CHAPTER – 3
A CONCEPTUAL MODEL FOR IMPLEMENTING LEAN SIGMA TO SOFTWARE DEVELOPMENT PROCESS
CHAPTER – 3

A CONCEPTUAL MODEL FOR IMPLEMENTING LEAN SIGMA TO SOFTWARE DEVELOPMENT PROCESS

3.1 Introduction

In the review of literature, it was observed that there is no standard framework or model for implementing Lean Sigma. It needs to be formulated around a specific business context by leveraging the advantages of both philosophies. This chapter presents a conceptual model for implementing Lean Sigma to improve the software development process.

3.2 Design for Improved Yield and Accuracy (DIYA)

The implementation of Lean Sigma tools and techniques to a software development process is about the customization of tools and techniques that have been applied to the manufacturing industry. The implementation framework should always be process focused and help align goals and objectives of Lean Sigma projects with organization’s strategic goals and objectives. We propose a conceptual model – *Design for Improved Yield and Accuracy (DIYA)*, for software development projects. The proposed conceptual model is shown in Figure 3-1. This model is derived on the basis of the literature review, wide discussions held with software professionals and process practitioners from the industry, and researcher’s 11 years of experience as a software professional handling software development projects and initiating Lean Sigma approach to improve software development process for leading software companies in India and the United States.

This conceptual framework proposes twelve important steps for implementing Lean Sigma to improve the capability of the software development process. This can be applied to any software development project.
Figure 3-1. The proposed conceptual model - Design for Improved Yield and Accuracy (DIYA) for implementing Lean Sigma to improve software development process

(Source: Developed by the researcher)
3.3 Twelve Important Steps in Implementing Lean Sigma to Software Development Process

**Step 1: Get commitment from top management:**
Lean Sigma is a business strategy and methodology that increases process performance resulting in enhanced customer satisfaction and improved bottom-line results. It is not just another quality improvement program (Pamfilie et al., 2012). The Lean Sigma implementation effort will succeed based on the commitment and involvement from the top management. If the senior executives are not involved, Lean Sigma program will have high probability of failure (George, 2003). Moreover, top management’s involvement definitely has a significant influence on the software development team. The top management’s attitude may lead to a positive employee perception regarding his involvement in implementing Lean Sigma method to improve software development process. Therefore, there is a need to develop and explicit guidelines on how top management should be involved and committed to increase the effectiveness of Lean Sigma implementation (Nonthaleerak and Hendry, 2008).

**Step 2: Achieve strategic fit**
In today’s competitive software development market, organizational survival and growth require effective means to align organization business objectives with software development project. At the organizational level, an effective management system offers numerous potential benefits such as focus on areas for financial critical success, effective use of resources, analysis of market potentials and opportunities for innovation, development of a learning environment, etc.

The goal of a software development project must fit with the organization’s competitive strategy. To achieve this goal, a company must select right projects that will create the highest return on invested capital (ROIC) and revenue growth. The value opportunity of projects must be clearly identified and projects must be well aligned with corporate business strategy. It is our observation that many software companies have formal project selection committee to help select high-potential projects.

**Step 3: Empower project team**
While top management commit and provide adequate resources for Lean Sigma deployment, the development team members should be empowered to take responsibility for the Lean Sigma initiatives. The success of Lean Sigma requires innovative way of
solving the problems in the software development process. Project team members should be encouraged to think innovatively. From the goal theory perspective specifying monetary goals (cost saving or revenue) motivates team members to achieve the desired outcome (Linderman et al., 2003). Accordingly, empowering project team members enables them to take appropriate decisions so that they can meet the goals. In that regard, the top management needs to empower development team members while making them financially responsible for their decisions and actions regarding the implementation of Lean Sigma to improve the capability of the software development process.

**Step 4: Enable innovation**

To stay competitive in today’s global business environment, organizations must innovate. Software development needs higher levels of innovative and intellectual thoughts about specific changes to the process to bring desired improvement in the process capability. But don’t innovate for the sake of it. It should be in line with the customer needs and market insights, not simply the development teams feeling and assumption that it needs to innovate.

To enable innovation top management should craft innovation vision based on keen understanding of market demands. Goals and objectives should be explicit and few in number to enable focus. Top management must be committed to establish enduring process, not just launching another initiative (Byrne et al., 2007).

Innovation, in the context of software development, is much broader than just product innovation - it is about how the development team handles requirement changes, how they optimize the software development process, and how they create and deliver value to the customer with higher quality, lower cost and shorter lead-time in order to be successful.

**Step 5: Create value for the customer (both internal and external customers):**

In software industry, between 30 and 50% of all features are unnecessary and add overhead (Ebert et al., 2012). According to the Standish Group Study (2002), about 45% of the features of software applications were never used. From this statistics (Figure 3-2), most of the features/functions in any software applications don’t add any value to the customer. Rather they create more defects, complexity into the software application, and thereby increase unnecessary cost to the project. Therefore, don’t build features that
nobody needs now; don’t write more specs than you can code; don’t write more code than you can test; and don’t more code than you can deploy (Corey Ladas, 2009).

Figure 3-2. Statistics on the usage of software features/functions

(Source: The Standish Group, 2002)

The critical step in implementing Lean Sigma is to create value for customers. Value is expressed in terms of how a specific software application meets the customer’s needs. Determination of which features and/or functions create value in the software application should be made from the perspective of the end-user or a subsequent process/owners.

**Step 6: Develop initial value stream:**

Once the value is defined from the customers’ standpoint, the next step is to identify the activities that deliver value to the customer. The entire sequence or stream of activities that contribute value is called the value stream.

Value stream map is widely used in Lean manufacturing. Its utility is comparatively unknown in the software industry. According to Ted Rivera of IBM, value stream map exists for two purposes – to identify and end wasteful activities, and to stop waste from happening in the first place (Rivera, 2010).

For the software development process, the value stream mapping begins with identifying the current steps in the process. The current value stream map shows work processes as they currently exists (Pillai et al., 2012). Once the each phase is identified in the current value stream mapping, the number of activities (in detail) in each stage is identified. Calculate the work time and waste time for each step by identifying how long each step is going to take. Then total work time and waste time for the entire process is calculated. Here the development team has got lot of opportunities to identify the value-add and non-value-add activities. Then more efficient and waste-free value steam can be tailored. The value stream mapping is the first step to transform to pull system. It is a very efficient tool.
Step 7: Establish real-time pull system (Kanban):

Kanban brings incremental improvements to the current process. It gradually eliminates the whole idea of iterations or sprints. Through continuous delivery of high-priority features/functions, it focuses on creating a highly responsive team rather than fitting the software development into rigid time-boxed iterations.

Kanban approach to software development means that only software application with features/functions requested by the client will be developed. This approach helps eliminate what is known as gold-plating, where developers include apparently sophisticated ‘bells and whistles’ that may only be interesting to developers, which have not been requested by client, and may never be used (Morien, 2009).

Pull system (or Kanban) for software development was developed by David J. Anderson in 2006. It is a method for making incremental and evolutionary improvements in the software development process. Pull approach limits the number of features/functions locked in work in progress (WIP) and the next work item will be taken only when the current prioritized feature/function is developed. This helps to balance the overall development activities by eliminating the clog in the development cycle. As a result, pull system (or Kanban) method will help achieve lower cost, high quality, better productivity, and lesser lead time.

Step 8: Measure current performance:

It is impossible to manage what we do not understand. And we cannot understand what we cannot measure – size matters most! Software size is extremely important in determining and controlling costs, schedules, quality and productivity in software project management.

The common problems that software development team encounter in instituting software measurement program include too much data collected, not the right data collected, and insufficient analysis of data that is collected. This leads to numerous problems, including decreased cost effectiveness of the measurement program and disillusionment about metrics on the part of developers and managers. The end result is often the eventual failure of the measurement program as a whole.

In response to such problems, several structured approaches to software measurement have been developed and are used in organizations. These approaches are referred to as
“goal-oriented” approaches because they use goals, objectives, strategies, or other mechanisms to guide the choice of data to collect and analyze in a systematic way. (Basili et al., 2007)

Put a software measurement framework in place before starting LSS project. According to Software Productivity research, software sizing with Function Points can help significantly improve the probability of completing a software project successfully – on time and within budget. Figure 3-3 shows the statistics of software projects done without and with FPA.

*Figure 3-3. Statistics of software projects done without and with FPA*

The FPA technique quantifies the user functions that are meaningful to the software user. The Function Point measure relates directly to the business requirements, it can be applied throughout the software development cycle – from requirement analysis to release. Function Point sizing can provide a solid basis of estimate for cost and schedule plans at the beginning of the project.

**Step 9: Identify and eliminate or automat non-value-added activities:**

Cycle time is the time measured when the actual work begins on the request and ends with the status ready for delivery. On the other hand, the lead time is measured from the time when the request is made and ends at delivery. Lead time is more important from the customers’ point of view, while cycle time is the way of measuring the process capability by the developers. Lead time is much longer than the cycle time.

During software development, work items spend a lot of time in queues – like waiting to be reviewed, waiting for documentation update, waiting to be signed off, handoff to next stage, etc. It is this queue that creates a bottleneck in the workflow and affects the overall lead time. A quick way to reduce the lead time is check for these queues and figure out a way to eliminate or reduce them. Or set limits on the amount of work that the team takes
on at one time – stop upstream activities until capacity is available downstream (Govindaraj and Tadipatri, 2011).

Pull system (or Kanban) does not limit the time needed to complete work item, rather it enforces to set limitations on WIP. WIP Limit is a limit that governs the number of work items that can be in that state at any instant. Kanban uses these work in progress limits to change the basic project management paradigm from pushing work downstream to pulling work from upstream. If the state is below its limit, it may take possession of the work item from upstream. If a state is at its limit, it must wait for one of its items to be completed, and pulled from downstream.

**Step 10: Analyze opportunities:**

To analyze opportunities, the development team studies the information gathered in the Measure step to determine the cause-and-effect relationships in the process and to understand potential root causes for variations, pinpoints bottlenecks, and identifies improvement opportunities. It is important to separate the sources of variations into common causes and assignable causes. Removing a common cause of variability usually means changing the process, while removing an assignable cause usually involves eliminating a specific problem. The Analyze performance phase is when technological recommendations are provided.

Facts with software development: Process variations can never be eliminated or even reduced below a moderate level. This is because – no two features/functions are alike so process performance always includes an intrinsic degree of variability. And – there are very large differences in skills and experiences from one developer to another that cause dramatic differences in process performance.

There are many statistical tools that are potentially useful in the Analyze step. The majority of these tools and techniques are directly applicable to everyday software development data analysis. They primarily include fishbone diagram, Pareto chart, process mapping for work flow optimizing, and control charts.

Look for the areas where you think you have got a problem – for example – defect rate is too high; schedule slips too often; too many misunderstandings between developer and customer; etc. Too many defects produced each week – then start collecting metric on number of function points completed each day, module/s that is having more number of
defects, developers and testers working on that module. Frequency of requirement change, number of hours each engineer spend in meeting, number of projects each engineer working simultaneously on different projects, and other possibly relevant facts

Step 11: Improve performance:
In the improve performance stage, the recommended solutions to fix the process issues are implemented. An implementation plan is developed and put into action through a pilot project. Wherever appropriate, workflows are streamlined, unnecessary documentations are eliminated, non-value-added activities are removed, certain activities are automated, new technology is implemented, and consulting services are initiated. Key factors of success during this stage are acceptance by end users or subsequent process owners, and process-wise change without any degradation in the current process capability and productivity.

Step 12: Control and strive for excellence:
This is the final step. In this step, better solution is implemented and ensures that the benefits derived from the improvement are maintained. Once a solution is implemented, place the necessary controls to assure improvements are maintained long-term. This involves monitoring the key process metrics to promote continuous improvement and to guard against regression. To achieve excellence, this effort is repeated and continuous attempt to remove non-value-added activities, reduce complexity, reduce re-work, improve cycle time and lead time to deliver value to the customer.