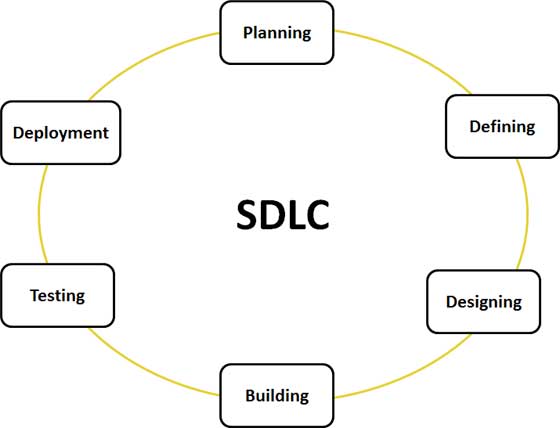
**Software Development Life Cycle (SDLC)**

Chan Zhi Mai – 1301354E

**What is Software Development Life Cycle (SDLC)?**

The software development life cycle (SDLC) is a framework or a process used by software industry to define tasks performed at each step in the software development process. It is also called as Software development process. SDLC is a structure followed by a development team within the software organization. It consists of a detailed plan describing how to design, 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.



**Figure 1: Graphical representation of the various stages of a typical (SDLC)**

**Stage 1: Planning and Requirement Analysis**

**Stage 2: Defining Requirements**

**Stage 3: Designing the product architecture**

**Stage 4: Developing the Product**

**Stage 5: Testing the Product**

**Stage 6: Deployment in the Market and Maintenance**.

**Software Development Models**

1. Waterfall model



Figure 2: Waterfall model

Waterfall model is also referred as a linear-sequential life cycle model. In a waterfall model, each phrase must be fully completed before the next phrase can begin. Waterfall model is usually used for small projects and there are no uncertain requirements. At the end of each phrase, a review takes place to determine if the project is on the right path and whether or not to continue or discard the project.

Advantages:

* This model is simple and easy to understand and use.
* It is easy to manage due to the rigidity of the model – each phase has specific deliverables and a review process.
* In this model phases are processed and completed one at a time. Phases do not overlap.
* Waterfall model works well for smaller projects where requirements are very well understood.

Disadvantages:

* Once an application is in the testing stage, it is very difficult to go back and change something that was not well-thought out in the concept stage.
* 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.
* Not suitable for the projects where requirements are at a moderate to high risk of changing.

When to use waterfall model:

* This model is used only when the requirements are very well known, clear and fixed.
* Product definition is stable.
* Technology is understood.
* There are no ambiguous requirements
* Ample resources with required expertise are available freely
* The project is short.

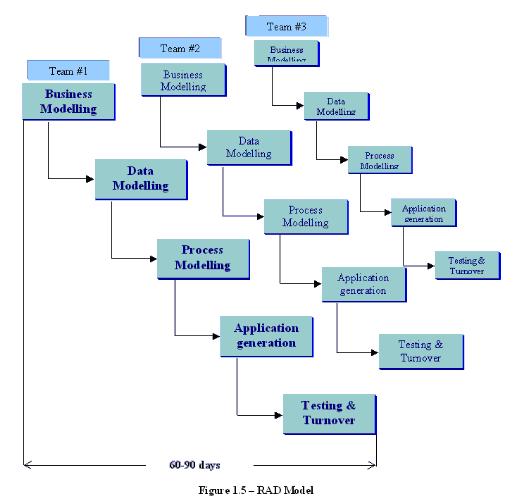
1. Rapid Application Development (RAD) model

Figure 3: Rapid Application Development (RAD) Model

Rapid Application Development Model is a type of incremental model. The components and functions are developed in parallel as if they were mini projects. The developments are time-boxed, delivered and then assembled into a working prototype. This can quickly give the customer something to see and use so as to provide feedback regarding the deliver and their requirements.

The phases in the rapid application development (RAD) model are:

**Business modeling**: The information flow is identified between various business functions.

**Data modeling**: Information gathered from business modeling is used to define data objects that are needed for the business.

**Process modeling**: Data objects defined in data modeling are converted to achieve the business information flow to achieve some specific business objective. Description are identified and created for CRUD of data objects.

**Application generation**: Automated tools are used to convert process models into code and the actual system.

**Testing and turnover**: Test new components and all the interfaces.

**Advantages of the RAD model:**

* Reduced development time.
* Increases reusability of components
* Quick initial reviews occur
* Encourages customer feedback
* Integration from very beginning solves a lot of integration issues.

**Disadvantages of RAD model:**

* Depends on strong team and individual performances 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
* Inapplicable to cheaper projects as cost of modeling and automated code generation is very high.

**When to use RAD model:**

* RAD should be used when there is a need to create a system that can be modularized in 2-3 months of time.
* It should be used if there’s high availability of designers for modeling and the budget is high enough to afford their cost along with the cost of automated code generating tools.
* RAD SDLC model should be chosen only if resources with high business knowledge are available and there is a need to produce the system in a short span of time (2-3 months).

1. Verification and Validation (V) model

Figure 4: Verification and Validation (V) Model

Verification and Validation model is a sequential path of execution of process. Each phase must be completed before the next phase begins. Testing of the product is planned in parallel with a corresponding phase of development.

The various phases of the V-model are as follows:

Requirements like BRS and SRS begin the life cycle model just like the waterfall model. But, in this model before development is started, a system test plan is created. The test plan focuses on meeting the functionality specified in the requirements gathering.

The high-level design (HLD) phase focuses on system architecture and design. It provide overview of solution, platform, system, product and service/process. An integration test plan is created in this phase as well in order to test the pieces of the software systems ability to work together.

The low-level design (LLD) phase is where the actual software components are designed. It defines the actual logic for each and every component of the system. Class diagram with all the methods and relation between classes comes under LLD. Component tests are created in this phase as well.

The implementation phase is, again, where all coding takes place. Once coding is complete, the path of execution continues up the right side of the V where the test plans developed earlier are now put to use.

Coding: This is at the bottom of the V-Shape model. Module design is converted into code by developers.

**Advantages of V-model:**

* Simple and easy to use.
* Testing activities like planning, test designing happens well before coding. This saves a lot of time. Hence higher chance of success over the waterfall model.
* Proactive defect tracking – that is defects are found at early stage.
* Avoids the downward flow of the defects.
* Works well for small projects where requirements are easily understood.

**Disadvantages of V-model:**

* Very rigid and least flexible.
* Software is developed during the implementation phase, so no early prototypes of the software are produced.
* If any changes happen in midway, then the test documents along with requirement documents has to be updated.

**When to use the V-model:**

* The V-shaped model should be used for small to medium sized projects where requirements are clearly defined and fixed.
* The V-Shaped model should be chosen when ample technical resources are available with needed technical expertise.
* High confidence of customer is required for choosing the V-Shaped model approach. Since, no prototypes are produced, there is a very high risk involved in meeting customer expectations.

Appendix

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

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

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

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