Proc. of the Intl. Conf. on Advances in Computer Science and Electronics Engineering — CSEE 2013

ISBN: 978-981-07-5461-7

Comparison of Software Development Life Cycle Models

Raj Kumari and Heena

***Abstract*—Software has become the necessity part of modern society. There are several software development methodologies in use today. Software development models basically fall into two categories that are heavyweight and lightweight. Heavyweight methodologies are traditional methodologies such as Waterfall, Spiral, Iterative, RUP (Rational Unified Process) and lightweight are agile methodologies such as Feature Driven Development, Scrum, XP etc. This paper deals with unfolding Traditional methodologies- Waterfall model and Agile methodologies-Feature Driven Development with their advantages and disadvantages.**

***Index Terms*—SDLC, Phases of SDLC, Heavyweight methodologies, Lightweight methodologies, Waterfall model, Feature Driven Development.**

I. Introduction

SDLC is a systematic approach to solve problem and is composed of several phases, each comprising multiple steps. A software cycle deals with various parts of software and phases from planning to deploying software. All the phases are conceded according to the needs. Each way is known as a Software Development Lifecycle Model (SDLC). [1]
It act as a framework that describes the activities performed at each stage of a software development life cycle [2]
It is a process used to develop high quality software system which meets customer requirements within time and cost estimates. Software processes consist of set of activities that lead to the production of software product. Today, companies have lot of choices of models to develop their software. Each model satisfies specific need of customer. Different methodologies have been developed for the improvement of the quality of the software still large and complicated software projects are vulnerable to large problems. [3]

II. Phases of SDLC

Software is build to solve problem and make it easy to deal with. Software is developed in many phases and in order to develop large and complex software, these phases are splitted into various activities and further activities into multiple steps for simplicity.

Raj Kumari Punjab University
Chandigarh, India

Heena Punjab University
Chandigarh, India

Phases for developing software system are:

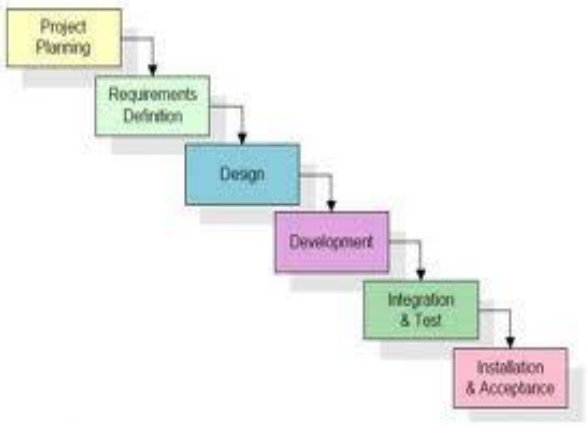


Figure 1. [3]

Strengths and Weaknesses of SDLC:

Strengths	Weakness
Control : High	Development time: High
Description: Detailed steps.	Development Cost: High
Documentation: Well defined.	Rigidity: Limited user input
Monitoring: Ability to monitor large projects.	Rework: If error occur in early stages of the project, rework is required.

TABLE1[4]

III. HEAVYWEIGHT METHODOLOGIES

Heavyweight methodologies are the traditional way of developing software and are also known as traditional methodologies. They are process oriented and follow predictive approach with well known requirements and clearly defined milestones. Heavyweight methods depend greatly on documentation with comprehensive upfront planning in order to succeed. They are pessimistic to changes; hence they are expensive in terms of refactoring cost. These methodologies deal with large project size and requires large team size.[5]

These methodologies include Waterfall model, Prototype model Spiral model, Incremental model, Rational Unified Process and various other models.

A. *Waterfall Model*

Proc. of the Intl. Conf. on Advances in Computer Science and Electronics Engineering — CSEE 2013

ISBN: 978-981-07-5461-7

Waterfall model is one of the earliest and most common life cycle models. It was first put forth by Winston Royce in 1970 in one of his articles. It is also known as sequential life cycle model because model develops systematically from one phase to other in sequence.

It consists of basically following phases:

I. Requirement
II. Design
III. Implementation
IV. Verification
V. Maintenance

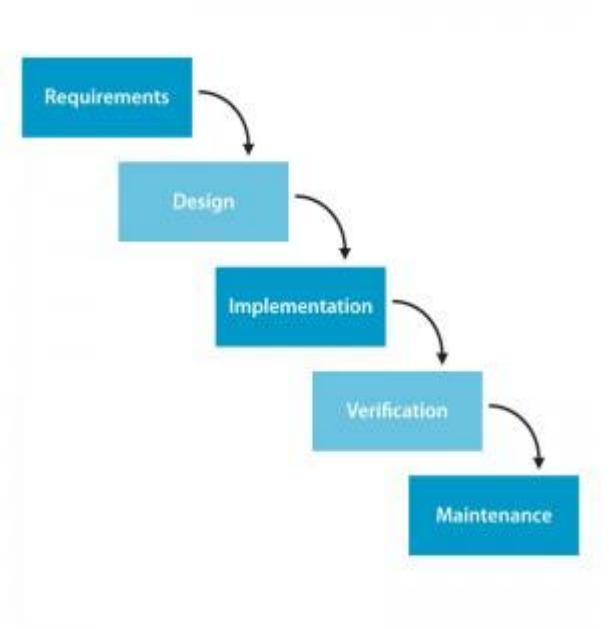


Figure 2 [6]

Brief description of phases:

I) **Requirements:** This is the first phase of the model where requirements are gathered on the basis of which software is developed. This phase indicate for what purpose software is developed.

II) **Design:** This phase starts on the completion of first phase. It includes basic design and technical design. Software plan is developed and functions of each part are decided.

III) **Implementation:** Source code is written in this phase.

IV) **Verification and Integration**: In this phase, whole design is checked against functionality. Errors are discovered in this phase and removed. Error free modules are integrated to form a system.

V) **Maintenance:** This phase ensures that software developed will work as desired. [6]

Advantages of waterfall model are:

1) **High documentation**: As documentation is produced at every stage, thus making understanding of product design easier.
2) **Simple implementation**: Due to its sequential nature its implementation is easy.
3) **Resources:** Minimum numbers of resources are required for its implementation.
4) **Milestones:** Each stage has well defined deliverables and milestones. So management is easy.[7]

Disadvantages of waterfall model are:

1) Low guarantee of success.
2) Low user involvement- users are involved only at the beginning.
3) No working version is developed until final stage gets completed.
4) Inflexible: Difficult to slot in changes if required.

Its comparison with other models:.

Models/Features	**Waterfall**	**Incremental**	**Prototype**	**Spiral**	**RUP**
Client involvement	Only at beginning	Intermediate	High	High	At beginning and at last stage
Requirement specification	Beginning	Beginning	Frequently Changed	Beginning	Beginning
Requirement understanding	Well understood	Well understood	Not well understood	Well understood	Difficult to understand
Project Size	Works well for smaller projects	Small projects	Large projects	Good for large and mission critical projects	Not suitable for small projects
Simplicity	Simple	Intermediate	Complex	Complex	Simple And Clear
Documentation	Vital	Yes	Weak	Yes	Yes
Time framework	Long	Very Long	Short	Depends upon project	Short time frame
Flexibility	Inflexible	Less Flexible	Highly Flexible	Flexible	Substantial
Guarantee Of accomplishment	Less	High	Good	High	Not Guaranteed
Reusability	Limited	Yes	Weak	Yes	Supports reusability of existing classes
Risk Involvement	High	Manageable	Low	Low	Critical risks in early stages
Maintenance	Least	Promotes maintainability	Regular maintenance	Usual	Easily maintained
Changes incorporated	Difficult	Easy	Easy	Easy	Easy

Table 2[3]

ISBN: 978-981-07-5461-7

IV. *Lightweight Methodologies*

Lightweight software development methods were introduced in the mid-1990s as a reaction in opposition to heavyweight methods. These are now typically referred to as agile methodologies, after the Agile Manifesto published in 2001. These methodologies are based upon iterative and incremental development methods. It emphasizes on customer satisfaction through continuous delivery of functional software It promotes adaptive planning, evolutionary development and delivery, follows iterative approach, and has few rules to follow as in contrast to heavyweight methodologies.

These methodologies include Feature Driven Development (FDD), Scrum, Extreme Programming (XP), Crystal Clear and various others.[8]

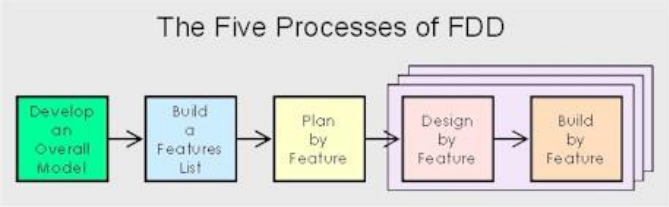
A. *Feature Driven Development*

Feature Driven Development is an agile software development methodology by Peter Coad and Jeff De Luca. Main idea is to deliver substantial, working software repetitively in a well-timed manner.

Phases of Feature Driven Development:

It consists of five phases:

1) Developing an overall model.
2) Build a feature list.
3) Plan by feature.
4) Design by feature.
5) Build by feature.



Brief description of phases:

1) **Developing overall model**: After having an understanding of needed business functionality, domain expertise, and overall scope of the project, project starts with walkthroughs. Detailed walkthrough of each domain is carried out, hence producing models overall design.
2) **Build a feature List**: List of features is identified on the basis of the information gathered in first phase. This Phase can also be well thought-out to be the functional decomposition of the Domain Model obtained from Phase 1.This can be done by decomposing into subject areas and further these subject areas consist of business activities which characterizes the features.
3) **Plan by Feature**: Development plan is produced on the completion of features list. Feature sets are ordered on basis of priority and assigned to chief programmers. Schedule and major milestones are set for features.
4) **Design by feature**: Class owners form feature teams that handle small group of features. Overall model is refined by working out on the sequence diagrams of features. Design inspection is also done in this phase.
5) **Build by Feature:** In this phase activity to produce complete feature is produced. Actual code of classes is developed by their class owners. Classes are migrated to the build after a successful code inspection.[8]

Advantages of Feature Driven Development are:

1) **Customer Satisfaction**: Here highest priority is to satisfy the customer through early and constant delivery of valuable software.
2) **Easy monitoring**: Parking lot charts and Feature Complete charts are used for progress tracking thus making it easy.
3) **Inbuilt tools**: Inbuilt tools are helpful in effective measuring and reporting of progress for management.
4) **Less Overhead**: Feature team is highly effective keeping less communication channels thus avoiding high overhead.[9]

Disadvantages of Feature Driven Development are:

1) FDD relies heavily on inspections to ensure high quality of designs and code.
2) It is not suitable for projects where requirements changes frequently.

Comparison with Extreme programming(XP):

Models/Features	FDD	XP
User Requirements	Critical part	Minimum effort required
Code Documentation	Yes	No
Tools	Management tools required	No special tools
Implemented	Features	Users stories
Use for projects	With stable requirements	Frequently changing requirements
Development team	Forms team hierarchy	No hierarchy
Team size for Iteration	Small volatile team	Whole team
Design stage	Formal	Not Formal
Refactoring	Discourages	Promotes
Progress tracking	precise	Not exact

Table 3[10]

ISBN: 978-981-07-5461-7

Conclusion

Various SDLC models are available today such as V model, RAD, Agile each with their own advantages and disadvantages. This paper explained waterfall model and its comparison with incremental model, prototype model, spiral model and RUP. In this paper Feature Driven Development model was studied and it was found that it was suitable for projects with suitable requirements opposite to XP.Different models are used by different organizations depending upon their needs.

References

[1] Raymond Lewallen - CodeBetter.Com - Stuff you need to Code Better! Published 08-01-2008.

[2]Tobin J Lehman, Akhilesh Sharma , "Software Development as a service: Agile Experiences", in annual SRII Global Conference (2011).

[3] Ms. Shikha maheshwari1 Prof.Dinesh Ch. Jain,.A Comparative Analysis of Different types of Models in Software Development Life Cycle,IJARCSSE, Volume 2, Issue 5, May 2012.

[4] http://www.waterfall-model.com/sdlc

[5] Sandrine Balbo and Ali Khan ,"A Tale Of Two Methodologies: Heavyweight Versus Agile"

[6] waterfall-model.com

[7] Ruchika, Shaweta ,"A comparative study of software development models."

[8] wikipedia.org

[9]www.step10.com/SoftwareProcess/FeatureDrivenDevelopment/FDDPractices.html

[10]Serguei Khramtchenko,"Comparing eXtreme Programming and Feature Driven Development in academic and regulated environments" Harvard University ,May17,2004

Author:

Heena is student at University of Institute of Engineering and Technology, Punjab University,Chandigarh,India