Comparative Study of Agile Methods and Their Comparison with Heavyweight Methods in Indian Organizations

Uma Kumari¹, Abhay Upadhyaya²

¹Ph.D Scholar, Department of Computer Science, Jagannath University, Jaipur, India
²Professor, University of Rajasthan, Jaipur, India

Email: umasecd@gmail.com, abhayu@rediffmail.com

Abstract: “Agile” means nimble or quick moving and “Agility” means the ability to think and draw conclusions quickly. In terms of software development agility means dynamic, content specific, growth oriented and able to adopt the changes quickly and easily. The roots of agile go back more than a decade but it is on the boom from the last two decades. It was February 2001, when a group of people kept a meeting in Utah, in order to find an alternative method to the existing heavy software development methodologies and the result was agile methodology. Although there are 13 agile methods and exact number of Agile method is still a debate question. I have included only 5 methods, Scrum, XP, DSDM, FDD, Lean Software

Keywords: Agile, Scrum, Extreme Programming, Dynamic Systems Development Method, Feature-Driven Development Method, Lean Software

I. INTRODUCTION

Traditional software development methods are not always feasible in rapidly changing business environment. This paper demonstrates the difference between heavyweight and agile software development. Agile Manifesto is collection of values and principles, which are found in most agile methods. Agile alliance formulated their ideas into values and further to twelve principles [1] that support those values which are as follows:

i. Highest priority is to satisfy the customer through early and continuous delivery of valuable software.

ii. Welcome changing requirements, even late in development. Agile processes harness change for the customer’s competitive advantage.

iii. Deliver working software frequently from a couple of weeks to a couple of months, with a preference to shorter timescale.

iv. Business people and developers must work together daily throughout the project.

v. Build projects around motivated individuals. Give them the environment and support they need and trust them to get the job done.

vi. The most efficient and effective method to conveying information to and fro within a development team is face to face conversation.

vii. Working software is the primary measure of progress.

viii. Agile processes promote sustainable development. The sponsors, developers and users should be able to maintain constant pace indefinitely.

ix. Continuous attention to technical excellence and good design enhances agility.

x. Simplicity- the art of maximizing the amount of work not done- is essential.

xi. The best architectures, requirements and designs emerge from self organizing teams.
Values of Agile Manifesto are given below:

i. Individuals and interactions over processes and tools.

ii. Working software over comprehensive documentation.

iii. Customer collaboration over contract negotiation.

iv. Responding to change over following a plan.

II. CHARACTERISTICS OF AGILE

Agile consists of following characteristics:

i. Modularity

ii. Incremental development

iii. Team composition

iv. People oriented

v. Cooperative

vi. Collaborative and communicating working style

vii. Maintaining the Integrity of the Specifications

viii. Lightness

III. SCRUM

Scrum is huddled mass of players engaged with each other to get a job done. Scrum for software development came out of rapid prototyping community because they wanted a methodology that would support an environment in which the requirements were not only incomplete at the start, but also could change rapidly during development. Scrum introduces the ideas of flexibility, adaptability and productivity. Scrum concentrates on how the team members should function together in order to produce the system flexibility in constantly changing environment. The main idea of Scrum is that system development involves several environment and technical values (e.g. requirements, time frame, resources and technology) that are likely to change during the process. This made the development process unpredictable and complex, requiring flexibility of the system development process for it to be able to respond to the changes. Scrum phases are introduced according to Schwaber [2].

A. Pre-game phase

This phase includes two sub-phases: planning and architecture/high level design. Planning includes the definition of the system being developed. A product backlog is created containing all the requirements that are currently known. The requirements can originate from the customer, sales and marketing division, customer support or software developers. The requirements are prioritized and effort needed for their implementation is estimated. The product Backlog list is constantly updated with new and more detailed items, as well as with more accurate estimations and priority orders.

In the architecture phase, the high level design of the system including the architecture is planned based on the current items in the product Backlog. In case of enhancement to existing system, the changes needed for implementing the Backlog items are identified along with the problem they may cause. A design meeting is held to go over the proposals for the implementations and decisions are made on the basis of this review.

B. The development phase

This phase is called game phase is the agile part of scrum approach. This phase is treated as “black box” where unpredictable is expected. The different environment and technical variables (such as time frame, quality, requirements, resources, implementation technologies and tools and even development methods) identified in Scrum, which may change during the process are observed and controlled through various Scrum practices during the Sprints, of development phase. In the development phase, the system is developed in Sprints. Sprints are interactive cycles where the functionality is developed or enhanced to produce new increments. Each Sprint includes the traditional phases of system development. One Sprint is planned to last from one week to one month. There may...
be 3 to 8 Sprints in one system. Also there can be more than one team building the increment.

C. The post game phase

This phase contains the closure of the release. This phase is entered when an agreement has been made that the environmental variables such as the requirements are completed. In this case, no more items and issues can be found nor can any new ones be invented. The system is now ready for release and preparation for this is done during post-game phase, including the tasks such as integration, system testing and documentation.

IV. Extreme Programming (XP)

Extreme Programming has evolved from the problems caused by the development cycles in traditional development models [3]. It first started as an opportunity to get job done with practices that had been found effective in software development processes. After a number of successful trials, XP methodology was developed on the key principles and practices used. The term ‘extreme’ comes from taking these commonsense principles and practices to extreme levels. The life cycle of XP consists of 5 phases:

A. Exploration phase

In this phase the customers write out the story cards that they wish to be included in first release. Each story card describes a feature to be included into the program. At the same time the project team familiarize themselves with the tools, technology and practices they will be using in the project. The technology to be used will be tested and architecture possibilities for the system are explored by building a prototype of the system. The exploration phase takes time between weeks to a few months, depending largely on how familiar the technology is to programmers.

B. Planning phase

This phase sets the priority order for the stories and agreement of the contents of the first small release. Programmers estimate the effort required by each story and schedule is then agreed upon. The time span of schedule of the first release doesn’t take more than two months. Planning phase itself takes about few days.

C. Iteration to release

This phase includes several iterations of the system before the first release. The schedule set in the planning stage is broken down to a number of iterations that will each take one to four weeks to implement. The first iteration creates a system with the architecture of the whole system. This is achieved by selecting the stories that will enforce building the structure for the whole system. The customer decides the stories to be selected for each iteration. The functional tests created by the customer are run at the end of each iteration. At the end of last iteration the system is ready for production.

D. The productionizing phase

This phase requires extra testing and checking of the performance of the system before the system can be released to the customer. At this phase, new changes may still be found and the decision has to be made if they are included in the current release. During this phase, the iterations may need to be quickened from 3 weeks to 1 week. The postponed ideas and suggestions are documented for later implementations during maintenance phase.

E. Maintenance phase

After the first release is produced for customer use, the XP project must both keep the system in the production running while also producing new iterations. In order to do this, this phase has been introduced. Maintenance phase may require incorporating new people into the team and changing the team structure.

F. The death phase

This phase is near when the customer has no longer any stories to be implemented. This requires that the system satisfies customer needs in other respects. This is the time in XP process when necessary documentation of the system is finally written as no more changes to the architecture, design or code are made. Death may also
occur if the system is not delivering the desired outcomes, or if it becomes too expensive for further development.

V. DYNAMIC SYSTEMS DEVELOPMENT METHOD (DSDM)

DSDM is widely used in U.K. DSDM is a non-profit and non proprietary framework for RAD development, maintained by DSDM consortium [4]. DSDM provides a framework of controls for RAD, supplemented with guidance on how to efficiently use these controls. The fundamental idea behind DSDM is that instead of fixing the amount of functionality in a product, and then adjusting time and resources to reach that functionality, it is preferred to fix time and resources and then adjust the amount of functionality accordingly.

DSDM consists of 5 phases: Feasibility study, business study, functional model iteration, design and build iteration and implementation. The first two phases are sequential and done only once. In the last three phases, the actual developments are done which are iterative and incremental. DSDM approaches iterations as timeboxes. A timebox lasts for a predefined period of time, and the iteration has to end within the timebox.

A. Feasibility study phase

It is where the suitability of DSDM for a given project is assessed. Judging by the type of project and most of all, organizational and people issues, the decision is made, whether to use DSDM or not. Two work products are prepared- a feasibility report and an outline plan for development. Optionally, a fast prototype can also be made if the business or technologies are not known well enough to be able to decide whether to proceed to the next phase or not. The feasibility study phase is not expected to take more than a few weeks.

B. Business study

It is the phase where essential characteristics of the business and technology are analyzed. The recommended approach is to organize workshops, where a sufficient number of customer’s experts are gathered to be able to consider all relevant facets of the system to be able to agree on the development priorities.

C. Functional model iteration phase

It is the first iterative and incremental phase. In each iteration the contents and approach for iterations are planned [5]. The iteration goes through and results are analyzed for further iterations. Both analysis and coding are done, prototypes are build and experiences gained from them are used in improving the analysis models. The prototypes are not to be entirely discarded. A functional model is produced as an output containing the prototype code and analysis models. Testing is also continuing and essential part of this phase. There are four outputs in the phase. Prioritized function is the prioritized list of the function that is delivered at the need of the iteration. Functional prototyping review documents collect the user comments about the current increment, working as input for subsequent iterations. Non functional requirements are listed, mainly to be dealt within the next phase. Risk analysis of further development is important document in the function model iteration phase because from next phase onwards, encountered problem will be more difficult to address.

D. Design and build iteration

This is where the system is mainly built. The output is a tested system that fulfils at least the minimum agreed set of requirements. Further development is based on user’s comment. The final implementation phase is where the system is transferred from development environment into actual production environment. Trainings are given to users and the system is handled over to them.

VI. FEATURE-DRIVEN DEVELOPMENT (FDD)

It does not cover the entire software development process, but focus on the design and building phase. FDD consists of 5 sequential processes.
A. Develop an overall model phase
In the beginning of this phase domain experts are already aware of the scope, context and requirements of the system to be built. Documented requirements such as use cases or functional specifications are likely to exist at this phase. After each walkthrough, a development team works in small group in order to produce object models for the domain area at hand. The development team then discusses and decides upon the appropriate object model for each domain areas.

B. Build a feature list phase
Walkthrough object models and existing requirement documentation give a good basis for building a comprehensive features list for the system being developed.

C. Plan by feature phase
It includes creation of a high-level plan, in which the feature sets are sequenced according to their priority and dependencies and assigned to chief programmers. Further, the classes identified in the process are assigned to individual developers, i.e. class owners.

D. Design by feature and build by feature phase
In this a small group of features are selected from the feature sets, and feature teams needed for developing the selected features are formed by the class owners. The design by feature and build by feature processes are iterative producers, during which the selected features are produced. One iteration should take few days to a maximum of two weeks.

VII. LEAN SOFTWARE
Lean Software Development helps software organizations to optimize their processes and production methods in order to deliver their products to the market much faster and with better quality. Lean puts main focus on people and communication. People who produce the software are respected and they communicate efficiently. It is more likely that they will deliver good product and the final customer will be satisfied [6].

Lean Software Development subsequently gave birth to agile software Development methods and its main branches are Scrum and Crystal Clear. Lean Software Development is not a management or development methodology, but it offers principles that are applicable in any environment to improve software development.

Lean software development follows seven principles which are

i. Eliminate Waste
ii. Amplify learning
iii. Decide as late as possible
iv. Defer Commitment
v. Deliver as fast as possible
vi. Empower the team
vii. Build integrity

VIII. FEATURES AND DRAWBACKS OF DIFFERENT AGILE METHODS
Features and drawback of Scrum, XP, DSDM, FDD and Lean Software are as per given in the Table I.

IX. DIFFERENCE BETWEEN AGILE AND HEAVYWEIGHT METHODS
Every method has its weakness and strengths, so depending upon need methodology is decided. Major factors that effect this selection can be categorized as per table II.

X. REFERENCES


<table>
<thead>
<tr>
<th>S. No</th>
<th>Method Name</th>
<th>Features</th>
<th>Drawbacks</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.</td>
<td>Scrum</td>
<td>Independent, self-organizing and small development teams, 30 day release cycle</td>
<td>Integration and acceptance tests are not detailed</td>
</tr>
<tr>
<td>2.</td>
<td>XP</td>
<td>Customer driven development, small teams, daily builds</td>
<td>While individual practices are suitable for many situations, overall view and management practices are less attractive</td>
</tr>
<tr>
<td>3.</td>
<td>DSDM</td>
<td>Application of controls to RAD, use of timeboxing and empowered DSDM teams. First truly Agile software development method, use of prototyping</td>
<td>While the method is available only consortium members have access to white papers dealing with actual use of the method.</td>
</tr>
<tr>
<td>4.</td>
<td>FDD</td>
<td>5 step process, object oriented component based development, method simplicity, design and implement the system by features, object modeling</td>
<td>It focuses only on design and implementation, it needs other supporting approaches as well</td>
</tr>
<tr>
<td>5.</td>
<td>Lean software</td>
<td>Focuses on people and communication, follow 7 principals, elimination of waste reduces project time and cost.</td>
<td>Decisions have to made promptly</td>
</tr>
</tbody>
</table>
### TABLE II
Comparison of Agile methods with heavyweight methods

<table>
<thead>
<tr>
<th>S. No</th>
<th>Criteria</th>
<th>Agile Methods</th>
<th>Heavyweight Methods</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.</td>
<td>Aim</td>
<td>To deliver software quickly [8]</td>
<td>Software is delivered at a defined speed</td>
</tr>
<tr>
<td>3.</td>
<td>Benefits</td>
<td>(a) Can work when requirements</td>
<td>(a) With frequent requirement changes it cannot work</td>
</tr>
<tr>
<td></td>
<td></td>
<td>constantly changes</td>
<td>(b) Cannot adopt new ideas in-between stages</td>
</tr>
<tr>
<td></td>
<td></td>
<td>(b) At the end of each stage it can</td>
<td>(c) These do not deliver product in-between production.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>adopt new ideas</td>
<td>(d) Does not allow specification changes as per user’s</td>
</tr>
<tr>
<td></td>
<td></td>
<td>(c) Agile team has launchable product</td>
<td>requirement</td>
</tr>
<tr>
<td></td>
<td></td>
<td>at the end of each tested stage</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>(d) Allow specification changes as</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>per user’s requirement</td>
<td></td>
</tr>
<tr>
<td>4.</td>
<td>Team size</td>
<td>Small</td>
<td>Large</td>
</tr>
<tr>
<td>5.</td>
<td>Documentation</td>
<td>Low</td>
<td>High</td>
</tr>
<tr>
<td>6.</td>
<td>Testing</td>
<td>Bugs are eliminated in the</td>
<td>Bugs are eliminated at the end only</td>
</tr>
<tr>
<td></td>
<td></td>
<td>development cycle and product is</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>double tested after the bug</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>elimination</td>
<td></td>
</tr>
<tr>
<td>7.</td>
<td>Success measurement</td>
<td>Business Value</td>
<td>Confirmation to plan</td>
</tr>
<tr>
<td>8.</td>
<td>Management Style</td>
<td>Decentralize</td>
<td>Autocratic</td>
</tr>
<tr>
<td>9.</td>
<td>Time period</td>
<td>Measured in weeks</td>
<td>Measured in months</td>
</tr>
<tr>
<td>10.</td>
<td>Emphasis</td>
<td>On cowboy coding (i.e. absence of</td>
<td>There exists a defined method, process oriented</td>
</tr>
<tr>
<td></td>
<td></td>
<td>defined method), People oriented</td>
<td></td>
</tr>
<tr>
<td>11.</td>
<td>Cycles</td>
<td>Numerous</td>
<td>Limited</td>
</tr>
<tr>
<td>12.</td>
<td>Return on investment</td>
<td>Early</td>
<td>Late</td>
</tr>
<tr>
<td>13.</td>
<td>Risks</td>
<td>Unknown risks</td>
<td>Well understood risks</td>
</tr>
<tr>
<td>14.</td>
<td>Architecture</td>
<td>Designed for current requirements</td>
<td>Designed for current and foreseeable requirements</td>
</tr>
</tbody>
</table>
Agile Survey : Indian Organizations

Uma Kumari
Jagannath University, Jaipur

Abstract
The software market is becoming more dynamic which can be seen in frequently changing customer needs. Software companies need to be able to quickly respond to these changes. For software companies this means that they have to become agile with the objective of developing features with very short lead-time and of high quality. In agile the goal is to understand the effect of migrating from a plan-driven to an agile development approach. Agile methods have proven to be beneficial in small projects, and there has also been growing interest in using these methods in large projects. This paper analyzes what agility and agile development are and their study in Indian organizations.

1. Introduction
During the past forty years, new software development approaches were introduced to fit the new cultures of the software development companies[1]. Most software companies nowadays aim to produce valuable software in short time period with minimal costs, and within unstable, changing environments. Agile methodologies were thus introduced to meet the new requirements of the software development companies [1].

1.1 Key features of agile methodology

- **Adaption**
The teams of developers are self-organized based on the daily meetings. Developers and customers self-organize at the end of every increment to guide the project and create the greatest value.

- **Emergence**
The architecture, team structure, and requirements emerge during the course of the project rather than being determined at its outset.

- **Modularity**
Modules are designed to work for a specific task. Agile methods break the whole task into small parts. Every part is solved individually with a very little planning. These tasks do not directly involve future planning. This means the whole concentration is on present module and its requirement at that time. The future scope of that module and what other modules will contain is not of concern at this stage. They are just containing one part and others are included at latter stages.

- **Incremental development**
The life cycle of project is divided into small iterations. Each iteration last from two to four weeks. Each iteration undergoes the whole process of software development life cycle. This includes requirement analysis, planning, design, implementation, unit testing and acceptance testing. After acceptance testing of software one iteration is over. Once it is released it is verified by the stakeholders and corrections are made. In incremental development, there are small software released with rapid cycles. So the whole application is built in small steps.

- **Team composition**
Team consists of experienced developers. It is because there is no such documentation for the project. Everything depends on how the stakeholder gives the definition of a particular module. The developer should understand the view point of the stakeholder. This understanding requires a lot of experience, so almost negligible place for fresh candidates.

- **Meetings**
Meetings provide an internal status of the project.

- **People oriented**
In agile methodology, instead of the processes and technology, people are favored. Whatever method of agile is followed, each agile team consists of one customer representative. Customer representative is the one who is involved as an end user of the project. This person is appointed by the stakeholders (customer representative who will give money). He acts on their behalf and can make any commitment required to carry on with the development of the project. He is also responsible for answering for questions raised
by the developer during any mid-iterations i.e. during development. At the end of the iteration this customer representative together with the stakeholders, review the progress of the project and re-evaluate the priorities of the modules. It is done in order to satisfy the company’s current needs and goals.

- **Contrast with cowboy coding**
  Cowboy coding is the absence of a defined method; team members do whatever they feel is right. Agile development’s frequent re-evaluation of plans, emphasis on face-to-face communication, and relatively sparse use of documents sometimes causes people to confuse it with cowboy coding. Agile teams, however, do follow defined (and often very disciplined and rigorous) processes. As with all methodologies, the skill and experience of the users define the degree of success and/or abuse of such activity. The more rigid controls systematically embedded within a process offer stronger levels of accountability of the users. The degradation of well-intended procedures can lead to activities often categorized as cowboy coding.

- **Cooperative**
  In agile, the customers and the developers work together with close communication. Most agile implementation used to have a daily meeting among the team members. Together with the developers, the customer representative or one of the stakeholders also attends the meeting. This meeting helps everyone to know the actual progress of the project. In this session each team that is handling individual modules is required to report what they have done last day and what they intend to do today. Also they discuss the difficulty they are facing. This face to face communication helps them to get the solution to their problems.

- **Collaborative and communicating working style**
  When the whole of the team working on a single project is in same location, instead of written documents, agile method to a great extent emphasize on face to face communication between the team members. So, the cost of moving the information between the people is reduced. This communication is facilitated through a single open office of agile team known as bullpen. The team size is typically small, ranging from 5 to 9 people per team. This simplifies the communication and the cooperation among the team members can be increased. It is not necessary that one team work on one module, multiple teams can work on one module. So, more development efforts can be there if they are needed. A prioritization across teams is done in order to achieve things in right manner and at right time. When team works at different locations, then contact is maintained through video conferencing, e-mails etc.

- **Lightness**
  Lightness signifies the ability to adapt to the changes. The numbers of processes that are followed during the development of project are very less. Also, the documentation part of the project takes a back seat. So, it is not necessary to provide a heavy documentation to the project at the beginning of the development itself. The documentation regarding the project is made available to the developers as and when required. Moreover the documentation are usually replaced the conversation between the people. The product that is released after iteration contains many bugs but after many iterations the product has full functionalities.

In past main methodology was heavyweight methodology which consists of heavy documentation, comprehensive planning and extensive design. New methodologies such as lightweight methodologies emphasis is on working software rather than documentation and responding to change over following a specific plan.

Due to all these problems a study of lot of agile methodologies has been worked out and their analysis has been done.

### 2. Analysis

To identify which methodologies software practitioners use in different sized companies in India for different sizes of project development, three different methods were followed. Information regarding opinions on agile methodology and heavyweight methodology was collected using questionnaire, web based survey and interviews.

#### 2.1 Questionnaire

It was divided into two sections- general questions and software based questions.
2.2 Web based survey

Survey studies a phenomenon for a population by surveying a sample that is representative for that population. Online questionnaires are preferred as they allow reaching a larger sample and are less time consuming from the researchers point of view, and are thus more efficient than interviews with regard to the number of data points that could be collected. Having collected the data of the sample statistical inference is used to draw conclusions for the overall population. Web based survey was offered over a period of three months (June, July, August) in the year 2013. In web based survey respondents were asked a total of 37 questions[2].

2.3 Interview analysis

Interviews are conversations guided by an interview protocol and are considered one of the most important resources for data when conducting case studies. The interview protocol can vary in the degree of structure, ranging from very structured (interviewee has to stick with research questions) over semi-structured (interviewee has a guide, but can change the course of the interview to follow interesting directions) to unstructured (rough definitions of topics to be covered). Very structured interviews were followed. Unstructured interviews were not considered as interviews. The data was primarily collected from a total of 11 interviews conducted at different companies in India.

Why are people abandoning waterfall and moving to agile?[4]

- Agile is Adaptive
- WYSIWYG (What You See Is What You Get) Development
- Shorter Time to Market
- Greater Employee Satisfaction
- Higher Quality
- Higher ROI

It’s sometimes hard to understand why waterfall is still practiced. But the truth is, adopting agile takes a paradigm shift in thinking that is not easy for individuals, much less organizations, can make it. It also takes experience not only in practicing agile, but also in managing organizational change.

3. Agile v/s different traditional models

Here comparison of agile with different models has been specified.

3.1 Agile v/s Waterfall model

Adoption of waterfall has helped to drive down the failure rate of software development projects, but even with rigorous project management and processes, a full 70 percent of software projects using this methodology fail to meet their objectives [3]. Organizations tried to cut the failure rate by insisting on more detail in the requirements and design phases. This process of requiring extensive, even exhaustive, documentation culminated in 1988 with the publication of the Department of Defense Standard for software development[3].
3.3 Agile v/s Spiral model

Spiral model is one of the popular process model used by industry. It is based on evolutionary approach which couples the iterative nature of prototyping with the controlled and systematic aspects of linear sequential model. The project is reviewed and the next phase of the spiral is planned [6]. It was proposed by Bhoem in 1988 and is a popular model used for large size projects.

Key difference between agile and spiral method are:

- Agile methodology is for smaller projects which consist of smaller teams but spiral model is for larger projects involving larger teams.
- Agile consists of experienced team members but spiral can consist of fresher’s as well.
- There is less documentation in agile but spiral consists of heavy documentation.

3.4 Agile v/s V-Shape model

V-shape model was developed to relate the analysis and design activities with the testing activities and thus focuses on verification and validation activities of the product. Main differences between V-shape model and agile methodology are:

- V-shape model does not support iterations of phases and dynamic changes in requirements throughout the lifecycle but agile does support.
- It does not take into account risk analysis but agile does support this.
- Testing cycle time of agile is relatively short compared to V-model, because testing is done parallel to development in agile.
- Agile model delivers a working version of the product very early compared to V-model. As more features are delivered incrementally, customer can realize some of the benefits early on.
- Agile is a proactive model (due to its very short cycles) compared to the much more reactive V-model [7].
- V-model is very rigid and relatively less flexible than agile model.
- V-model does not provide a clear path for problems found during testing phases.

3.5 Agile v/s Prototyping model

A prototype is made first and based on it final product is developed. A prototype is a model or a program which is not based on strict planning, but is an early approximation of the final product or software system. A prototype acts as a sample to test the process. From this sample we learn and try to build a better final product. This prototype may or may not be completely different from the final system we are trying to develop. This type of System Development Method (SDM) is employed when it is very difficult to obtain exact requirements from the customer. While making the model, user keeps giving feedbacks from time to time and based on it, a prototype is made. Completely built sample model is shown to user and based on his feedback; the SRS (System Requirements Specifications) document is prepared [8]. There are some differences between agile and prototyping model such as:

- Prototyping is usually done at the cost of the developer. So it should be done using minimal resources. It can be done using Rapid Application Development (RAD) tools.
- Sometimes the start-up cost of building the development team, focused on making prototype, is high [8].
- It is sometimes called as "Throw-away" prototype.
- Prototyping is a slow process [8].

3.6 Agile v/s Rapid Application Development (RAD) model

Rapid Application Development (RAD) model is incremental software development process model that emphasizes short development cycles. The Rapid Application Development model was proposed by IBM in 1980’s and later on was introduced to software community by James Martin through his book Rapid Application Development. There are some differences between RAD and agile such as:

- One of the agile methodologies DSDM uses RAD model. DSDM grew to provide an industry standard project delivery framework for what was referred to as Rapid Application Development (RAD).
- RAD was extremely popular in the early 1990’s, the RAD approach to software delivery evolved in a fairly unstructured manner. As a result, the DSDM Consortium was created.
4. Detailed findings

- **Organization size**
  As far as organization size is considered, most of the organizations were of medium size targeting around 40.5% and 26% were of small size and 33.33% of large size. Most of people who currently use agile are in Information technology field.

- **Adoption**
  From an adoption standpoint it was observed that people are using agile in India or they want to adopt agile methodology in near future. Most of the persons have average knowledge of agile methodologies and average knowledge of traditional methods.

- **ASD methodologies**
  There are several ASD methodologies available today. Scrum is most liked software development technology that is used by maximum users for small scale software development. The technology changes as the size of software to be developed increases.

- **Most preferred heavyweight methodology**
  When it comes to use of heavyweight methodology then waterfall model is the most well known model used by individuals for small scale projects. If we concern medium and large sized projects then spiral model is mostly used.

- **Improvement in technology when agile is used**
  As far as questions related to quality, productivity, business stakeholder’s satisfaction and cost of system development is concerned, maximum responded that cost of system development reduce on using agile. In case of productivity it becomes better. Maximum responded that their quality was better on using agile. When it comes to business stakeholder’s satisfaction then business stakeholders are more satisfied on using agile as compared to heavyweight methodologies.

- **Problems faced while practicing agile**
  According to finding the most common problem while practicing agile methodologies for small and medium scale projects is lack of skilled people, who can follow agile methodologies but for large scale projects it is project size or complexity.

- **Agile or heavyweight which technology is more suitable**
  It has been found that agile is more suitable technology for small scale and medium scale projects but heavyweight methodology is more suitable for large scale projects. This happens because small scale projects can be managed easily but large scale projects management becomes tough because there are lot of teams and managing large teams is a difficult task without control of management.

- **Which aspect of agile methodology appeal most**
  This question tells as that why people are moving towards agile. Maximum responded that changes can be given by stakeholders at any stage in case of agile. But in traditional methods a specific plan is followed. So maximum responded that “Respond to change verses following a plan” appeal them most to adopt agile for different type of software development.

- **Which aspect of methodology do people dislike the most**
  It was found that people dislike “low planning” in case of agile most but in case of heavyweight methods they dislike “lack of project structure” strategy.

**References**

4.025

Comparison of Agile methodology with heavyweight methods

Uma Kumari
Jagannath University, Jaipur
umasecd@gmail.com

"Agile" means nimble or quick moving and "Agility" means the ability to think and draw conclusions quickly. In terms of software development agility is dynamic, content specific, growth oriented and able to adopt the changes quickly and easily. It was February 2001, when a group of people kept a meeting in Utah, in order to find an alternative method to the existing heavy software development methodologies and the result was agile methodology. Various Agile methodologies like Scrum, Extreme Programming (XP), Dynamic Systems, Development Method (DSDM), Feature-Driven Development (FDD), Lean Software etc. are used in the paper. The comparison with Waterfall Model, Interactive Enhancement Model, Spiral Model, V-Shape Model and RAD Model are shown. Different types of the Agile methodologies and their comparisons with heavyweight methods are shown.