

Figure 3 The Waterfall Model



**Method Comparison:**

Let us compare two of these approaches

|  |  |
| --- | --- |
| **Waterfall** | **Agile** |
| Linear and sequential | Continuous iterative development and testing |
| Very structured | Flexible |
| Testing occurs at build phase | Testing throughout development |
| Good for small projects | Good for large projects |
| Little interaction with client | Client focused |

**Documentation:**

Documentation is produced throughout the SDLC, let us look at what documentation is produced at which stage of the cycle.

**Analysis:**

At the analysis phase documents like the Requirement Specification, Minutes of Meetings held and maybe even a Use Case are produced. The Requirement Specification collects all the requirements that are set out in the design and verification of the product. A Use Case helps clarify and organise these System Requirements. The Minutes of Meetings documents what has been discussed and action during meetings.

**Design:**

The most important document created here is the Design Document, this specifies the design pattern, code and guidelines (including user interface guidelines) that the project will adhere to. Design Reports may be made here which deal with reasons that design decisions have been made, potential alternative design solutions and perhaps even testing.

**Documentation Phase:**

During this phase documents like a System Document (describing the system and its parts), User Manuals, FAQs, Tutorials, Troubleshooting Manuals and Installation Guides. Mos of these are produced for the end user.

**When to code?**

So, with all these different phases when is the right time to implement programmers and start coding? Well, that will happen during the implementation phase, after all the major design decisions are made, we can start building (implementing) our code.