SDLCs FOR NEW ERA

Mohd.Salman Khan[1], Shalabh Chaturvedi[2],Tanu Ruhela[3]

[1]*Asst. Professor, ABES IT- Ghaziabad, (India)*

[2,3]*B.Tech (IT) – Scholar, ABES IT- Ghaziabad, (India)*

ABSTRACT

The traditional methodologies today cannot be used up with the new e-business environment. They are often to "heavy" to keep up with the pace of e-business software development projects. To have overcome this problem, the so called "light" SDLC methodologies have recently been developed and put to use.

I INTRODUCTION

Almost since the very beginning, a rational, engineering-based approach, such as the Waterfall method, has been used to develop projects. This approach seems to assume that problems can be well defined, processes can be optimized, and results can be well predicted. Extensive up-front planning and research is done to measure and control the variations in the development life cycle. The shortcomings arouse on the assumption that customer requirements are well understood and well defined at the beginning and don't change much as the project progresses. Another fundamental problem is that processes take too long, such that when end users see the system, many things — including user requirements — have changed drastically. As said they are often too "heavy" to keep up with the pace of e-business software development projects. To help solve this problem, the so called "light" SDLC methodologies have recently been developed and put to use. They are considered light because of the reduced documentation and managerial effort required.

II TRADITIONAL SDLC METHODOLOGIES

The software development process is divided into a number of phases by SDLC, each of which is further divided into steps. Progress through the steps is measured by the completion of forms and checklists. The output produced by one phase becomes an input to another phase, in a sequential manner, and so because of this, a traditional SDLC was often called "a waterfall". Once a phase was completed, there was no returning to it as shown in Fig:1

For the waterfall approach, there are some known disadvantages:

1. If followed slavishly, it can result in the generation of unnecessary documents.
2. It is difficult for the customer to identify all requirements early in the project.
3. The customer is involved only periodically, rather than being an active participant throughout the project.

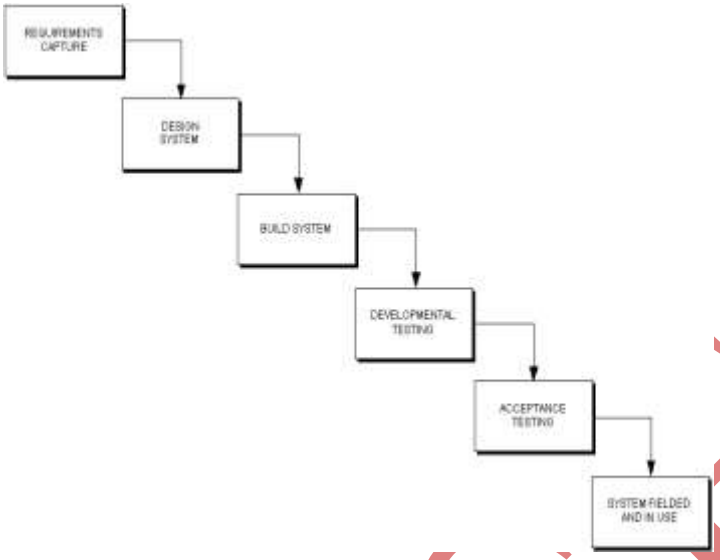

Fig: 1. Traditional SDLC (Waterfall Model)

III LIGHTWEIGHT SDLC METHODOLOGIES

In response to traditional approaches to software development, new lightweight methodologies have appeared. Light SDLC techniques are a compromise between no process and too much process. In the following sections, the lightweight SDLC methodologies are discussed. These are Adaptive Software Development (ASD), Agile Software Process (ASP), Crystal, Dynamic System Development Method (DSDM), Feature Driven Development (FDD), Rational Unified Process (RUP).

3.1 ASD

ASD is grounded in the science of complex adaptive systems theory and has three interwoven components: the Adaptive Conceptual Model, the Adaptive Development Model, and the Adaptive (leadership-collaboration) Management Model.

In contrast to the typical waterfall (plan, build, implement) or the iterative (plan, build, revise) life cycles, the adaptive development life cycle (speculate, collaborate, learn) acknowledges the existence of uncertainty and change and does not attempt to manage software development using precise prediction and rigid control strategies. There are six basic characteristics of an adaptive lifecycle The process is mission focused, component based, iterative, timeboxed, risk driven, and change tolerant.

Adaptive lifecycles are mission focused. Although the final results may be fuzzy in the initial phase, the overall mission is well defined. In addition, adaptive lifecycles are component based in the context that a group of features are developed (i.e. results, not tasks, are the focus). The process is also iterative because it emphasizes “re-doing” as much as “doing.” Another characteristic of the practice is timeboxing (i.e. setting fixed delivery times for projects).

Timeboxing forces ASD project teams and their customers to continuously re-evaluate the project's mission, scope, schedule, resources, and defects.

Lastly, adaptive lifecycles are risk driven and change tolerant. Similar to the spiral development model, adaptive cycles are guided by the analysis of critical risks. In addition, ASD is tolerant to change. The ability to incorporate change is viewed as a competitive advantage (not as a problem).

The benefits of ASD include the following:

1. Applications are a closer match to customer requirements due to constant evolution.
2. Changing business needs are easily accommodated.
3. The development process adapts to specified quality parameters.
4. Customers realize benefits earlier.
5. Risk is reduced

3.2 ASP

The Agile Software Process (ASP) was first proposed at the 1998 International Conference on Software Engineering in Kyoto Japan. Unlike traditional software process models based on volume, the ASP is time-based and quickly delivers software products. The model accomplishes this by integrating lightweight processes, modular process structures, and incremental and iterative process delivery. The ASP methodology offers five major contributions to the field. These include:

1. A new process model with a time-based enaction mechanism.
2. A software process model that provides evolutional delivery.
3. A software process architecture that integrates concurrent and asynchronous processes.
4. A process-centered software engineering environment.

However, ASP is a complex process and is therefore more vulnerable to disruption than are other lightweight and traditional SDLC methodologies.

Benefits of the ASP process are its ability to efficiently manage large-scale software development efforts . Evidence of this is the 75 percent reduction in development cycle time realized by Fujitsu when ASP was employed to manage a major communication software project.

3.3 CRYSTAL

The Crystal family of lightweight SDLC methodologies is the creation of Alistair Cockburn. Crystal is comprised of more than one methodology because of Cockburn's belief that differing project types require differing methodologies. Project types are classified along two lines: the number of people on the development team and the amount of risk.

Crystal methodologies are divided into color-coded bands. “Clear” Crystal is the smallest and lightest. “Yellow”, “Orange”, “Red”, “Maroon”, “Blue”, and “Violet” follow for use with larger groups using more complex methodologies. Each color has its own rules and basic elements. Each methodology is as light as possible and is tuned to the current project using techniques developed by Cockburn. These techniques are based on the following four principles:

1. Use larger methodologies for larger teams.
2. Use denser methodologies for more critical projects
3. Interactive, face-to-face communication is most effective.
4. Weight is costly.

Finally, Crystal methodologies are based on the premise that people issues can easily determine a project’s results (Cockburn, 2000, September). Software development methodologies should recognize this and take the characteristics of people into account. People are not just nameless resources as is the case with many traditional SDLC methods.

3.4 DSDM

The Dynamic Systems Development Method (DSDM) is a framework used to control software development projects with short timelines. It was developed in 1994 (and is currently maintained) by a consortium formed by a group companies in Great Britain. The methodology begins with a feasibility study and business study to determine if DSDM is appropriate. The rest of the process consists of three interwoven cycles. These are functional model iteration, design and build iteration, and implementation.

3.5 FDD

Feature Driven Development (FDD) is a model-driven short-iteration software development process. The FDD process starts by establishing an overall model shape. This is followed by a series of two-week “design by feature, build by feature” iterations. FDD consists of five processes: develop an overall model, build a features list, plan by feature, design by feature, and build by feature.

There are two types of developers on FDD projects: chief programmers and class owners. The chief programmers are the most experienced developers and act as coordinator, lead designer, and mentor. The class owners do the coding. One benefit of the simplicity of the FDD process is the easy introduction of new staff. FDD shortens learning curves and reduces the time it takes to become efficient.

Finally, the FDD methodology produces frequent and tangible results. The method uses small blocks of user-valued functionality. In addition, FDD includes planning strategies and provides precision progress tracking.

3.6 RUP

The Rational Unified Process (RUP) works well with cross-functional projects. Published by Rational Software, RUP contains six best practices: manage requirements, control software changes, develop software iteratively, use component-based architectures, visually model, and verify quality. RUP is a process framework and can be used in either a traditional (e.g. waterfall style) or a lightweight manner.

Finally, although RUP was originally intended to help manage software projects, its flexible design makes it applicable to large e-business transformation projects (Bloomberg, 2001). After applying a few critical augmentations to the process, RUP can effectively provide a framework for enterprise-wide e-business transformation.

IV STRENGTH AND WEAKNESSES

A recent study by the Cutter Consortium found that traditional SDLC methodologies "fall short in the new e-business environment. They are unable to keep up with fast-paced, ever-changing e-business projects". Perhaps the greatest strength of the new lightweight methodologies is that they provide a palatable alternative to the code and fix mentality that permeates today's environment. Simpler lightweight processes are more likely to be followed than traditional ones when a developer is used to no process at all. In addition, lightweight methods excel when requirements are uncertain and volatile. Traditional processes require stable requirements in order to have a stable design and follow a planned process. Another advantage of lightweight methodologies is that they force developers to think clearly about the end products they are developing.

On the other hand, one of the biggest limitations of lightweight SDLCs is their inability to handle large development teams. Although XP and Crystal have been successful with teams of 45 to 50, beyond that number there is no evidence on how to use a lightweight method.

V CONCLUSION

Lightweight methodologies are a compromise between no process and too much process. These new methods were developed to efficiently manage software projects subjected to short timelines and excessive uncertainty and change. Lightweight SDLCs are Adaptive Software Development (ASD),Agile Software Process (ASP), Crystal, Dynamic System Development Method (DSDM),Feature Driven Development (FDD), Rational Unified Process (RUP).

Strengths of these new light methodologies include their simpler processes and easier acceptance by developers who are only familiar with code and fix techniques. In addition, these lightweight SDLCs aid developers in thinking clearly about the end products they are creating. Disadvantages include their inability to handle large development teams. Lightweight methodologies are most appropriate when there are uncertain and volatile requirements, responsible and motivated developers, and customers who wish to become involved.

REFRENCES

1. https://www.google.co.in/search?q=WATERFALL+MODEL&espvd=210&es_sm=93&um=1&ie=UTF8&hl=en&tbm=isch&source=og&sa=N&tab=wi&ei=oCZHUsD6DoLUrQfuooHYCA#es_sm=93&hl=en&q=TRADITIONAL+SDLC+WATERFALL+MODEL&tbm=isch&um=1
2. System%20Development%20Life%20Cycle%20-%20Research%20Paper.htm
3. http://www.ism-journal.com/ITToday/AU4426_C016.pdf
4. http://www.scrumalliance.org/community/articles/2013/january/traditional-and-agile-methods-an-interpretation