E-Gram Seva

SDLC Model v1.0

Team 22
February 17, 2013

REVISION HISTORY

Version	Author	Date
Version 1	Rutvik Jhala, Siddharth Vadnagara	February 4, 2013
Review Version 1	Aayushi Sharma, Surbhi Singhal	February 5, 2013

CONTENTS

1. Introduction

These are the following Software Development Models we have analyzed:

- Classical Waterfall Model
- Iterative Waterfall Model
- Prototyping Model
- Rapid Application Development (RAD) model
- Evolutionary Models
  - Incremental Model
  - Spiral Model
  - Concurrent Development Model
- V-shaped Model

We have chosen the **Iterative Waterfall Model** for our project.

2. Classical Waterfall Model (CWM)

Classical Waterfall Model performs well for products with clearly understood requirements or when working with well understood technical tools, architectures and infrastructures. Its weaknesses frequently make it inadvisable when rapid development is needed. In those cases, other software development life cycles models should be used.

Advantages

- Minimizes planning overhead since it can be done up front.
- Structure minimizes wasted effort.

Disadvantages

- Inflexible as it discourages the reviewing of the phases that are completed.
- A major blunder undetected in the earlier stages could be disastrous later on.
- The client should be patient as a working version of the product is available only in the later stages.

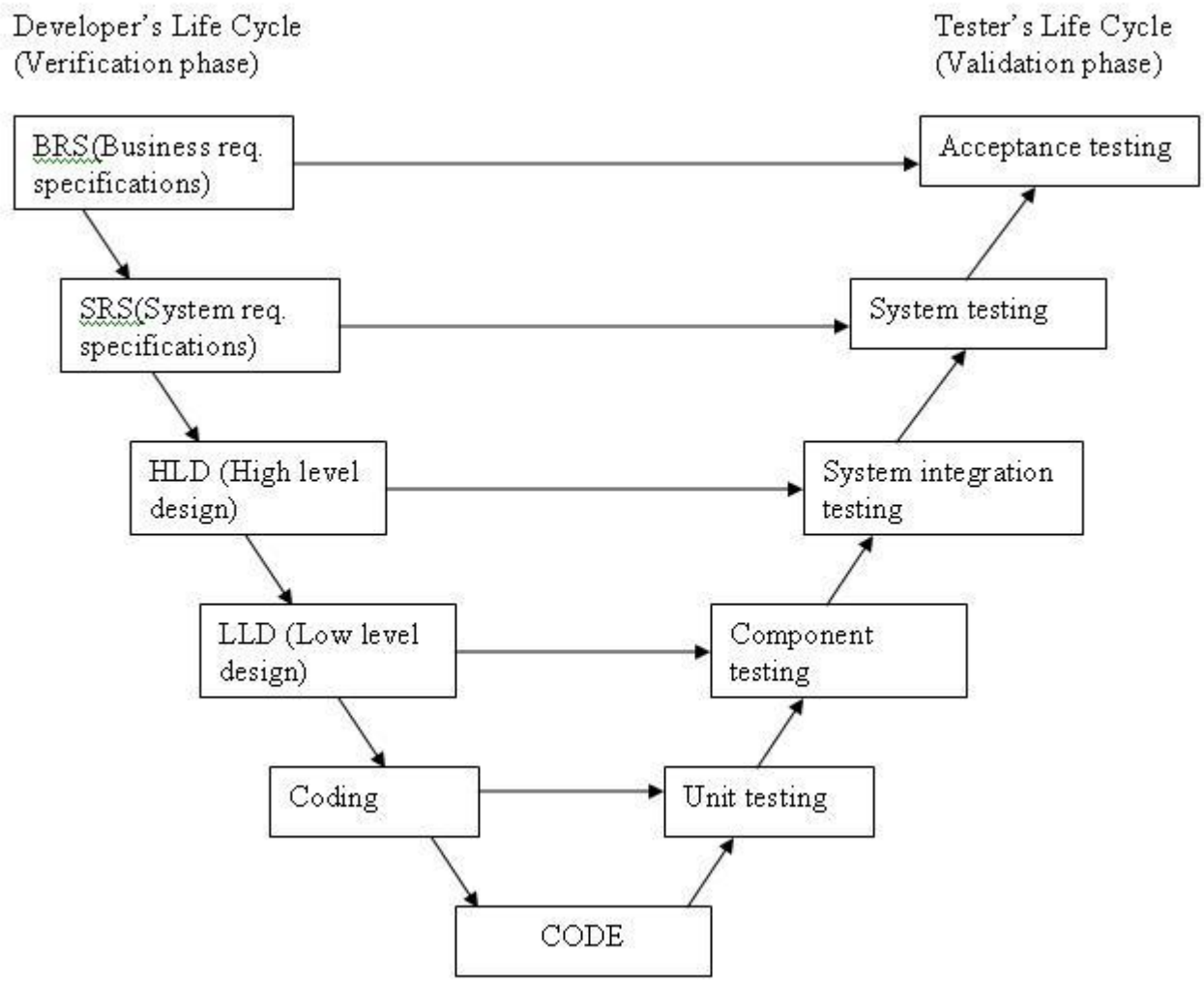
Justification for rejection

We have rejected this model because it requires well understood requirements. It works well when the problem statement is clear and we know the solution beforehand. Since the group as a whole lacks experience and CWM provides no scope for correcting mistakes therefore, undetected mistakes may be problematic at a later stage. We have to go through rigorous reviewing.

3. V model:

V- model means Verification and Validation model. Just like the waterfall model, the V-Shaped life cycle is a sequential path of execution of processes. Each phase must be completed before the next phase begins. Testing of the product is planned in parallel with a corresponding phase of development.

Diagram of V-model



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

- 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

- 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.

Justification for rejection

V-shape model is not pertinent to our software project as it necessitate ample technical resources with equally strong technical expertise. Our project team is inexperienced and naive in software developing. This project requires a very flexible method, so that the unintentional errors can be amended in midway also.

4. Rapid Application Development (RAD) model

It is an incremental software development model that aims at fast development of software by having planning interleaved in coding phases mostly.

Advantages:

- It is cost effective due to reuse of existing program components.

- Development of software within a very short span of time.

Justification for ruling out

- This model requires larger teams with technical expertise which cannot be delivered by a small group of beginners.
- Our projects demands a lot of field work and planning for which this model is not suitable.
- RAD requires that the requirements are well understood but we have made few assumptions other than the requirements of the client.

5. Prototyping Model

Prototyping Model is a lifecycle model which involves building of a prototype of the final product. It is a lower quality version of the actual end product, involving lower implementation of the functionalities. The prototype serves the purpose of eliciting the user's and client's correct response to the design proposed by the developer since only a description might not suffice. Rework on prototype is done until the client is satisfied that all the requirements and functionalities are met.

Advantages

- A prototype helps the client to better visualize the end product. This helps the developer to obtain the client's requirements well before the actual designing, implementation and testing of the software.
- The clarity of requirements and functionalities obtained due to the prototype helps save cost, time and effort. This is because changes made after the product development result in a greater overhead in terms of effort and money.
- It also helps to make the technical requirements very clear in the developer's head so that any prospect of risks is reduced.

Justification for ruling out

- There is a consensus between the client and the developing team regarding the requirements of the product. On evaluating the tradeoff between the clarity obtained after making a prototype and the time and effort spent on building the prototype, saving on time to work on the final product is agreed upon.
- The technical aspects of the software are not very precisely identified while building the prototype and might result in misunderstanding with respect to the final product among the developers.
- The users might believe that the prototype is the final product or requires only a little rework which might result in demanding of the product sooner than agreed upon.

6. Evolutionary Model:

Evolutionary module is one in which complete application/software is broken down into modules and then these modules are delivered to the client in a successive manner.
It is of three types:

6.1 Incremental Model

In an Incremental model after delivering the 'core product', other functionalities are added as increment to the software according to the feedback received by the client. This process is repeated until the complete product is produced.

Advantages

- All the functionalities will be as per the client's requirements.
- Testing at each phase make the core module more efficient.
- It is useful when staffing is unavailable for the complete implementation.

Justification for ruling out

In the project

- It is hard to be divided into functional units which can be implemented in an incremental manner.
- Also the core product is also not very well differentiated.

6.2 Spiral Model

In spiral model at every stage refinement of the product is made through identification, designing and evaluation of a certain feature. It incorporates the idea of iterative development along with controlled aspects of waterfall model.

Advantages

- Time and cost estimates are very close to real value as every stage process moves through all the four quadrants of identification, development and evaluation.
- Risk management at every stage proves efficient.

Justification for ruling out

This is not an effective way of approaching our project as in initial stages we are not aware about everything so could not suggest alternative, nor we can estimate possible risk forthcoming, so we opted this out.

6.3 Concurrent Development Model

Concurrent Development model consists of a series of events that will trigger transitions from state to state for various activities involved in the project.

Advantages

- People are involved in more than one activity at a time.
- Status of the project can be determined at any stage.
- Represented systematically as a series of major technical tasks, states and activities.

Justifications for ruling out

We ruled this out because under this there is little scope to add new features at the end but our project may require some features to be added at last as per the need of the client.

7. Iterative Waterfall Model

- Each iteration involves Design Analysis and Implementation as well as verification of the current build/version of the system.
- If the application lacks any requirements or has any problems then the phase will be looped back to previous iteration.
- It supports redesign, acceptance and review of any new requirement.
- It involves analysis of usability, achievement of goals, reliability, efficiency, and structure.

Advantages

- More flexible than the pure waterfall model.
- If there is personnel continuity between the phases, documentation can be substantially reduced.
- Implementation of easy areas does not need to wait for the hard ones.

Disadvantages

- Milestones are more ambiguous than the pure waterfall model.
- Activities performed in parallel are subject to miscommunication and mistaken assumptions.
- Unforeseen interdependencies can create problems.

Why did we choose this model?

This model fits into the requirement of our system very well since it is a modular approach with well-defined entry and exit criteria for every phase. One phase is entered only when entry criteria for that previous phase are met and deliverables of the previous phase are ready. Also, error is corrected as soon as it is detected by going back to the phase where it was introduced. This model is appropriate as problem statement is well understood.