# Approach and Methodology

The chosen software methodology for this project is Agile System Development Life Cycle (SDLC). The SDLC model is a combination of the iterative and incremental models which allow for the rapid delivery of software products. (1) The project is broken up into small incremental builds which are provided in iterations. In each iteration planning, analysis, design and testing are all performed. At the end of an iteration the chosen piece of the project is delivered in a fully working and tested order.

The SDLC model comprises of six phases: Concept, Inception, Construction, Transition/Release, Production and retirement.(2)

**Concept - Pre Planning**

The concept phase is performed at the very beginning of the project. In this phase information is gathered on the area of interest to help identify and clarify the scope of the project and the goals of the project are loosely defined. Also in this phase, a small amount of feasibility analysis is performed to determine if the project is worth investing time and money into.

**Inception - Project Initiation**

The goal of this phase is to initiate the project. This is achieved by:

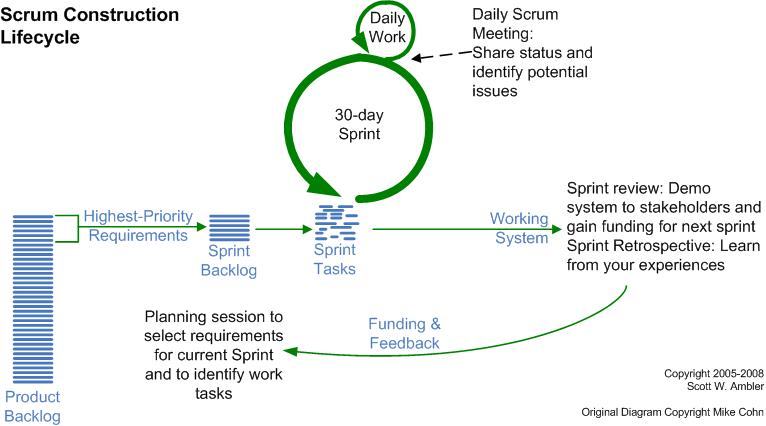
* Defining the scope of the system in terms of high level use cases
* Deciding upon an initial architecture for the system
* Choose a language and platform for the system
* Setting up the development environment (choosing an OS, IDE)
* Define initial requirements

Based on the initial requirements, project scope and initial architecture it should now be possible to give a rough estimate of time required for the project. This estimate will evolve as the project continues.

**Construction**

The construction phase of the project is when all of the development is performed. The project is broken down into different sections which are separated by individual requirements. These requirements are then organized by priority and development work is carried in order of the highest to lowest priorities. Each set of requirements is completed in a chosen timeframe called a Sprint. A Sprint can be any amount of time but usually ranges between one week to one month. In a Sprint all the designing, developing and testing is performed for the chosen requirements. The developers meet up for Daily Scrum Meetings where they share their progress and identify potential issues. By having daily meetings, the project leader can see whether the project is on track and identify, right away, any problems which may derail the project or extend estimates.

Once the Sprint is completed the working piece of software is delivered to the customer. Development can now begin on the next set of requirements while the recently finished requirements can be tested by the customer in parallel. Any changes which the customer requires can now be added to the list of requirements with a priority level. This allows for the customer to ask for drastic changes early on in the life cycle should they identify a major flaw or problem with the delivered product.



**Transition - Release**

In this phase of the project the final deployment of the product is carried out. Before the product is released one final round of testing is carried out. At this point most flaws should already be identified and fixed, so this round of testing is to make sure the product acts as the end users perceive it should. Hence a beta may be released or the product may be tested with a subset of the end users. Once the system is tested with these users it may be deployed.

**Production**

The goal of this phase is to keep the product useful, functional and productive after it has been deployed. Once the system is kept running and users use and know how to use the system the goal of this phase has been completed. This phase continues until the decision has been made to retire the product.

**Retirement**

The final phase of the Agile SDLC methodology is the retirement phase. This phase may be reached should the system become redundant or obsolete, get replaced, be no longer needed or should a newer version be released.

# Design

This chapter will cover the initial design elements of the system and it's architecture. These designs may be subject to change as the projects grows and evolves.

## Architecture Design

This system uses a Three-Tier Architecture comprising of a Presentation Tier, Middle Tier and Data Tier. Three-tier is a client-server architecture pattern in which the user interface, business/functional logic and data storage are developed as individual modules on separate platforms. (3) This modular application makes it possible to replace, update or upgrade any tier independently of the other tiers.

**Presentation Tier - The Client**

This is the User Interface for the system and in a web application it is usually the client's web browser. This is the tier from which the user interacts with the system. The user may view content, log in, request files or data and fill in forms. All of these request are sent to and handled by the Middle Tier.

**Middle Tier - Application Server**

This tier handles all of the logic and controls the functionality of the system. Any request from the user are processed here with validation and authentication being performed. Requests for data are taken from the user are received here, sent to the data tier and the response generated and returned to the user. This tier is sometimes referred to as middleware.

**Data Tier - Database Server**

This tier is concerned with the maintaining and serving of all data required by the system. The data may be stored in a shared folder but is more commonly stored in a database, especially for web applications. This tier usually provides a means of exposing the data to the Application server as well as a way to manage it which may be done by making an API available.



more to follow...