AGILE METHODOLOGY IN SOFTWARE DEVELOPMENT

Shivangi Shandilya, Surekha Sangwan, Ritu Yadav
Dept. of Computer Science Engineering
Dronacharya College Of Engineering, Gurgaon

***Abstract*- Looking at the software engineering principles from a historical perspective, we can see how the software processing methodologies evolved since past 50 years, but probably the most discernible exchange to software business in recent years has been the introduction of evince "Agile". Most software companies nowadays aim to produce valuable software in short time period with minimal costs, and within unstable, changing environments. As numerous areas have overblown, there is a requirement to realize the components and narration, as easily as how the agile methodologies are diverse from the traditional one. Nimble practices to develop software projects are rattling hot and are advantageously glorious today. Agile Methodologies were thus introduced to meet the new requirements of the software development companies. Methods like SCRUM, Extreme programming (XP), Feature driven Development (FDD), Adaptive software development (ASD) etc. are increasingly being used to develop software using an adaptation approach rather than a predictive one. But, there is a scarcity of the resources which describe on how these resources can be integrated with the agile methodologies. This paper presents a review of three agile approaches including Extreme Programming, Agile Modeling, and SCRUM, describes the differences between them and recommends when to use them. This paper has the following structure: Section 2 briefs the history. Section 3 explains the evolvement of software development towards agile methodologies, and presents the values and concepts of agile development. It also covers the main and most used agile methodologies. Section 4 describes the limitations to apply agile methodologies, and the last section concludes the paper.**

I. INTRODUCTION

As we notice, software development is expanding. Software has merged into many diverse fields, and is becoming more complex. Changing requirements from customers is making it even more difficult. Old software development approaches are not able to satisfy the new requirements of the market in the best way, anymore. Software development is an organized thrives to deliver products in faster, better and cheaper ways. There have been many studies and suggestion in improving the development process. Software development is an organized process that thrives to deliver products in faster, better and cheaper ways. There have been many studies and suggestion in improving the development process. As a result, new software development approaches are evolved, as agile methodologies, mainly to solve such problem. The new methodologies include modifications to software development processes, to make them more productive and flexible. To overcome the fast changing organizational business needs using traditional methods agile methods were introduced. Agile methods aid in and focus on developing solutions more quickly and efficiently. Agile methods highlight customer satisfaction by structuring the development process into iterations where in each iteration produces sizeable amount of working code and artifacts of interest to customers. Agile software development (ASD) is a relative new term within software engineering. Agile processes, or development methods, represent an apparently new approach for planning and managing software development projects. ASD differs from traditional approaches as it puts less emphasis on up-front plans and strict plan-based control and more on mechanisms for change management during the project. Despite being a new approach, the foundational principles of ASD are based on some existing principles and theories, both from the field of software engineering, information systems and others such as production management. Our paper is focused on the agile software development, agile methods centered on current practices in industry. Most commonly used methods will examined from the angle of their applicability, strengths and weaknesses and their adoption in industry. In order to

investigate and analyze, there is a need to compare the issues in the literature, research studies and industry. This will lead us to find benefits, limitations and difficulties in transition from traditional to agile software development.

II. HISTORY AND BACKGROUND

The software development methodology (also known as SDM) framework didn't emerge until the 1960s. According to Elliott (2004) the systems development life cycle (SDLC) can be considered to be the oldest formalized methodology framework for building information systems. The main idea of the SDLC has been "to pursue the development of information systems in a very deliberate, structured and methodical way, requiring each stage of the life cycle from inception of the idea to delivery of the final system, to be carried out rigidly and sequentially within the context of the framework being applied. The main target of this methodology framework in the 1960s was "to develop large scale functional business systems in an age of large scale business conglomerates. Information systems activities revolved around heavy data processing and crunching routines".

Background Agility, for a software development organization, is the power of software to choose and react expeditiously and fittingly to various changes in its surround and to the demands imposed by this surround. An agile process is one that readily embraces and supports this degree of flexibility. So, it is not simply about the size of the process or the speed of delivery; it is mainly about flexibility. This term was agreed during a big gathering when seventeen of the developers of the "lightweight" approaches to software development came together in a workshop in early 2001. Previously, circumscribe of assorted groups have independently developed methods and practices to act to the changes they were experiencing in software processing and development.

Agile Software Development is presently an emerging discipline in the field of Software Engineering. It is presently advocated by many software professionals. The Agile software development principles that are followed and advocated emerged from the traditional software development principles and various experiences based on the successes and failures in software projects. According to, customers found it difficult to define their needs because of the fast changing technology and the companies using them in products. New methods, now called agile methods are were designed to define the changing requirements in software environments. Traditional methods refer to the older and commonly used methods like the waterfall methods. These traditional methods have often been criticized to be far from the real ways software engineers functioning in developing the software. Agile Software Development emerged in February 2001 when a group of software consultants signed the Agile Software Development Manifesto. Agile methods focus on the challenges of unpredictability of the real world by relying on people and their creativity rather than processes. The main theme in agile methods is to promote and speed up responses to changing environments, requirements and meeting the deadlines. The agile manifesto states the main focus of the agile development as the following: 1) Individuals and interactions over processes and tools 2) Working software over comprehensive documentation. 3) Customer collaboration over contract negotiation. 4) Responding to change over following a plan.

III. AGILE MATHODOLOGY

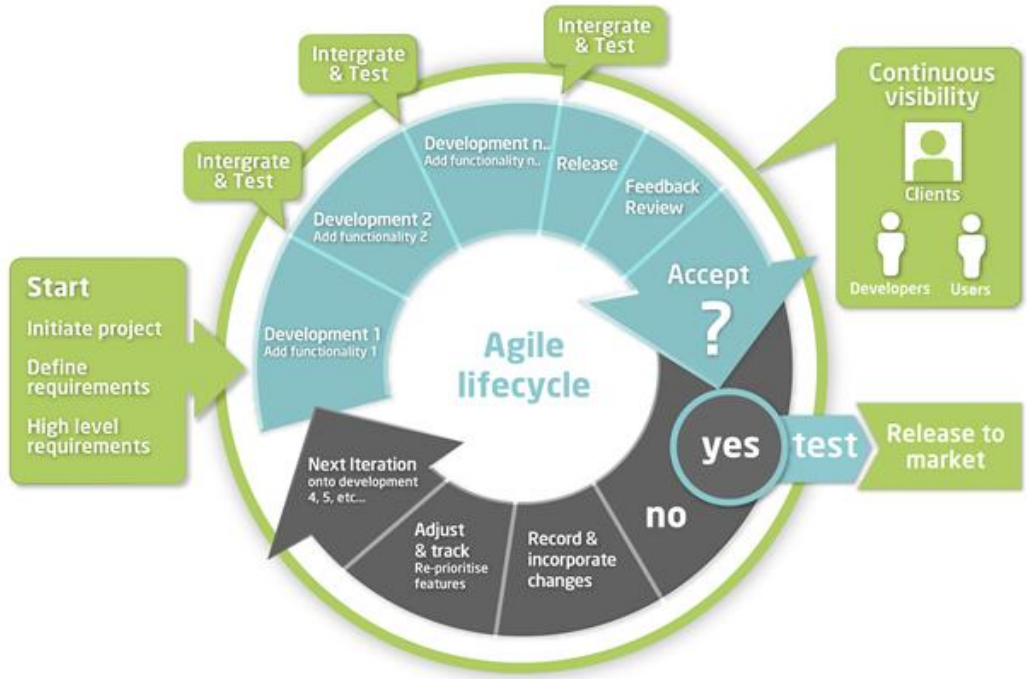
At the early years of software development, most of the users' requirements were fairly stable, and development followed the plans without major changes. However, as software development involved more critical and dynamic industrial projects, new difficulties emerged according to the growth of companies. These difficulties include:

• Evolving requirements: customer requirements are changing due to evolving business needs or legislative issues. Most of the customers do not have a clear vision about the specifications of their requirements at the early stages. Some customers realize what their true requirements are only when they use an application that does not really meet their needs. Another source of change comes from experiences gained during the development.

• Customer involvement: lack of customer involvement leads to higher chances of project failure. Many companies usually do not allocate any effort for customer involvement.

• Deadlines and budgets: often, customers do not accept failure. On the other hand, companies usually offer low budgets, tight deadlines, while at the same time, requiring high demands, and all of this is because of competition in the markets.

• Miscommunications: one cause of the misunderstanding of requirements is the miscommunication between developers and customers. For example, each party uses its own jargon, and this leads to misunderstanding of customer's needs.



Agility in short means to strip away as much of the heaviness, commonly associated with traditional software development methodologies, as possible, in order to promote quick response to changing environments, changes in user requirements, accelerate project deadlines, and the like. Agile methodologies prefer software development over documentation. Their philosophy is to deliver many working versions of the software in short iterations, then update the software according to customers' feedback. Applying this philosophy will help to overcome the problems mentioned earlier, by welcoming changes, satisfying user requirements, faster development, and at the end, users will have just the system they need. Agile methodologies include:

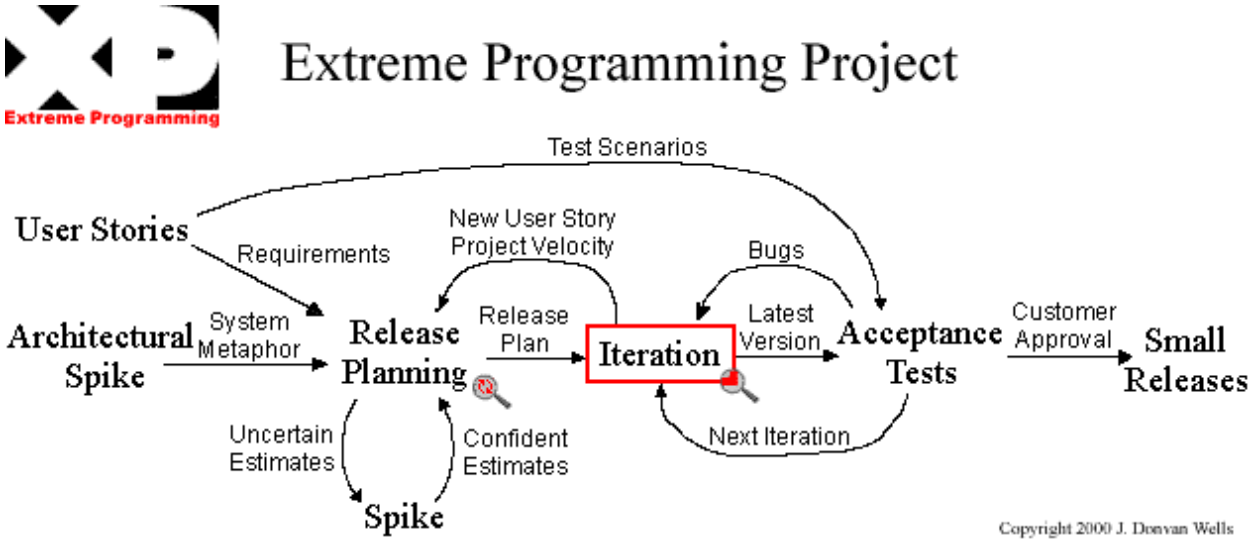
• Extreme Programming
• Agile Modeling
• SCRUM
• Crystal methodologies family
• Feature-Driven Development
• Adaptive Software Development

I. Extreme programming:-

- XP stands for extreme programming. It concentrates on the development rather than managerial aspects of soft- ware projects. XP was designed so that organizations would be free to adopt all or part of the methodology.
- XP development XP projects start with a release planning phase, followed by several iterations, each of which concludes with user acceptance testing. When the product has enough features to satisfy users, the team terminates iteration and releases the software.
- Users write "user stories" to describe the need the software should fulfill. User stories help the team to estimate the time and resources necessary to build the release and to define user acceptance tests. A user or a representative is part of the XP team, so he or she can add detail to requirements

as the software is being built. This allows requirements to evolve as both users and developers define what the product will look like.

- To create a release plan, the team breaks up the development tasks into iterations. The release plan defines each iteration plan, which drives the development for that iteration. At the end of an iteration, users perform acceptance tests against the user stories. If they find bugs, fixing the bugs becomes a step in the next iteration.
- Iterative user acceptance testing, in theory, can result in release of the software. If users decide that enough user stories have been delivered, the team can choose to terminate the project before all of the originally planned user stories have been implemented.



II. Agile modeling:-

Modeling is an important step in software development. It enables software developers to think about complex issues before addressing them in programming. Agile Modeling (AM) was established by Scott Ambler in 2002. It is a collection of values, principles, and practices for modeling software that can be applied on a software development project in an effective and light-weight manner. Agile Modeling was built to be adapted to, and used with existing methodologies, as XP and RUP, aiming to allow a developer to build a software system that truly meets the customer's needs. The values of AM, which are considered to be an extension to the values of XP include: communication, simplicity, feedback, courage, and humility. Humility means to admit that you may not know everything; others may know things that you do not know, and thus, they may provide useful contribution to the project. Again, the principles of AM are quite similar to those of XP, such as assuming simplicity, embracing changes, incremental change of the system, and rapid feedback. In addition to these principles, AM principles include the knowledge of the purpose for modeling; having multiple effective models; the content is more important than the representation; keeping open and honest communication between parties involved in the development process; and finally, to focus on the quality of the work. The practices of AM have some commonalities with those of XP, too. An agile modeler needs to follow these practices to create a successful model for the system. AM practices highlight on active stakeholder participation; focus on group work to create the suitable models; apply the appropriate artifact as UML diagrams; verify the correctness of the model, implement it and show the resulting interface to the user; model in small increments; create several models in parallel; apply modeling standards; and other practices. Agile Model Driven Development (AMDD) is the agile

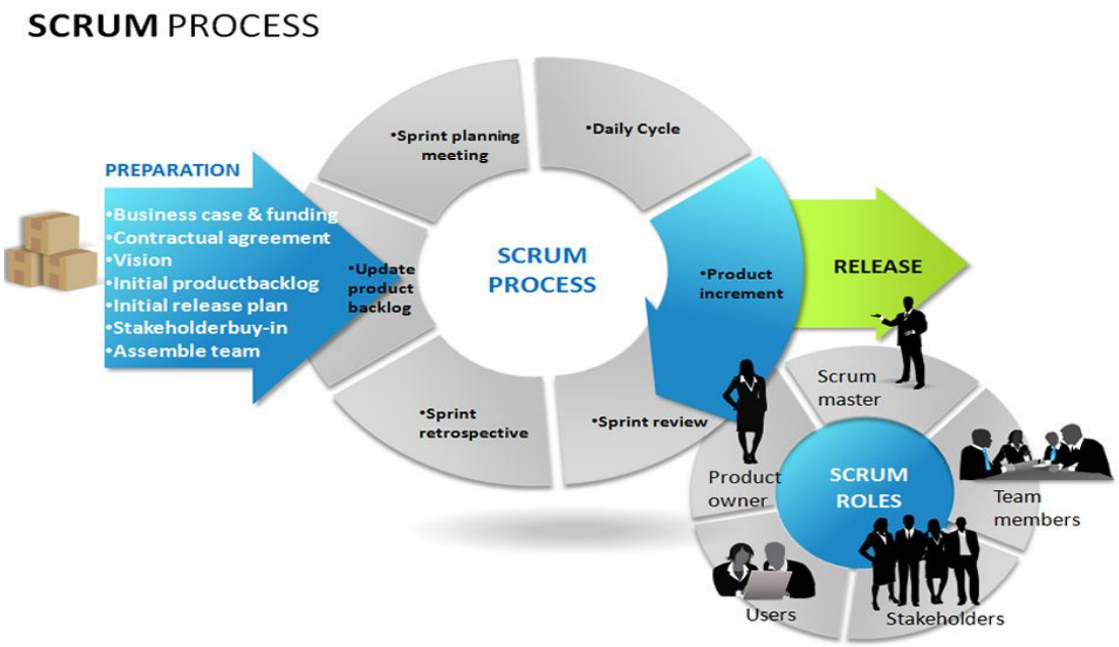
version of model driven development. To apply AMDD, an overall high level model for the whole system is created at the early stage of the project. During the development iterations, the modeling is performed as planned per iteration. Usually, AM is applied along with other methodologies, such as Test Driven Development (TDD), and Extreme Programming (XP), to get the best results. AM basically creates a mediator between rigid methodologies and lightweight methodologies, by suggesting that developers communicate architectures through applying its practices to the modeling process. In a nut, agile modeling defines a collection of values, principles, and practices which describe how to streamline the modeling and documentation efforts. It is usually applied in conjunction with agile implementation techniques for good results.

III. Scrums:-

Scrum is an iterative and incremental agile software development methodology for managing product development. It defines "a flexible, holistic product development strategy where a development team works as a unit to reach a common goal", challenges assumptions of the "traditional, sequential approach" to product development, and enables teams to self-organize by encouraging physical co-location or close online collaboration of all team members, as well as daily face-to-face communication among all team members and disciplines in the project.

A key principle of scrum is its recognition that during a project the customers can change their minds about what they want and need (often called "requirements churn"), and that unpredicted challenges cannot be easily addressed in a traditional predictive or planned manner. As such, scrum adopts an empirical approach—accepting that the problem cannot be fully understood or defined, focusing instead on maximizing the team's ability to deliver quickly and respond to emerging requirements.

SCRUM shares the basic concepts and practices with the other agile methodologies, but it comprises project management as part of its practices. These practices guide the development team to find out the tasks at each development iteration. In addition to the practices defined for agility, one main mechanism recommended by SCRUM is to build a backlog. A backlog is a place where one can see all requirements pending for a project, sized based on complexity, days or some other unit of measure the team decides. Inside a product backlog, there is a simple sentence for each requirement; something that will be used by the team to start discussions and putting details of what is needed to be implemented by the team for that requirement.



IV. LIMITATIONS OF AGILE MATHODOLOGY

- Active user involvement and close collaboration are required throughout the development cycle. This is very engaging, rewarding and ensures delivery of the right product. It's the fundamental principle in agile that ensures expectations are well managed. And since the definition of failure is not meeting expectations, these are critical success factors for any project. However these principles are very demanding on the user representative's time and require a big commitment for the duration of the project.
- Requirements emerge and evolve throughout the development. This creates the very meaning of agile – flexibility. Flexibility to change course as needed and to ensure delivery of the right product. There are two big flip sides to this principle though. One is the potential for scope creep, which we all know can create the risk of ever-lasting projects. The other is that there is much less predictability, at the start of the project and during, about what the project is actually going to deliver. This can make it harder to define a business case for the project, and harder to negotiate fixed price projects. Without the maturity of a strong and clear vision, and the discipline of fixing timescales and trading scope, this is potentially very dangerous.
- Agile requirements are barely sufficient. This eliminates wasted effort on deliverables that don't last (i.e. aren't part of the finished product), which saves time and therefore money. Requirements are clarified just in time for development and can be documented in much less detail due to the timeliness of conversations. However this can mean less information available to new starters in the team about features and how they should work. It can also create potential misunderstandings if the teamwork and communication aren't at their best, and difficulties for team members (especially testers) that are used to everything being defined up front. The belief in agile is that it's quicker to refactor the product along the way than to try to define everything completely up front, which arguably is impossible. And this risk is managed closely through the incremental approach to development and frequent delivery of product.
- Testing is integrated throughout the lifecycle. This helps to ensure quality throughout the project without the need for a lengthy and unpredictable test phase at the end of the project. However it does imply that testers are needed throughout the project and this effectively increases the cost of resources on the project. This does have the effect of reducing some very

significant risks, that have proven through research to cause many projects to fail. The cost of a long and unpredictable test phase can, in my experience of waterfall, cause huge unexpected costs when a project over-runs. However there is an additional cost to the project to adopt continuous testing throughout.

- Frequent delivery of product and the need to sign off each feature as *done* before moving on to the next makes UAT (user acceptance testing) continuous and therefore potentially quite onerous. The users or product owner needs to be ready and available for prompt testing of the features as they are delivered and throughout the entire duration of the project. This can be quite time-consuming but helps drastically to ensure a quality product that meets user expectations.

To get the advantages of applying agile methodologies in the development, there is a set of assumptions that are assumed to be true. To mention some are: cooperation and face to face relation between the customers and the development team; evolving and changing requirements of the project; developers having good individual skills and experiences; in addition to many more. If these assumptions do not apply to a software development project, then it is better to look for other methodologies to apply for the development process, in order to get better results.

V. CONCLUSION

Software development methodologies have evolved since the 1970s. Agile methodologies came into existence after the need for a light way to do software development in order to accommodate changing requirements environment. Agile methodologies provide some practices that facilitate communication between the developer and the customer, and undergo develop-deliver-feedback cycles, to have more specific view of the requirements, and be ready for any change at any time. The main aim of agile methodologies is to deliver what is needed when it is needed. Agile approaches are meant to increase flexibility, agility and to be more adjusted to the environment where software development projects are present and working today. This is a contradiction to large global project organizations with no overview and multiple interdependencies that cannot be effectively monitored. However, nothing speaks against incorporating ideas and practices from agile methods in order to increase agility even in large projects though keeping in mind that the fundamental conditions are different and that that needs to be fully understood. The ideal approach would likely be to break large projects into smaller projects which would become more flexible. This idea is brought up both by the interviewees, respondents of the survey as well as recommended by the agile methodologists.

REFERENCES

[1] http://agilemethodology.org/
[2] http://www.serena.com/docs/repository/solutions/intro-to-agile-devel.pdf
[3] http://en.wikipedia.org/wiki/Crystal_Clear_(software_development)
[4] http://en.wikipedia.org/wiki/Agile_modeling
[5] http://www14.informatik.tumuenchen.de/konferenzen/Jass06/courses/3/presentations/AgileModeling.pdf