Volume 6, Issue 11, November 2016 ISSN: 2277 128X
International Journal of Advanced Research in Computer Science and Software Engineering
Research Paper
Available online at: www.ijarcsse.com

A Comprehensive Study of Traditional and AGILE Software Development Methodologies

Priyanka[*], **Praveen Kantha**
CSE Department, MDU University,
Haryana, India

Abstract: In the development of modern society software is playing a vital role. Among a variety of software development methodologies majority speaks about two kinds of methodologies: heavyweight and lightweight. The traditional way to develop software, with comprehensive planning, detailed documentation, and costly design is considered as Heavyweight methodologies. Agile modelling that is also known as the lightweight methodologies, have gained significant attention from the software engineering community in the last few years. Agile methodologies make use of short iterative cycles, and rely on implicit knowledge within a team as opposed to documentation. This research paper describes the characteristics of some traditional and agile methodologies that are widely used in software development. Different success factors of agile methods, the success rate of agile projects and comparison between traditional and agile software development will also be discussed.

Keywords: Software Process, Software Development Methodology, Agile, Heavyweight Methodologies

I. INTRODUCTION

Software development life cycle (SDLC) illustrate the vitality of software artefact in terms of processes from its beginning to its expansion, completion and implementation, release, application and maintenance and preservation. Software development can be carried out in stages.

A set of activities that lead to the production of software product is called a software process (stage). A process can be defined as a collection of events that are used to a product by gathering a set of goals or principles. Several process models are explained in the literature such as Waterfall, Prototype, Rapid Application development(RAD),Spiral Model, Object Oriented, Agile and Component Based Development. Following are software enhancement factors

- Development speed (time to market)
- Managerial overhead
- Product excellence and Project visibility
- Risk exposure
- Customer relations, etc.

These methodologies of software development impose a disciplined process upon software development with the aim of making software development more predictable, capable and efficient. Customary or heavyweight systems are arrangement driven in which work starts with the elicitation and documentation of a complete arrangement of prerequisites, trailed by structural and abnormal state plan improvement and investigation. This approach got to be known as heavyweight because of these great viewpoints. The name "agile" came to realization in 2001. Spry or Agile approaches are picking up fame in industry in spite of the fact that they trade off a blend of acknowledged and uncertain programming designing practices.

II. SOFTWARE PROCESS MODELS

A Process Model describes the sequence of phases for the entire lifetime of a product. Therefore it is sometimes also called Product Life Cycle. This covers everything from the initial commercial idea until the final de-installation or disassembling of the product after its use.

Following are various software process models:

1. WATERFALL MODEL

he **waterfall model** is a sequential (non-iterative) design process, used in software development processes, in which progress is seen as flowing steadily downwards (like a waterfall) through the phases of conception, initiation, analysis, design, construction, testing, production/implementation and maintenance. Each phase consists on a definite set of activities and deliverables that must be completed before the following phase can begin. However, the term waterfall is used as a standard name to all sequential and chronological software engineering methodology. Figure 1 below shows a conventional waterfall lifecycle.

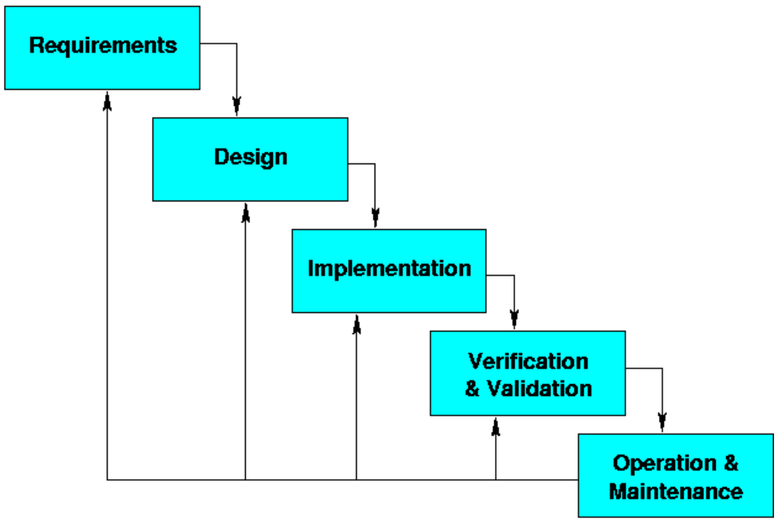


Figure 1: Waterfall Model

Advantages of Waterfall model

- The heavy documentation and structure proposal is a benefit for a new member.
- This model is Straightforward and easy to use.
- Every stage has a predictable result which is effortless to coordinate due to rigidity of model.

Disadvantages of Waterfall model

- Some requirements may materialize after the requirements gathering stage and that generate problems.
- Real projects rarely follow the sequential flow and iterations in this model are handled indirectly.
- Estimation of time and budget for each phase is very difficult.
- Very high threat in the whole life cycle of the expansion of the software product.

Figure 2 illustrates the deliverables required for each phase to be able to continue to the next.

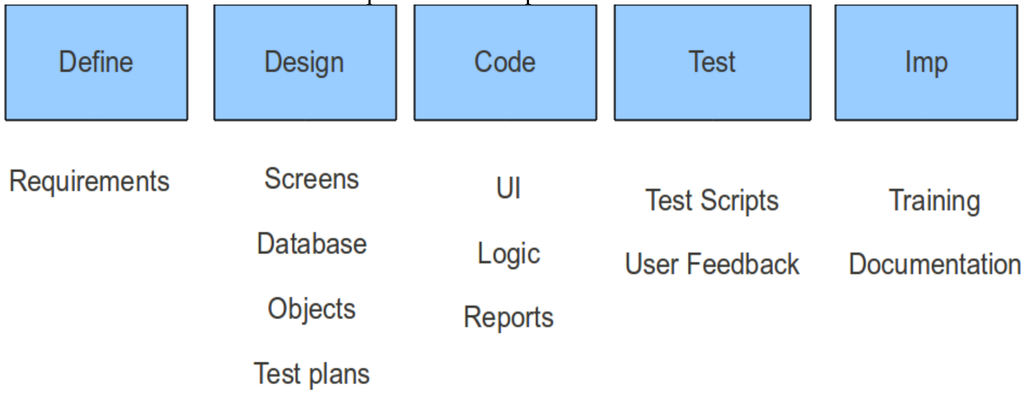


Figure 2: Waterfall Deliverables

2. SPIRAL MODEL

Spiral model joins components of both outline and prototyping-in-stages, with an end goal to consolidate points of interest of top-down and base up ideas. There are four main phases of the spiral model [2]:
A software project repeatedly passes through these phases in iterations called Spirals.

- **Identification:** Business prerequisites are assembled in the pattern winding. In the ensuing spirals as the item develops, distinguishing proof of framework prerequisites, subsystem necessities and unit prerequisites are all done. Toward the end of the winding the item is conveyed in the recognized business sector.
- **Design:** It begins with the calculated outline in the standard winding and includes engineering plan, sensible configuration of modules, physical item outline and last outline in the ensuing spirals.
- **Construct or Build:** Construct stage alludes to creation of the real programming item at each
winding. In the pattern winding when the item is just considered and the outline is being produced a POC (Proof of Concept) is created in this stage to get client criticism. In the ensuing spirals a working model of the product called fabricate is created. These manufactures are sent to client for input.
- **Evaluation and Risk Analysis:** It incorporates recognizing, evaluating, and observing specialized achievability and administration dangers, for example, plan slippage and cost overwhelm. In the wake of testing the work, toward the end of first cycle, the client assesses the product and gives input.

Following figure 4 shows the activities in each phase:

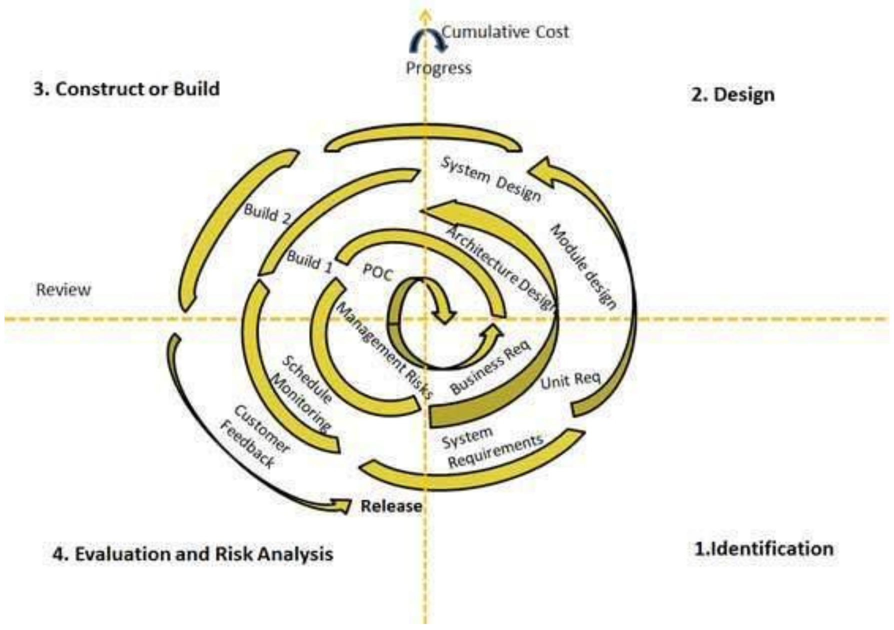


Figure 3: Spiral Model

1) Advantages of Spiral Model

- Project monitoring is very simple and effective.
- Risk administration is one of the in-assembled elements of the model, which makes it additional alluring contrasted with different models.
- Changes can be introduced later in the life cycle as well.
- Project estimates in terms of schedule, expenditure etc become more and more pragmatic as the project moves forward and loops in spiral get concluded.

2) Drawbacks of Spiral Model

- Cost involved in this model is usually high.
- It is a convoluted approach especially for projects with a clear SRS.
- Skills required, evaluating and reviewing project from time to time, need expertise and proficiency.
- Rules and protocols should be pursued properly to put into practice this model effectively. Doing so, through-out the duration of project is tough.
- Due to an assortment of customizations allowed from the client, using the same prototype in other projects, in future, is difficult.
- It is not appropriate for low risk projects.
- Amount of documentation required in intermediary stages makes management of project very complex matter.

3. RATIONAL UNIFIED PROCESS MODEL (RUP)

A substantial iterative approach that supplies to oblige change and versatility amid the advancement process is the Rational Unified Process (RUP). In RUP, programming item is outlined and worked in progression of incremental cycles.

The principle objective of RUP is to assure the generation of amazing programming by addressing the necessities of its end-clients inside of an predictable timetable and spending plan. This Model gives rules, layouts and devices to programming building group to take full favorable circumstances. Following are the guidelines for best practices:

1. Develop software iteratively
2. Manage requirements
3. Use component-based architectures
4. Verify software quality

RUP undergoes a series of phases as follows:

Inception: (**Understand what to build**)**:**The idea for the project is stated. The development team determines if the project is worth pursuing and what resources will be needed .**Elaboration: (Understand how to build it):** In this phase, the project's architecture and required resources are further evaluated. Developers consider possible applications of the software and costs associated with the development. **Construction: (Build the Product)**In this phase, the project is developed and completed. The software is designed, written, and tested. **Transition: (Transition the Product to its Users):** In this phases of two system tested, user tested, rework edand deployed.

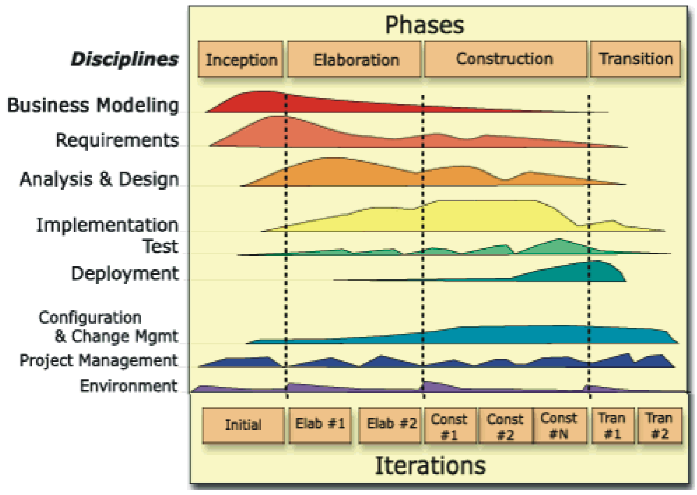


Figure 4: Phases of RUP model[1]

Applications of RUP model

a) Distributed systems
b) Very large or complex systems
c) Systems combining several business areas
d) Systems reusing other systems
e) Distributed development of a system

Advantages of RUP

- Process details are expressed in general terms, allowing local customization
- Heavy emphasis on documentation (UML)
- Can embrace incremental releases
- The development time required is less due to reuse of components.

Drawbacks of RUP

- Expertise of team members required for software development.
- Complexity is more in development process.
- Integration throughout the process of software development cause more issues during the stages of testing.
- On cutting edge projects which utilize new technology, the reuse of components will not be possible. Hence the time saving one could have made will be impossible to fulfil.

The comparison of various heavyweight methodologies on the basis of different parameters are shown in table 1 [3]:

Table 1: Comparison of various heavyweight methodologies

MODEL/MERITS	WATERFALL MODEL	ITERATIVE WATERFALL MODEL	PROTOTYPE	V- SHAPED MODEL	SPIRAL MODEL
Success rate	Low	High	Good	High	High
Cost	Low	Low	High	Very high	High
Flexibility	Rigged	Less flexibility	Highly flexible	Little flexible	Flexible
Risk analysis	High	No risk analysis	Low	Low	Low
Cost control	Yes	No	No	Yes	Yes
Reusability	Limited	Yes	Weak	Low	Yes
Risk analysis	Only at beginning	Low	No risk analysis	Low	Yes
Implementation time	Long	Less	Less	Less	Depends on project
Interface	Minimum	Crucial	Curial	Minimum	Crucial
Security	Vital	Limited	Weak	Limited	High
Expertise required	High	High	Medium	Medium	High
Simplicity	Simple	Simple	Simple	Intermediate	Intermediate
User involvement	Only at beginning	At the beginning	High	At the beginning	High
Resource control	Yes	Yes	No	Yes	Yes

III. AGILE MODELING

Agile – devoting "the quality of being agile; readiness for motion" as mentioned in the Oxford Dictionary [4] – software development methods are attempting to offer once again an answer to the eager business community asking for lighter weight software development processes.

The most well-liked agile methods include Scrum (1995), Crystal Clear, Extreme Programming (1996), Adaptive Software Development, Feature Driven Development, and Dynamic Systems Development Method (DSDM) (1995). These are now collectively referred to as agile methodologies, after the Agile Manifesto was published in 2001.

Following are the Agile Manifesto principles:

- **Individuals and interactions** - In agile development, self-organization, motivation and spur are important, as are interactions like co-location and pair programming.
- **Working software** - Demo working programming is viewed as the best method for correspondence with the client
- **Customer collaboration** - Consistent client collaboration is vital to get appropriate item necessities.
- **Responding to change** - Agile advancement is centered on speedy reactions to change and consistent improvement.

Some of the most used agile methodologies are listed below.

1. EXTREME PROGRAMMING (XP)

Extreme programming (XP) has evolved from the problems caused by the long development cycles of traditional development models [5]. The XP procedure can be portrayed by short improvement cycles, incremental arranging, persistent input, dependence on correspondence, and developmental configuration [6]. A rundown of XP terms and practices is appeared beneath [5]:

- **Planning:** The developer gauges the exertion required for execution of client stories and the client chooses the extension and timing of discharges in light of appraisals.
- **Small/short releases:** An application is created in a progression of little, as often as possible redesigned renditions. New forms are discharged anyplace from day by day to month to month.
- **Metaphor:** The framework is characterized by an arrangement of representations between the client and the developers which depicts how the framework functions.
- **Simple Design:** The accentuation is on planning the least difficult conceivable arrangement that is executed and superfluous multifaceted nature and additional code are uprooted promptly.
- **Refactoring:** It includes rebuilding the framework by uprooting duplication, enhancing correspondence, streamlining and including adaptability however without changing the usefulness of the system
- **Pair programming:** All generation code is composed by two developers on one PC.
- **Collective ownership:** No single individual claims or is in charge of individual code sections rather anybody can change any part of the code whenever.
- **Continuous Integration:** Another bit of code is coordinated with the present framework when it is prepared. Whenever incorporating, the framework is fabricated again and all tests must go for the progressions to be acknowledged.
- **40-hour week:** A most extreme of 40-hour working week else it is dealt with as an issue.
- **On-site customer:** Customer must be accessible at all times with the advancement group.
- **Coding Standards:** Coding rules exist and are trailed by the software engineers in order to bring consistence and enhance correspondence among the advancement group.
- The lifecycle of an XP project, shown in Figure 5 [6], is divided into six phases:

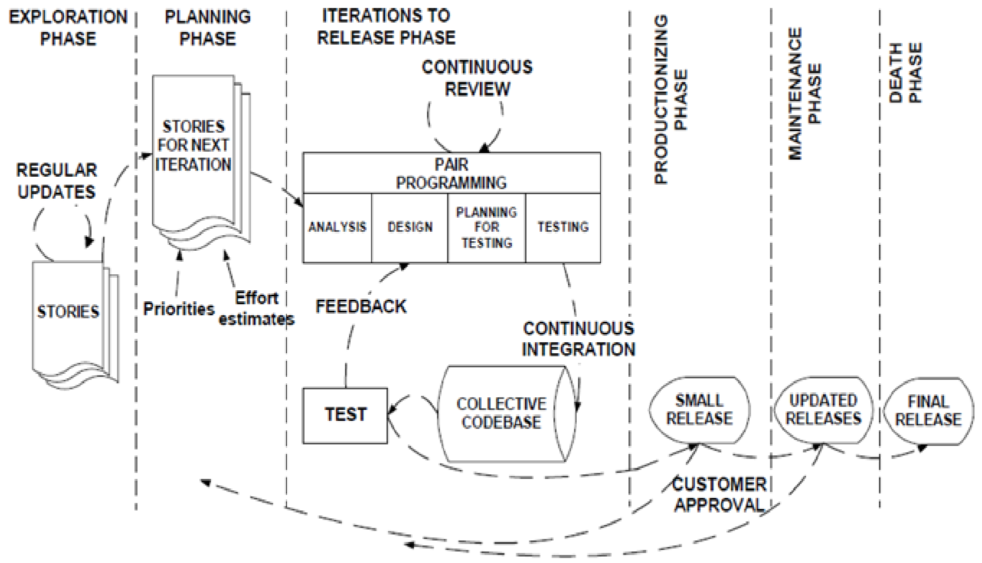


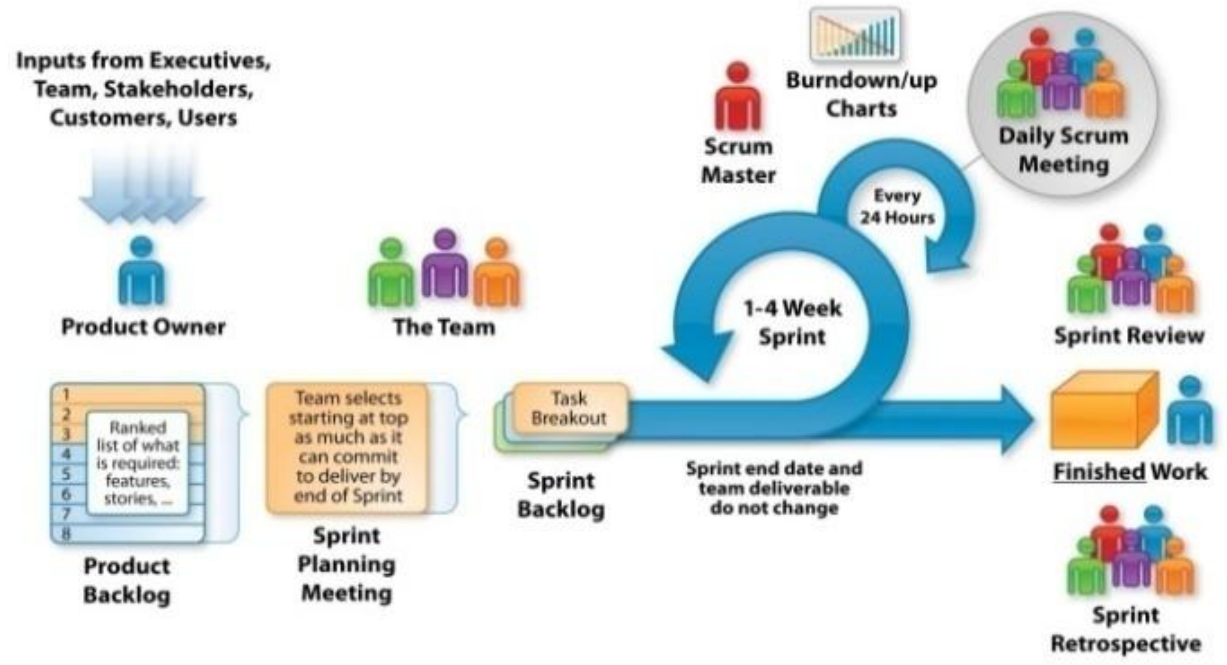
Figure 5: Life Cycle of XP Process

In the *Exploration stage*, the client works out the story cards they wish to be incorporated into their system. This prompts *Planning stage* where a need request is set to every client story and a timetable of the main discharge is produced. Next in the Iterations to *Release stage*, the advancement group first emphasis is to make a framework with the engineering of the entire framework then persistently coordinating and testing their code. Additional testing and checking of the execution of the framework before the framework can be discharged to the client is done in the *Production stage*. Put off thoughts and proposals found at this stage are recorded for later execution in the upgraded discharges made at the *Maintenance stage*. At long last the *Death Phase* is close when the clients have no more stories to be executed and all the important documentation of the framework is composed as no more changes to the engineering, outline or code is made.

2. SCRUM

Scrum is an iterative, incremental procedure for building up any item or dealing with any work. Scrum focuses on how the colleagues ought to work to create the framework adaptability in a continually evolving environment. Toward the end of every cycle it delivers a potential arrangement of usefulness [7]. Key Scrum practices are examined beneath [8]and the process is shown in Figure 7.

- **Product Backlog:** This is the organized rundown of all elements and changes that have yet to be made to the framework fancied by different performing artists, for example, clients, advertising and deals and venture group. The Product Owner keeps up Product Backlog.
- **Sprints:** hey are 30-days long; it is methodology of adjusting to the changing natural variables (necessities, time, assets, learning, and innovation). The working apparatuses of group are Sprint Planning Meetings, Sprint Backlog and Daily Scrum gatherings.
- **Sprint planning meeting:** Sprint arranging meeting is initially gone to by the clients, clients, administration, Product proprietor and Scrum Team where an arrangement of objectives and usefulness are settled on.
- **Sprint Backlog:** List of components right now allotted to a specific Sprint. When all elements are finished another emphasis of the framework is conveyed.
- **Daily Scrum:** Daily meeting for roughly 15 minutes, which are sorted out to monitor the advancement of the Scrum Team and address any impediments confronted by group.


Inputs from Executives, Team, Stakeholders, Customers, Users
Product Owner
The Team
Scrum Master
Burndown/up Charts
Daily Scrum Meeting
Every 24 Hours
1-4 Week Sprint
Sprint Review
Ranked list of what is required: features, stories, ...
Product Backlog
Team selects starting at top as much as it can commit to deliver by end of Sprint
Sprint Planning Meeting
Task Breakout
Sprint Backlog
Sprint end date and team deliverable do not change
Finished Work
Sprint Retrospective

.

Figure 6: Scrum Process [5]

The Scrum procedure may change the expected set of responsibilities and traditions of the Scrum venture group impressively. As of late, end eavors have been made to join XP rehearses with Scrum venture administration system to frame an incorporated bundle for programming improvement group [6].

3. FEATURE DRIVEN DEVELOPMENT (FDD)

Feature Driven Development (FDD) was utilized surprisingly as a part of the advancement of an extensive and complex managing an account application venture in the late 90's [7]. The FDD approach does not cover the whole programming improvement handle but rather concentrates on the outline and building stages [7]. The initial three stages are done toward the start of the task. The last two stages are the iterative part of the procedure which underpins the lithe improvement with speedy adjustments to late changes in necessities and business needs.It consists of five sequential steps (Figure 7).

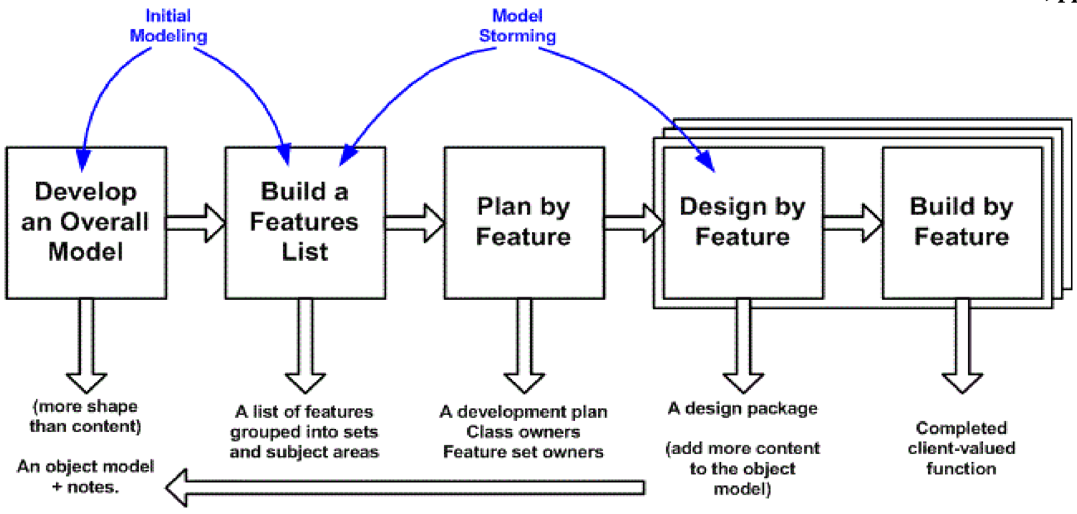


Figure 7: Feature Driven Development Process [6]

- **Develop an Overall Model:** An abnormal state walkthrough of the framework degree and its setting is performed by the area master to the colleagues and boss modeler. Recorded prerequisites, for example, use cases or useful determinations are produced.
- **Build a Features List:** A classified rundown of elements to bolster the prerequisites is delivered.
- **Plan by Feature:** The advancement group arranges the capabilities as indicated by their need and conditions and relegated to *boss developers*.
- **Design by Feature & Build by Feature:** Features are chosen from the list of capabilities and highlight groups expected to build up these elements are picked by the class proprietors. The outline by highlight and work by highlight are iterative methodology amid which the group delivers the succession charts for the doled out components. These charts are gone on to the engineers who execute the things important to bolster the configuration for a specific element. There can be different element groups simultaneously outlining and fabricating their own particular arrangement of elements. The code created is then unit tried and examined. After a fruitful emphasis, the finished components are elevated to the principle assemble.

4. DYNAMIC SYSTEM DEVELOPMENT METHOD

It is quick application advancement and Iterative improvement rehearses [9]. The basic thought behind DSDM is to settle time and assets, and afterward change the measure of usefulness in like manner as opposed to altering the measure of usefulness in an item, and afterward conforming time and assets to achieve that usefulness [10]. DSDM consists of five phases (Figure 8):

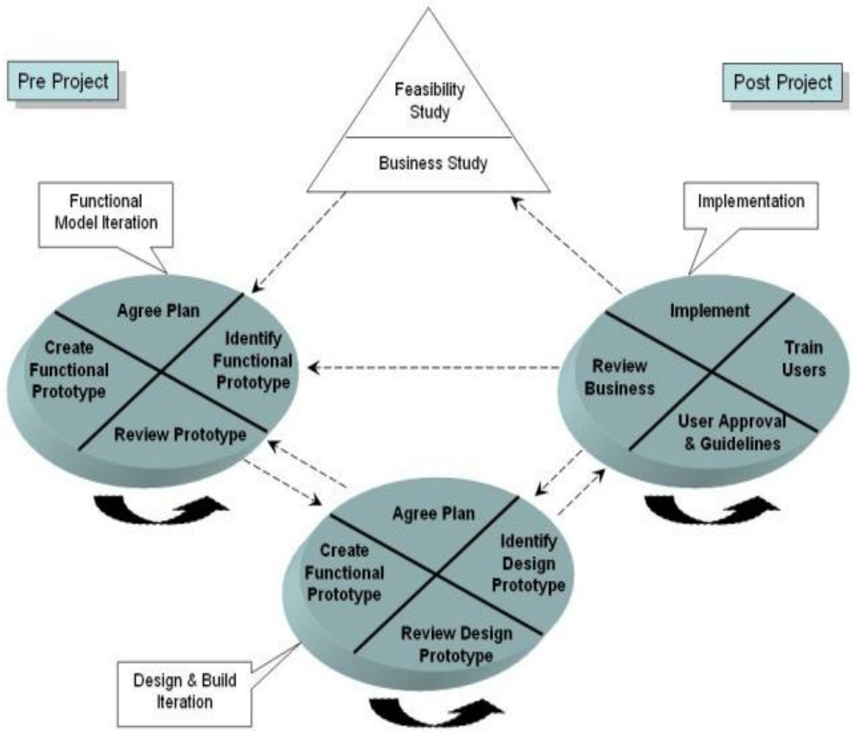


Figure 8: DSDM Process Diagram [10]

- **Feasibility Study:** In this stage a choice is made whether to utilize DSDM or not by judging the kind of venture and, hierarchical and individuals issues.
- **Business Study:** In this stage a workshop is sorted out to comprehend the business area of the venture. The key yields of this area are System design definition and an Outline model arrangement.
- **Functional Model Iteration:** The key yield is a useful model which comprises of the model code and investigation models.
- **Design and Build Iteration:** The framework is fundamentally inherent this stage. The configuration and practical models are explored by the clients and further advancement depends on the clients' remarks.
- **Implementation:** In this last stage the framework is given over to the clients. Preparing is given. Client Manuals and a Project Review Report.

A percentage of the basic standards incorporate dynamic client collaboration, continuous conveyances, enabled groups, and testing all through the cycle. There is an accentuation on high caliber and adaptivity towards evolving necessities.

5. ADAPTIVE SOFTWARE DEVELOPMENT (ASD)

Adaptive Software Development (ASD), offers a dexterous and versatile way to deal with rapid and high-change programming ventures [11]. In ASD, the static arrangement outline life cycle is supplanted by an element estimate work together learn life cycle. ASD point of convergence is on three non-straight and covering stages (Figure 9) [9]:

- **Speculate:** To characterize the venture mission, clarify the acknowledgment about what is hazy
- **Collaborate:** Highlights the significance of cooperation for growing high change frameworks.
- **Learn:** This stage focuses on the need to concede and respond to botches, and that necessities may well change amid improvement.

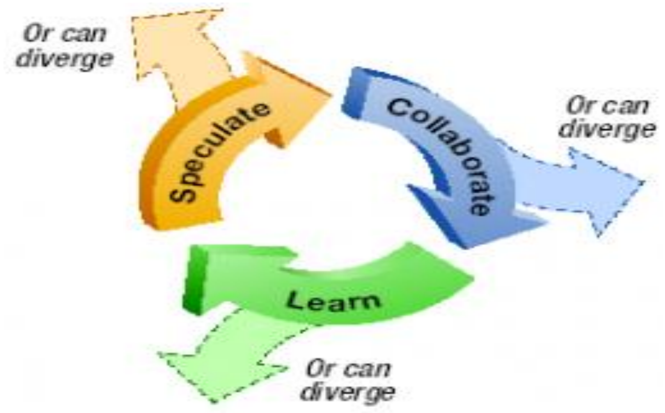


Figure 9: ASD Life Cycle [9]

ASD concentrates more on results and their quality than the errands or the procedure utilized for creating the outcomes. ASD gives a system on the best way to energize joint effort and learning inside of the venture. ASD is not introduced as a strategy for doing programming extends yet rather it is a methodology or a mentality that must be received by an association while applying deft procedures [9].

6. LEAN SOFTWARE DEVELOPMENT (LSD)

Lean Software Development (LSD) is the use of incline standards to the specialty of programming improvement. "Lean Software Development decreases surrenders and process durations while conveying a constant flow of incremental business esteem." [12]. Lean Software Development is more deliberately centered than other nimble strategy. The objectives are to create programming in 33% the time, with 33% the financial backing, and with 33% the deformity rate. [13]

6.1 The LSD Principles

There are seven LSD Principles:

- **Eliminate waste**. n programming advancement, waste is anything that does not enhance the nature of code, decreases the measure of time and exertion it takes to create code, or does not convey business quality to the client.
- **Amplify learning**. For software engineers to build up a framework that conveys business esteem, they will need to find out about numerous things. Some are specialized and others are prerequisites related.
- **Decide as late as possible**. This is the minute at which, if the group does not settle on a choice, the choice will be made for them. The advantages of this are keeping away from or deferring the expenses of progress.
- **Deliver as fast as possible**. This is the establishment of iterative improvement. Necessities change as a rate of the first prerequisites increments non-straightly as the measure of time increments. Commonplace 9-12 month

ventures produce around a 25 percent change in prerequisites. Notwithstanding, the measure of prerequisites change over a month midpoints just 1-2 percent. What's more, it is much less demanding to motivate clients to acknowledge holding up until one month from now as opposed to one year from now.

- **Empower the team**. The nature of a product group (the general population element) is the most essential component in effectively conveying programming.
- **Build integrity in**. The creators make the qualification between saw honesty and calculated trustworthiness. Seen respectability is the client's involvement with your product. Reasonable honesty is the means by which well the design and framework parts stream together to realize the apparent respectability. Clearly testing, unit and incorporation, is a noteworthy piece of honesty.
- **See the whole**. Frameworks thinking have been around for some time, however the ordinary reaction to taking care of issues is to separate them into their constituent parts and advance.

6.2 Applying LSD

There are 22 devices incorporated into the Lean Software Development. Every device is utilized to distinguish issue zones and find conceivable arrangements. Every rule likewise incorporates an "attempt this" indicating steps you can take to promptly begin applying Lean at work. Since these are standards, they will must be altered to every circumstance with a specific end goal to bring any worth. This is not something that can be purchased and tossed at a current issue with no exertion.

7. KANBAN

Kanban (Japanese for a billboard or board) is a booking framework for incline and in the nick of time generation. It is likewise a structure to bolster light-footed philosophy. Kanban controls the logistical chain from a generation perspective.

In item administration, Kanban capacities as a progression of visual work process sheets. These sheets help item directors affirm if discharges are under or over limit. At last, the objective of Kanban is to help item directors organize their work to concentrate on building what makes a difference most right at this point. The Kanban system was added to as a way to deal with incremental, transformative procedure change for associations. It was then connected to manual for programming improvement and item administration. A definitive objective was to streamline work process between these groups. Coordinated item groups regularly utilize Kanban to concentrate just on work that is effectively in advancement. Once a particular element has been sent, the item proprietor will pull another from the excess. At the point when working under the Kanban system, the item proprietor keeps up control over this build-up and is allowed to re-organize as required.

8. CRYSTAL FAMILY

The Crystal technique is a standout amongst the most lightweight, versatile ways to deal with programming advancement. Precious stone is really included a group of light-footed procedures, for example, Crystal Clear, Crystal Yellow, Crystal Orange and others, whose one of a kind attributes are driven by a few variables, for example, group size, framework criticality, and undertaking needs.

8.1 project Categories of Crystal Family

1. Projects are ordered by criticality of the framework being delivered and the span of the undertaking.
   - Four levels of criticality have been characterized, in light of what may be lost due to a disappointment in the delivered framework
   - Comfort (C)
   - Di-scretionary Money (D)
   - Essential Money (E)
   - Lif (L)

2. The most extreme number of individuals that may need to get included in a venture is viewed as the measure of the task's size.
3. A class L40 task is an undertaking including up to 40 individuals building up an existence basic framework.

8.2 +Crystal Methodologies : Family Members

1. Crystal approaches named in the writing: Clear, Yellow, Orange, Red, Maroon, Blue, and Violet (in rising request of multifaceted nature) as appeared in figure 10.
2. Others can be included if an utilization connection arrangement.

   Only those that have been for all intents and purposes utilized as a part of genuine tasks have been characterized: Crystal Orange was introduced in 1998, targeting C40, D40 and E40 projects.
   - Crystal Orange Web was introduced in 2001, and is a variant of Crystal Orange targeting ongoing web development projects.
   - Crystal Clear was launched in 2004, primarily targeted at C6 and D6 projects.

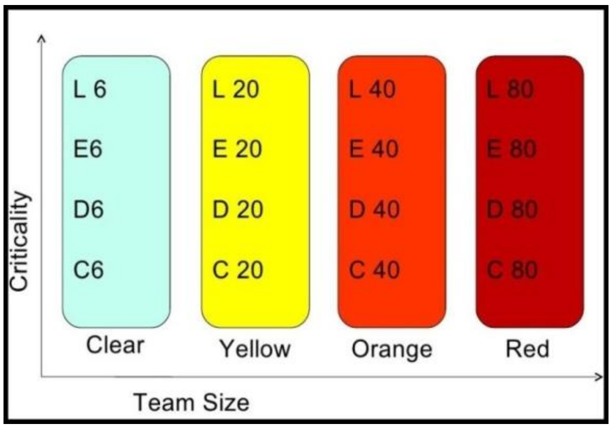


Figure 10: Crystal Methodology: Family Members

8.3 Crystal Methodologies: Flexibility

1. Every Crystal Methodologies :
    - Enforces a advance process framework.
    - Requires that a set of certain common process elements be used
    - Requires that certain work products be formed.
2. But a large body of finer-grained detail is left to the development team to decide; developers are even allowed to use procedures borrowed from other methodologies
    - The development team(s) selects a base methodology at the start of the project (in the form of a minimal set of working conventions).
    - Reflection Workshops are recurrently held to examine and refrain the process.

9. COMPARISON OF HEAVY AND LIGHT WEIGHT MODELS IN TERMS OF DIFFERENCES AND ISSUES

Comparison of Heavy and Light Weight Models in terms of dissimilarity and issues are listed in Table 2[14, 15].

Table 2: Comparison of Heavy and Light Weight Models

Differences	**Light weight**	**Heavy weight**
Customers	Committed, knowledgeable, collocated, collaborative, representative, empowered	Access to knowledgeable, collaborative, representative,empoweredcustomers
Requirements	Mainly emergent, rapid change.	Knowable early, largely stable.
Architecture	Designed for current requirements	Designed for current and foreseeable requirements
Size	Smaller Team and Products	Larger Team and Products
Primary objective	Rapid value	High assurance
Developers	Knowledgeable, collocated, Collaborative	Plan-driven, adequate skills, access to external knowledge.
Release cycle	In phases (multiple cycles)	Big bang(all functionality at once)
Issue	**Light Weight**	**Heavy Weight**
Development life cycle	Linear or incremental	incremental
Style of development	Adaptive	Anticipatory
Requirements	Emergent – Discovered duringthe project.	Clearly defined and documented
Documentation	Light (replaced by face to face communication)	Heavy / detailed Explicit knowledge
Team members	Co-location of generalist senior technical staff	Distributed teams of specialists
Client Involvement	Onsite and considered as a team member Active/proactive	Low involvement
Market	Dynamic/Early market	Mature/Main Street market
Measure of success	Business value delivered	Conformance to plan

IV. CRITERIA FOR THE SELECTION OF PROCESS MODEL

Selection of the appropriate software model to use within an organization is vital for overall accomplishment of the project. The different aspects which need to be kept in mind while decide on a suitable process model can be summarized as follows:-

- Reprocess of existing Components? Component Based approach may be ideal choice.
- Very large project with High risks or high cost of failure? A Spiral process Model may be your best choice.
- Although your customer has defined business goals but the requirement is not freeze yet Agile (light Weight) Model will have the benefits over others as it has the litheness to change the requirement at any stage.
- All the developers are experts? And if the project is small enough, an agile approach may work for you.
- Want to keep stakeholders involved? An Incremental process Model may be what you need.
- Don't have an on-site stakeholder to sit with the developers? Agile development model is not for you.
- Developers are not highly experienced? A Waterfall or Spiral process model can help keep them on track and out of trouble.

V. CONCLUSION

During the contemporary situation due to encroachment in the internet field and e-business, it is not possible for the software companies to use the traditional software model for their software development. Those companies need such software development methods that are flexible to requirements and changed requirements and can deliver the software product in short possible time and within budget.

Agile methods have been employed by software companies and the success rate of agile methods is much more than the conventional methods. In agile methods due to the customer involvement in the development phases there are no waste requirements in the development phases, and almost all the requirements are used in the software product. The software developed using agile methods is of high quality, deliver on time, within budget and there is customer satisfaction involved.

The prospect of agile methodologies seems very dominant and prevailing. In general, there are some facets of software development project can promote from an agile approach and others can benefit from a more predictive traditional approach. When it comes to methodologies, each project is different. One thing is clear: that there is no “one-size-fits all” solution.

The same study can be used in future while developing a cost benefit analysis model for software engineering or re-engineering.

REFERENCES

[1] Ashley Aitken, Vishnu Ilango, A Comparative Analysis of Traditional Software Engineering and Agile Software Development, 2013 46th Hawaii International Conference on System Sciences, 1530-1605/12 $26.00 © 2012 IEEE, DOI 10.1109/HICSS.2013.31

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

[3] Ahmad Rather, Mr. Vivek Bhatnagar, A COMPARATIVE STUDY OF SOFTWARE DEVELOPMENT LIFE CYCLE MODELS, International Journal of Application or Innovation in Engineering & Management (IJAIEM), Volume 4, Issue 10, October 2015, ISSN 2319 -4847

[4] *Oxford English Dictionary*, http://www.dictionary.oed.com, Accessed 20/4/2005

[5] K. Beck, Embracing change with Extreme Programming. *IEEE Computer*, Vol. 32, Issue 10 October 1999.

[6] K. Beck, Extreme Programming explained: Embrace change. *Reading, Mass., Addison-Wesley, Nov16, 2004*

[7] S.R. Palmer and J.M. Felsing, *A Practical Guide to Feature-Driven Development*. Upper Saddle River, NJ, Prentice-Hall, 2002

[8] K.Mar and K.Schwaber, “Experiences of using Scrum with XP”, http://www.controlchaos.com/XPKane.htm, Accessed on 2/2/2005

[9] J. Stapleton, *Dynamic systemsdevelopment method – The method in practice*. Addison Wesley 1997.

[10] Dynamic System Development Method Consortium, “DSDM Tour”, http://www.dsdm.org, Accessed 20/2/2005

[11] J. A. Highsmith, *Adaptive Software Development: A Collaborative Approach to Managing Complex Systems*. Addison Wesley 2000.

[12] Windholtz, Mark. Lean Software Development.

[13] Highsmith, Jim. Agile Software Development Ecosystems.

[14] Sheetal Sharma, Darothi Sarkar, Divya Gupta, Agile Processes and Methodologies: A Conceptual Study, / International Journal on Computer Science and Engineering (IJCSE), ISSN: 0975-3397, Vol. 4 No. 05 May 2012

[15] P. Kruchten, “What is Rational Unified Process?”, *The Rational Edge*, http://www.therationaledge.com/content/jan_01/f_rup_pk.html Accessed 2/2/2005