SOFTWARE SYSTEMS DEVELOPMENT LIFE CYCLE
A REFERENCE GUIDE

BUSNIESS INFORMATION ENGINEERING CORPORATION

Author: Sarah Taghavi
Document: Software Systems Development
Category :Software Engineering Training Material
June 2008

SOFTWARE DEVELOPMENT LIFE CYCLE MODELS

The process of developing a new software product or maintenance updates of existing product consists of activities that need to be defined and managed according to a plan or model. The life cycle model is a prescriptive model of what and when the steps of development are to take place. It specifies and establishes the various tasks and the order and inter-relationships of software development for the development team to follow. A well managed project would benefit greatly from a master plan by streamlining your project and improving the quality of the final product.
It is important to know the characteristics of all the FLC models and choose the one that best meets your development goals. Choosing the wrong model or not choosing one at all , will likely lead to slow progress, lack of control, repeated work risk exposure , frustration and poor work quality. On larger software engineering projects with many sub-projects or modules, not using the FLC framework could also lead to individually on target , but collectively misdirected outcome.

Table a-1 lists the available FLC models :

FULL LIFE CYCLE MODELS
Pure Waterfall **Code-and-Fix** **Spiral** **Modified Waterfalls**
Salmon LC **Sashimi** **Waterfall with Sub Projects** **Waterfall with Risk Reduction**
Evolutionary Prototyping **Staged Delivery** **Design-to-Schedule** **Evolutionary Delivery** **Commercial Off-the-Shelf Software**

Pure Waterfall Model

The pure waterfall model is the oldest lifecycle development model , serving the basis for other and more practical models of software development.
The stages of the waterfall model are clearly defined and are followed in sequential order. Each stage of development produces documentation to be reviewed at the end of the stage and approved prior to moving to the next stage of development. The stages do not overlap and it would be very difficult and costly to back up and revise or correct mistakes.
The phases of the pure waterfall model are as follows:

SOFTWARE CONCEPT

REQUIREMENTS ANALYSIS

ARCHITECTURAL DESIGN

DETAILED DESIGN

CODING AND DEBUGGING

SOFTWARE TESTING

Advantages :

- Planning is done up-front , minimizing changes later in code development
- Control over the phases of the project
- Easy to track progress
- Ideal for well understood but complex or later releases of software systems
- Works well when quality dominates cost/schedule
- Provides structure which is necessary when the technical staff is un-experienced
- Error are detected early in the project

Disadvantages :

- Documentation effort is extensive
- It is difficult to fully specify the requirements at the very beginning of development
- No code generation or components developed for customer/user until later in the project . Not suited for Rapid Development of most business driven appellations.
- Not flexible , very difficult and costly to backup to previous phases of development.
- Activities that span the phases are difficult to accommodate

Code_and_Fix Model

This model, as it's name implies is that you immediately begin to code with not well defined or often well understood requirements. It is not a useful model for any project except for very small and manageable projects. The general idea is to start with a mixture of coding , informal design, debug and testing with little time for documentation or formal design. Nonetheless , because of the production demands, this model is commonly used.

Advantages:

- No overhead, no time spent on planning, QA or documentation.
- Lack of a model , so everyone (specially developers) could work with little expertise or spend time on the understanding or following development models
- Useful for proof-of concept small projects or smaller demos

Disadvantages:

- Unless the project is simple and small , this is a dangerous model in assessing progress , quality and risks.
- You could be well into coding, having spent a large portion of the budget before a fundamental design flaw is identified and need to start over.
- Not suited for software development even RAD.

Spiral Model

This model is oriented specifically to reduce risks by dividing up the project into small manageable cycles. Each cycle is well defined and addresses one or more risks. Risks could be in flaws of design, misunderstood requirements, performance problems or the development tools utilized. The Spiral model is a sophisticated and complex model , in which each phase of development corresponds to an iteration of the spiral.
The goal is to start small and as you progress assess risks and commit to the next iteration of development.
Each iteration primarily involves the steps outlined below:
1- determine objectives, alternatives and constraints
2- resolve risks , if any.
3- evaluate alternative
4- develop and verify the deliverables in the current iteration
5- Plan the development of the next iteration

You can decide to choose a different model for the development efforts of each iteration of the cycle , as long as risk evaluation is performed prior to start.

Advantages:

- As costs increase, risks decrease (the more time you spend on evaluating risks , the less risks you are taking
- Control over the project
- Suited for RAD

Disadvantages:

- Complicated and requires knowledgeable management to maintain and enforce
- Not suited for straight forward , modest risk projects