Heavyweight vs. Lightweight Methodologies: Key Strategies for Development

Prabha Singh[1], Shubham Srivastava[2], Chaynika Srivastava[3], Aditya Singh[4]

Computer Science & Engineering Department, ITM, GIDA, Gorakhpur, Uttar Pradesh, (India)

ABSTRACT

A software development methodology refers to the framework that is used to plan, manage and control the process of developing a Software Product. The main objective of this paper is to represent different models of software development and different aspects of each model to help the developers to select specific model at specific situation depending on customer demand. There are two software development methodologies used: Heavyweight and Lightweight. Heavyweight methodologies, also considered as the traditional way to develop software, claim their support to comprehensive planning, detailed documentation, and expansive design. The lightweight methodologies have gained significant attention from the software engineering community in the last few years. Unlike traditional methods, these methodologies employ short iterative cycles, and rely on tacit knowledge within a team as opposed to documentation. This survey represents the strengths and weakness between the two opposing methodologies and provided the challenges associated with implementing both processes in the Software Industry.

Keywords: Heavyweight development model, Lightweight development model, Software Development Life Cycle,

I. INTRODUCTION

Software Development Life Cycle (SDLC)[5] is a measure to check

- Whether the software that is built is as per the customers' requirements
- Work efficiently and effectively
- And are less expensive to build and cost – effective to upgrade.

SDLC provides a series of steps to be followed to design and develop a software product efficiently. SDLC framework includes the following steps:

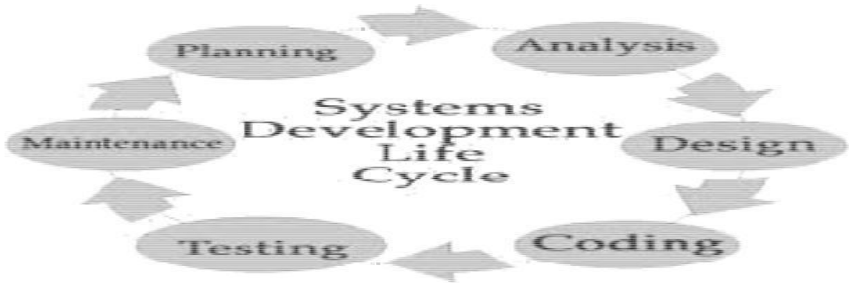

Fig.1. SDLC Framework

SDLC is divided into two categories; heavyweight and lightweight. Software industry has an option to choose suitable methodology/process model for its current needs to provide solutions to give problems. In software development, "lightweight" methodologies are gaining ground on more traditional "heavyweight" methodologies. Both have their advantages and disadvantages. The main difference is that the highly structured "heavyweight" methodology used by the shuttle designers is predictable, while the flexible "lightweight" methodology used to develop cutting-edge software solutions is not. These solutions use new technologies and designs Heavyweight Methodologies: There are various heavyweight development models or methodologies. They are as follows:

II.WATERFALL MODEL

Strengths:

- Easy to manage due to the rigidity of the model, because each phase has specific
- Reinforces good habits: define-before-design, design-before-code.

Weakness:

- Unrealistic to expect accurate requirements so early in project.
- Software is delivered late in project, delays discovery of serious errors.

1. Spiral Model[4]:

Strengths:

- High amount of risk analysis hence, avoidance of Risk is enhanced.
- Software is produced early in the software life cycle.
- The model makes use of techniques like reuse, prototyping.

Weakness:

- The model is not suitable for small projects as cost of risk analysis may exceed the actual cost of the project.
- Different persons involved in the project may find it complex to use.

III. ITERATIVE MODEL[3]

Strengths:

- In iterative model we are building and improving the product step by step. Hence we can track the defects at early stages. This avoids the downward flow of the defects.
- In iterative model we can only create a high-level design of the application and we can get the reliable user feedback.

Weakness:

- Each phase of iteration is rigid with no overlaps.

- Costly system architecture or design issues may arise requirements are gathered up front for the entire lifecycle

IV.INCREMENTAL MODEL[3]:

Strengths:

- Easier to test and debug during a smaller iteration.
- Easier to manage risk because risky pieces are identified and handled during its iteration.

Weakness:

- Total cost is higher than waterfall.
- Requires heavy documentation. Follows a defined set of processes, defines increments based on function and feature dependencies.

Lightweight Methodologies: There are various lightweight development models or methodologies. They are as follows:

1. Extreme Programming(XP):

Strengths:

- Produces good team cohesion.
- Emphasizes final product.

Weakness:

- Programming pairs is costly.
- Test case construction is a difficult and specialized skill.

2 .Feature Driven Development (FDD)[12]:

Strengths:

- FDD used for larger size projects and obtain repeatable success.
- This model has just enough detail to form a good shared understanding, vocabulary and conceptual framework for the project.

Weakness:

- No written documentation provided to clients in this methodology so, they are not able to get a proof for their own software.
- FDD depends on user requirements; changes in user requirement during project development can affect project progress.

3. Rational Unified Process (RUP):

Strengths:

- It is proactively able to resolve the project risks that are associated with the clients evolving requirements for careful changes and request management.
- Very less need for integration as the process of integration goes on throughout the development process.

Weakness:

- Integration throughout the process of software development causes more issues during the stages of testing.
- This process is too complex therefore it is very hard to understand.

HH. Comparisons between Heavyweight and Lightweight Strategies:

Table I. Comparision between Heavyweight and Lightweight Strategies[13]

Sr.	**Parameters**	**Heavyweight Strategies**	**Lightweight Strategies**
1	Budget	High Budget allocation is done	Low Budgets are required
2	Team size	Large team size	Small team size / Creative team
3	Project criticality	Extremely Critical	Low Criticality
4	Technology used	Process Oriented	People Oriented
5	Documentation	Explicit knowledge is required	Face to face communication is possible
6	Training	Heavy training is required as the software is delivered once it is totally ready	As the development team and the customer are interacting with each other less

			training is required
7	Best practices/lessons learned	More emphasis on process hence no communication	Face – to – face conversations between the client and the team
8	Tools and techniques	Tool and techniques like waterfall model is used	new like XP, FDD are used
9	Existing processes	Water fall Model	XP, FDD
10	Software	Predictive	Adaptive
11	Testing	Testing happens only after the completion of the development.	Testing team work in parallel with the development team which helps to find the defect as soon as possible.
12.	delivery	Usually deadline not met	Delivered at deadline

V.CONCLUSION

It will be inappropriate to say that size is the only criteria which can be helping us in choosing the right methodology. Different methodologies are appropriate in various situations. Many a time Heavyweight methodologies were considered appropriate as they were disciplined. However, lightweight methodologies have a different aspect altogether. They compromise between no process and many processes. These new methods managed the projects which are having short time box, and uncertain and are dynamic to change. The lightweight methodologies made the work of the developers easier in terms of cost and time. The developers because of these methodologies were able to visualize their end product clearly. Lightweight methodologies made the developers re – examine the heavyweight strategies in respect to requirement analysis and process improvement.

REFERENCES

[1] L. Jiang and A. Eberlein,” Towards A Framework for Understanding the Relationships between Classical Software Engineering and Agile Methodologies“, ACM, 2008.

[2] Basili, V. R. & Reiter, “A Controlled Experiment Quantitatively Comparing Software Development Approaches”. IEEE Transactions on Software Engineering, Vol. 7, 3 (3), pp. 299-320, 1981.

[3] Larman, C. & Basili, V. R. “Iterative and Incremental

[4] Development: A Brief History”. IEEE Software, Vol. 20, pp.47-56,2003.

[5]B. Boehm, “A Spiral Model of Software Development and Enhancement,” IEEE Computer, May 1998.

[6] Davis, A, Bersoff, E, Comer, E, “A Strategy for Comparing Alternative Software Development Life Cycle Models”, IEEE Transactions on Software Engineering, vol.14, iss.10, pp.1453-1461, 1988.

[7]M. Sami Abd EI-Satar” Software Development Life Cycle Models and Methodologies”, 2012.

[8] E. Mnkandla, "About Software Engineering Frameworks and Methodologies", IEEE AFRICON 2009.

[9] Steve Easterbrook, "Software Lifecycles", University of Toronto Department of Computer Science, 2001.

[10] Kaiser, G., P. Feiler, and S. Popovich, Intelligent Assistance for Software Development and Maintenance, IEEE Software , 5, 3, 1988.

[11] A Comparison between Five Models Of Software Engineering IJCSI International Journal of Computer Science Issues, Vol. 7, Issue 5, September 2010 ISSN (Online): 1694-0814.

[12] T Bhuvaneswari and Prabaharan S.(2013) “A Survey on Software development life cycle model”, Journal of Computer Science and Information Technology, Vol2 (5), 263-265.

[13] Nabil Mohammed Ali Munassar Ali and Govardhan A (2010) “A Comparison between Five Models of Software Engineering” International Journal of Computer Science, Vol. 7(5), 98-100.