# CHAPTER 4

# SOFTWARE PROCESS IMPROVEMENT

This chapter discusses the experiences of organizations that were not comfortable to detach fully from their traditional software development approaches, but wish to transition themselves to take advantage of agile software development approaches. In most cases the top management would like to play safe with funding and would not wish to take much risk. This chapter compares the Software Process Improvement (SPI) pertaining to traditional development with that of the software process improvement pertaining to agile software development.

A software process can be defined as the sequence of steps required to develop or maintain software providing technical and management framework for applying methods, tools, and people to the software task (Humphrey 1995). In traditional SPI approaches (Humphrey 1995), organizational improvement played a vital role through planning and control of SPI initiatives. Main feature with this approach noticed is the low speed with which visible and concrete results are achieved (Dyba 2000).

From the late-1990s onwards, agile software development provides a radically different approach to SPI, in which the knowledge of individual software developers and software development teams in process improvement is valued and acknowledged.

## 4.1 PLAN-DRIVEN APPROACH OF TRADITIONAL METHODS

A common feature for the plan-driven process models that evolved around 1980s is their emphasis on defining the scope, schedule, and costs of the project stated upfront with extensive documentation of the end product requirements. Plan driven approach follows phase wise development approach. The phases generally listed are the analysis phase, design phase, coding phase, testing and maintenance phase. The disadvantage in this approach is the need to complete design prior to start of coding, to define requirements prior to start of design.

Another aspect of the plan-driven approach to software development is requirements definition holding the project scope. This leads to contract decisions based on cost estimations, schedules and resources. The success of software projects is measured upfront against these estimates from the viewpoint of both supplier and the customer.

Plan-driven models of software development may repeat the phases or even the entire process, if required. The purpose of the software process models was not to welcome changes during the development, but rather to minimize changes.

### 4.1.1 Iterative Change-Driven Model

Iterative development refers to the overall lifecycle model in which the software is built in several iterations. Iterative development results in several internal iteration releases. An iteration release is a stable, integrated and tested system.

Incremental development involves adding functionality to a system over several releases. One incremental delivery may be a combination of

several iterations. A development approach where the system is developed with several iterations for each incremental release is quite common.

The spiral model consists of four iteratively repeatable steps as given in (Thangasamy 2012). They are 1) Determine objectives, alternatives, and constraints 2) Risk analysis, make prototype 3) software construction activities, and 4) planning for the next cycle with feedbacks incorporated.

### 4.1.2 Software Process Improvement using Traditional Elements

The aim of improving the software process followed in organizations is to improve return on investment over development resources through improved efficiency of development process, increased quality of the end product, higher predictability of cost and schedule and increased level of reuse. Focus on quality leads to high return on investment and competing position in the market.

One of the characteristics of improving the software process is in the emphasis on continuous improvement of organization wide processes in terms of performance, stability and capability. Both traditional methods and agile methods are classified based on four different elements for improving the software process. They are the organizational framework, tailoring the process, process deployment, measuring process parameters.

The details of traditional process model and the agile process models were discussed in chapter 2.

Elements used in software process improvement is shown in Figure 4.1.

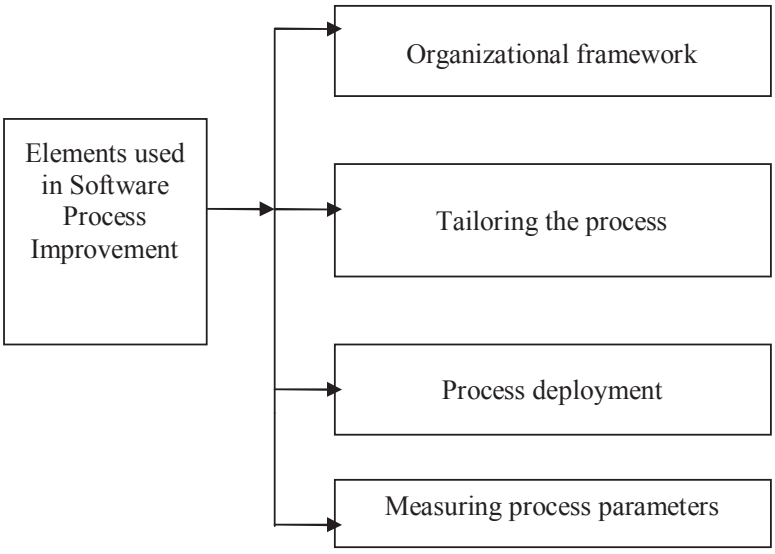


**Figure 4.1 Elements used in software process improvement**

#### 4.1.2.1 Organizational framework

The organizational framework uses continuous, top-down approach that focuses on quality improvement. The software quality improvement provides process-focused mechanisms to improve the quality and productivity of software. The software process is responsible for setting of goals and scheme of improvement, and analysis and storing of experiences and acquiring feedback resulting from the project learning cycle.

#### 4.1.2.2 Tailoring the process

Process tailoring is done based on the project specification. It has been suggested that it is done either up-front, while included as part of a project plan, or conducted as dynamic tailoring when the need is identified during the progress of the project (Fitzgerald et al 2000).

#### 4.1.2.3 Process deployment

The deployment may include activities such as piloting the processes and evaluating their effect on the software development. An organization may adopt big bang or an evolutionary approach for deploying new processes, practices and tools. They require repetitive activation of different deployment related mechanisms and also for overcoming resentment among practitioners due to the frequent changes they are requested to implement. Big bang approach has been suggested in cases where the business is no longer capable of achieving its purpose with the current process and the external circumstances are changing rapidly. This requires the motivation and activities of its members to be changed at a fast pace.

#### 4.1.2.4 Measuring process parameters

Various measurement mechanisms provide quantitative support for understanding the current status, planning improvements and assessing learning needed to ensure gains. Software processes as well as their outputs have measurable attributes which can be observed to describe the quality, quantity, cost, and timelines of the results produced. Software metrics are used for measuring specific attributes of a software product or a software development process. This is done in order to figure out the cost estimates, track the progress, analyze the defects and thus assess the SPI.

## 4.2 AGILE SOFTWARE DEVELOPMENT

Features of agile method are to make the teams to self organize and determine the management of work during the project (Boehm and Turner 2005). Uncertain schedule, scope and budget are the key aspects in agile software development.

### 4.2.1 Software Process Improvement using Agile Software Development

When considering the relationship between agile software development and SPI, there are three principles from agile manifesto. They are, valuing of individuals and interactions over processes and tools, the principle that encourages regular reflection by software development teams in order to become more effective, and the self-organization of software development teams. Taking regular improvement within project teams as one of the twelve principles of agile software development highlights the importance attached to continuous improvement in the agile software development. In order to welcome changes throughout the agile software development project, the software process along with its practices must be able to continuously adapt to the specific context.

#### 4.2.1.1 Organizational framework

In (Boehm and Turner 2005), the authors have addressed the difficulty of scaling the organizational framework in integrating agile methodologies with traditional top down systems of development. Degree of documentation and the infrastructure requirement are some of the issues of concern to agile development.

#### 4.2.1.2 Tailoring the process

Agile specific methods are needed to tailor agile software development practices within individual projects and within the entire organization. The tailoring activity is broadly classified into two groups. 1. Static tailoring - At the beginning of the project, 2. Dynamic tailoring – that takes place throughout the life-cycle of an individual project as a continuous process adaptation.

Three different strategies for process tailoring in the context of agile software development were suggested by Keenan (2004).

1. Using a comprehensive pattern based process framework for selecting appropriate elements at the beginning of each project which is a static tailoring approach,

2. Defining a set of processes, as in Crystal, selecting the best match for the project at hand with possible fine tuning which is again a static tailoring approach,

3) Defining a tailored process for the project by blending ideas and techniques from best practices and local experience which is an automatic dynamic tailoring approach.

The objective of the agile project management is to define the approach the project team will use to deliver a product as given by Highsmith (2009). Agile project management has the process and practice tailoring approach that includes both static tailoring at the beginning of the project and dynamic tailoring throughout the project that is to be conducted iteratively.

#### 4.2.1.3 Process deployment

The deployment process consists of six steps as: 1. To assess the current state, 2. To set and to modify goals, 3. To detect risks, 4. To plan the mode of execution or the process, 5. To execute the planned process, and 6. To measure the process implementation. The deployment steps identified include decision making and make certain the use of process knowledge gained during current iteration.

#### 4.2.1.4 Measuring process parameters

As our research focuses on scrum framework and extreme programming methodology that was carried out in Indian software organizations, the process method in scrum framework is measured with the help of information radiators such as story boards and burn down charts. Story board acts as an information radiator. Tasks are written as small user stories. Big user stories are called as an epic. They are placed onto the white plain boards called the scrum boards. Team members are allowed to pick the user stories as per their wish and the priority assigned on the stories. A burn down chart is a graph displaying the cumulative work remaining on a daily basis, designed to guide the development team towards an on-time and successful sprint. Burn down chart act as an information radiator.

## 4.3 DIFFERENCES BETWEEN TRADITIONAL SOFTWARE PROCESS IMPROVEMENT AND THE AGILE SOFTWARE PROCESS IMPROVEMENT

The differences between traditional software process improvements with that of agile software process improvement are tabulated in Table 4.1.

**Table 4.1 Difference between traditional SPI and Agile SPI**

| Traditional SPI | Agile SPI |
|---|---|
| High predictability and stability are emphasized | Flexible, adaptive and facilitative approach is required |
| Goals are defined by planning upfront. | Short term goals are defined through daily working practices. |
| Based on the specification of the project, process tailoring is done. | Process tailoring is done based on changing project needs. |
| Software metrics are used to calculate the estimates, progress tracking and to analyze the defects | Progress tracking is calculated with the help of information radiators such as story boards and burn down charts. |

## 4.4 MANAGEMENT OF TESTING CARRIED OUT IN TRADITIONAL SOFTWARE DEVELOPMENT APPROACH AND IN AGILE SOFTWARE DEVELOPMENT APPROACH

Testing is done in both the process models, but the way of execution differs. It depends on the team member responsible for using and the time of execution varies based on the approach that is being used. The differences in testing practice while following the two approaches are summarized in Table 4.2.

**Table 4.2 Differences in testing practice while following the two approaches**

| | Traditional Testing | Agile Testing |
|---|---|---|
| **Change in requirements** | Should be separately managed | Accept when it comes |
| **Planning** | Detailed upfront test design has to be given | Can be planned when the team takes up a new functionality |
| **Documentation** | Good documentation has to be maintained | Only as much as needed for the work |
| **Handoffs** | Should have a formal entrance and exit criteria with client signoffs whenever needed | Can be collaborated with the existing development team |
| **Test Automation** | It is done after the full code has been created | All levels, built by anyone from the team, form an integral part of the project |

### 4.4.1 Execution of Different Types of Testing in Agile Development Approach

The testing is generally classified into different types such as unit testing, integration testing, functional testing, system testing, regression testing and performance testing. Details on special features of test cases selections, their stage in the process iterations while following agile approach is the responsibility of the developer and the tester to generate test cases and carry out testing at various stages. Table 4.3 shows the different types of testing and the execution context.

**Table 4.3 Tests during agile development**

| Type of testing | Responsible | When to execute |
|---|---|---|
| Unit testing | Developer | Immediately after Coding |
| Integration testing | Developer /Tester | Throughout the sprint |
| Functional testing | Tester | Throughout the sprint |
| System testing | Tester | Throughout the sprint |
| Regression testing | Tester | Throughout the sprint |
| Performance testing | Tester | Stabilization sprints/Last sprint in the release cycle |

## 4.5 CONCLUSION

Many organizations are working with both traditional software development approaches and agile software development approaches on the projects simultaneously. For such organizations, both traditional SPI and agile SPI have to be balanced together in order to benefit from the strengths available in each approach. Hierarchical and departmentalization used in traditional projects can be combined with quick minimum marketable feature

based delivery in agile projects. Limited customer interaction in traditional projects and mission critical projects do not favor exploiting features of agile approach.

Although SPI of both the process models are dealt with, success factor gets increased only when organizations fully transition their approach to agile software development. Role of agile coach in SPI will be highly challenging and will be highly demanding during initial stages of the team members transitioning from the traditional development to agile development. Higher customer satisfaction and higher quality of end products can be obtained through face to face communication and continuous collaboration with incremental marketable feature delivery of working software.