Chapter 2

REVIEW OF LITERATURE

In this section, the literature related to the topic of this thesis is reviewed. The chapter helps in identifying the subject area precisely through what has been already proven. Firstly, in order to situate the position of agile software development among the other software development approaches, sub-section 2.1 is included which includes a discussion of the evolution of the process models used in software development. In this chapter, the software development models are divided into heavyweight or traditional or plan-driven models and iterative change-driven models. In addition, agile software development and current status of agile in software industry are defined in this chapter.

2.1 Software development process

Software development process is a series of identifiable stages that a software product undergoes during its lifetime and primary function of software development process models is to “determine the order of the stages involved in software development, evolution and to establish the transition criteria for progressing from one stage to the next [59]. Evolution of process models in the past decades (Fig. 2.1) illustrate that we are moving from heavyweight software development to iterative change-driven models of software development i.e. agile.
Software development started in 1950’s and through a number of stages and changes has reached to agile software development in 2000.

2.1.1 Plan-driven models for software development

The plan-driven way to develop software methodologies follow the generic engineering paradigm of requirements, design, build, and maintain. A system development lifecycle has three primary objectives:

- High quality system should be delivered.
• Strong management control should be there.
• Maximum productivity.

Methodologies that follow a specific plan are also called Waterfall based taking from the classical software development paradigm. They are also known by many other names like Plan driven [60], Documentation driven, Heavyweight methodologies, and Big design upfront [61]. The first publication on the Waterfall model is credited to Walter Royce’s article in 1970 [62]. Common features for the Waterfall or Plan-driven process model are defining the scope, schedule, and costs of the project including, an early fixing stage and extensive documentation. One common characteristic could also be the recurrence of the software development phases only once during the development process [63]. In this thesis, the process models of this category are referred to as heavyweight software development models.

The two-step process model of Code-and-fix, used in the early days of software development, resulted in difficulties that necessitated explicit sequencing of the phases of software development. In particular, the need to design prior to coding, to define requirements prior to design, and the need for early preparation for testing and modification were identified [59]. One of the first models to rise to that challenge was the Stage wise model as early as in the middle of the 1950s. This
model evolved from the problems caused by the increasing size of software programs, which could not be handled by a single programmer [64].

The Waterfall model is predictable and pays attention to planning the architecture and structure of the software system in detail which is especially important when dealing with large systems. Without having focus on architecture planning there is a risk that design decisions are based on tacit knowledge and not explicitly documented and reviewed [65]. The Waterfall model provided two main advances over the stepwise model. First it introduced prototyping to parallel the stages of requirements analysis and design, and secondly it provided feedback loops between the sequential stages [59]. It should also be noted that, already in the early Waterfall model [62], it had been realized that it might be necessary to first build a pilot model of the system, i.e., to conduct two cycles of development and to obtain feedback to adjust the model. This iterative feedback-based step has been lost in most descriptions of this model. Today, the Waterfall model has been adopted for most software acquisition standards in all types of industry. While the Waterfall model has solved various core problems in software development, it also includes features not appropriate for every software development context. One central problem of the Waterfall model has been identified as its emphasis on fully elaborated documents as completion criteria for early requirements and design phases [59]. Despite its problems waterfall development is still a widely used way of working in software development companies.
According to the survey study [66], despite numerous existing software development methodologies, as much as 60% of software development organizations do not apply any development methodologies. An additional problem has been identified in using a disciplined approach to software development which is that, rather than focusing on the end i.e. the development of software, developers become pre-occupied with the software development method [67]. It can be argued that the heavyweight models of software development can and should be applied in a dynamic way by repeating the phases or even the entire process, if necessary. However, the original purpose of these process models was not to welcome changes during the development, but rather to try to fix factors, such as scope, time and money, up-front in order to eliminate change which was considered a risk factor.

Larman points out that a study at United Kingdom showed that out of 6,700 projects four out of five key factors contributing to project failure were associated with and aggravated by Waterfall model, including inability to deal with changing requirements and problems with late integration. Also Larman pointed out that over 400 waterfall projects reported that only 10% of developed code was actually deployed and of that only 20% was actually used [68].

So, it is true that waterfall approach is risky and expensive to build software. This is the real reason to move towards iterative development.
2.1.2 Change driven models for software development

The central software development models, developed after the Waterfall model, seem to have the common aim of enabling, at least to some degree, the evolution of product requirements during the process of software development. This contributed one main modification to the earlier software development models: the adoption of the iterative and incremental approach. Iterative development refers to the overall lifecycle model in which the software is built in several iterations in sequence.

According to Larman, each iteration can be considered as a mini-project in which the activities of requirements analysis, design, implementation and testing are conducted in order to produce a subset of the final system, often resulting in internal iteration release. An iteration release has been defined as a stable, integrated and tested partially complete system. Incremental development involves adding functionality to a system over several releases i.e. repeating delivery of a system into the market or production. Thus, one incremental delivery may be composed of several iterations. A development approach where the system is developed in several iterations is called iterative and incremental development (IID), yet it is often referred to as iterative development [68].

Among the first models that focused on increasing the possibility of determining product improvements throughout the development process, was the
evolutionary development (Evo) model. This concept was first introduced in 1981 [69] and has been expanded [70]. This method suggested an iterative development approach in which the product increment was understood as a delivery to the real customer rather than a prototype [69]. The Spiral model of the late 1980s [68], typically consists of four iteratively repeatable steps

- Determining the objectives, alternatives, and constraints.
- Evaluating alternatives, identifying and resolving risks.
- Development and verification.
- Planning the next phase.

The Spiral model may be thought of as a meta-model (model for other models) for the software development processes. The Spiral model is a risk-driven approach for software development [59]. In the Spiral model, the iteratively evaluated strategy for resolving the risks of the next spiral has an effect on the choice of the software development approaches to be adopted.

Agile software development, which emerged in the mid-1990s, can also be classified as an iterative and change-driven software development approach. It could be argued that at present there is no common agile process model with specified phases, but there is rather a set of fundamentals [71] common to the methods claiming to be agile. The agile approach to development is about agility of the development process, development teams and their environment [60]. However, Extreme Programming (XP) [69], which is probably the best-known among the first
agile methodologies, contains an underlying process model for agile software development that has been adapted by its successors. The simplified illustration of the different software development models provides an overview of the suggested differences between the models. The common feature of agile methods is the recognition that software development cannot be considered to be a defined process, but rather an empirical (or nonlinear) one due to the constant changes that are welcomed during the development of the software product [73].

In modern iterative methods, the recommended length of one iteration is between one and six weeks, whereas the incremental deliveries are often between three and twelve months [68].

2.2 Agile software development

One of problems of TSDMs is the inability to respond to change that often determines the success or failure of a software product [74]. Because of this agile methodologies came in picture. The agile approach to software development is based on the understanding that software requirements are dynamic, where they are driven by market forces [75]. Agility is the ability to sense and response to business prospects in order to stay inventive and aggressive in an unstable and rapidly shifting business environment. Agile systems development methods emerged as a response to the inability of previous plan-driven approaches to handle rapidly
changing environments [76]. According to Williams and Cockburn agile
development is about feedback and change [73].

The emergence of agile methodologies can be said to have begun in the mid-
1990s, when software methodologies and techniques such as Extreme Programming
[77], Scrum [78], Crystal Family of Methodologies [79], Dynamic Systems
Development Method (DSDM) [80], Adaptive Software Development (ASD) [81],
and Feature-Driven Development (FDD) [82] began to emerge. In software
development, the agile movement was launched in 2001 when the various
originators and practitioners of these methodologies met to identify the common
aspects of these methods that both combined old and new ideas, and clearly shared
some particular ideologies in common. As a result, the manifesto for agile software
development was drafted and the term "agile" was chosen to combine the methods
and techniques that would share the values and principles of agile software
development. The values and principles of the Agile Manifesto [27] set out the
central elements of agility that should be embedded in any method claiming to be
agile.

Agile methodologies involve a set of best practices such as pair
programming, continuous integration or daily deployments [77]. The ideologies of
agile software development can be traced back to Lean manufacturing in the 1940s
as well as agile manufacturing in the early 1990s. Lean manufacturing is based on
the fundamentals of short-cycle time, reduced setup, multi-skilling and flow being in place while driving out waste in time, activity, inventory and space [83]. The debate between the actual differences of Lean and agile is still going on in the manufacturing sector [84], the central ideologies of both can be found in the fundamentals and methodologies of agile software development. For example, in Lean Software Development [85] the Lean principles are integrated with agile practices. Many of these ideologies and related agile software development methodologies have roots in the preceding iterative methodologies. To collect metrics to measure productivity, quality and schedule, estimation for an agile software development project using XP [86] and investigating the usage of a subset of XP practices at a group in IBM [40] was done. The product developed at IBM using XP was found to have significantly better pre-release and post-release quality compared to an older release. The teams using XP reported an improvement in productivity and morale. In addition, customers were more satisfied with the product developed using XP because the teams delivered more than what the customers had originally asked for. The development of a web based system by nine full time employees in a small company that used XP and observed substantial productivity gains compared to their pre-XP timeframe [87].
2.3 Current status of agile software development

ASD is a major paradigm in the field of software engineering which has been widely adopted by the industry, and much research, publications have conducted on agile development methodologies over the past decade. Due to constant changes in the technology and business environments, it is a challenge for traditional software development methods to create a complete set of requirements up front [75]. In the early days of agile software development, an increasing amount of interest has been paid to agile methods, by both practitioners and researchers, thus creating a growing body of empirical data on the different aspects of agile software development. Apart from the individual methods and practices of agile software development, problematic issues have arisen, such as the scalability of agile software development for large and multisite projects and the compatibility of agile methods with existing standards. Recently, the organizational and business aspects of agility have been receiving more attention [88].

In addition, there has been some confusion regarding the relationship between heavyweight coding and agile software development. It has been proposed that one reason for this confusion is the piecemeal approach of agile software development. For instance, quality in design in agile software development is prioritized in ongoing design done in smaller chunks instead of massive up-front design of the system [75]. In fact, the existing agile methodologies, such as Scrum
for agile project management and XP for implementation of software, all seem to propose a rather disciplined approach to conducting the tasks of software development [89].

T. Dyba, & T. Dingsoyr summarizes the differences between agile development and traditional development on the basis of an unpredictable world, as well as emphasizing the value competent people and their relationships to software development. Agile methods address the challenge of an unpredictable world, emphasizing the value competent people and their relationships to software development [90].

A model for integrating Scrum and XP was proposed by Zaigham Mushtaq in 2012 for good project management paradigm and to produce quality software product that is aligned with customer requirements and company objectives [91]. But this model is not tested using industrial environments.

According to a model suggested by Boehm and Turner, Carlton, Hakan et al. consider the model a useful starting point for generating hypothesis for future research, which could lead to generalizable and actionable recommendation for scientific software development community in future. According to them agile is well suited for exploratory, iterative and collaborative but agile methodologies might not be applicable in all situations pertaining to scientific software development projects [92].
According to the study of research papers and books it can be concluded that majority of empirical studies of sufficient quality focused on a single process model namely extreme programming and studies on agile often focused on agile implementation in smaller scale. In India agile is still in its embryo stage. A lot of work still can be done in this field.