Software Process Improvement –TRIZ and Six Sigma (Using Contradiction Matrix and 40 Principles)

Garikapati Pavan Kumar
Email: pavan.garikapati@patni.com

ABSTRACT

This paper proposes an innovative application of Theory of Inventive Problem Solving (TRIZ) and Six Sigma for software process improvement. Use of contradiction matrix and 40 principles readily help in developing the solutions for the problems. This paper discusses out how to use TRIZ principles while carrying out six sigma projects using 3 cases where TRIZ application has directed the process improvement teams towards solutions.

Keywords: TRIZ, Software process improvement, Six Sigma

INTRODUCTION:

Six Sigma[1] is being used in software process improvement for quite some time now. Based on the rigorous methodology and statistical techniques, Six Sigma helps us in identifying the root causes for variation and low performance in the process. By eliminating these root causes we can realize dramatic improvements. However, many a times to eliminate this root causes one needs creativity in the Improve phase for developing solutions. Here, many creative tools like lateral thinking techniques are used to develop solutions. However, these tools while giving results, limit the usage due to their inability to directly drive towards solutions.

One of the simple, powerful yet easy tools is TRIZ for systematic process innovation. TRIZ directs towards solution and once the principles are identified the solutions are almost apparent. Also, the process can be improved using the solutions developed and can be refined until an optimum solution is obtained.

WHAT IS SIX SIGMA:

Six Sigma is a compelling method for breakthrough improvements for delivering world-class processes with a defect rate of less than 3.4 ppm (parts per million). Typically Six Sigma follows a five–step methodology known as DMAIC (Define, Measure, Analyze, Improve and Control).

Define phase focuses on developing business cases, defining the problems and the scope. It also helps understand the customer requirements to identify the process that delivers the same.

Measure phase measures the problem in quantifiable terms, which helps estimate the process capability that meets the needs of the customer.
Analyze phase identifies different causes for the failure of delivering the customer requirements. A variety of statistical and graphical techniques are used for prioritizing

different causes and identifying the root causes having high impact on customer requirements.

Improve phase is a creative phase, where one is aware of the problem and the cause. The cause needs to be eliminated through developing the solutions. However this is not very simple to remove the cause. We need to think creatively to develop the solution. These solutions will be piloted and then with confirmed results will go ahead for organization wide implementation.

To avoid fall back of improvement actions, there is a need to create lock-in effect and develop advanced warning system. *Control* phase takes care of these requirements.

FRAMEWORK FOR USE OF CONTRADICTION MATRIX:

Contradiction matrix helps us by directing towards right solution, is a perfect fit in the Improve phase of the methodology. Following framework is developed to use contradiction Matrix.
Step 1: Identify the root cause using Six Sigma Methodology
Step 2: Establish Contradiction and select principles
Step 3: Apply principles to develop the solution

CASE EXAMPLES

Case 1: CODING PRODUCTIVITY IMPROVEMENT
This case illustrates how the coding productivity in software maintenance process was improved. Customer is concerned with coding productivity, as the customer wants to improve the pace of maintenance. Coding Productivity is about 2.1 Size unit/day which is required to be increased to 3-size unit/day to meet the customer requirements.

Stage 1: Use of Six Sigma Methodology to find out the Root cause
A Six Sigma project was taken up for improving the coding productivity. In the define phase the problem statement was developed and project scoped. Productivity figures were collected and it was found that Mean of productivity was 2.1 and Standard Deviation was as high as 1. Different tools and techniques like cause and effect diagram, sub process mapping were used to list down the causes impacting the productivity variation. Through the prioritization techniques like ANOVA, it was understood that the amount of code used from earlier programmes by the programmers was the high impact cause.

To detail the scenario, the change requests being given by the customer could have used some of the code already written. However, each programmer kept his or her own word files to re-use the code. Additionally, the programmer does not have any access to the code written by others. Hence, every programmer only used the code, which they have written earlier.

Step 2: Establish Contradiction and select principles

Once the root cause was identified, it was understood that increasing the percentage usage of code already written is required to improve the productivity. The system features contradicting are 39. Productivity and 38. Automation.

Since the aim was to maximize the productivity while automating it was assumed that if we increase automation then productivity would reduces, which was the contradiction. The principles to be applied were then selected such that productivity increases while automation increases.

The principles from Contradiction Matrix are:

5. Merging
12. Equipotentiality
35. Parameter changes
26. Copying

Step 3: Apply principles to develop the solution
After brainstorming different principles, the following principles have driven the solution:

Principle: Merging

What it says: Bring together (or merge) the identical or similar objects; assemble identical or similar parts to perform parallel operations.

What was done: All the word files from different team members were collated under a common knowledge base, which was made accessible to everybody, thus enabling parallel operation. .

Principle: Copying

What it says: Instead of an unavailable, expensive, fragile object, use simpler and inexpensive copies.

What was done: Reusable code was divided into different design elements, which could be reused.

Summary
Coding productivity improved from 2.1-size unit/day to 3.2 size unit/day. This case example illustrates the effectiveness of applying TRIZ in Software maintenance process.

Case 2: TIME REDUCTION IN MIGRATION

Migration project was all about upgrading the development technology to the latest available. Technology of current project under discussion is VC++ 6.0, to be upgraded to VC++7.1. Process of migration typically includes selecting a project and building a project in higher version of software. Then, the software poses a number of errors. Correcting these errors such that we get 0 errors/warning makes the code migrated in to higher version of the software successful.

Step 1: Use of Six Sigma Methodology to find out the Root cause
Six Sigma project is taken up for reducing the migration time. There are several projects in a code base, which needs to be migrated. In the analysis phase of Six Sigma the root cause identified was - A lengthy build process.

Step 2: Establish Contradiction and select principles
The contradiction is established as Migration Time increases as the build time increases. Migration time vs. lengthy Build.

Migration time is the efforts spent by team members. For the process, effort is metaphorical to weight to a technical system. Lengthy build process is metaphorical to Length of a technical system.

From this established contradiction using the contradiction matrix the principles are selected. <u>Contradiction:</u> 2. Weight of Stationary object increases when 4. length of stationary object increases.

The principles from Contradiction Matrix are numbers
10 Preliminary Action
1 Segmentation
29 Pneumatics and hydraulics
35 Parameter Changes

Step 3: Apply principles to develop the solution
After brainstorming different principles, the following principles were used to develop the solution:

Principle: Preliminary Action
What it says: Perform, before it is needed, Pre-arrange objects such that they can come into action from the most convenient place.
What was done: Analyzed the code-base and grouped the dependant projects together. This preliminary action helps in arranging for batch builds.

Principle: Segmentation
What it says: Divide an object into independent parts, Make an object easy to disassemble, Increase the degree of fragmentation or segmentation.

What was done: Divided an object into independent parts, each group of batch builds is now independent from each other. Earlier all projects were having dependencies.

Summary
Migration time decreased from 2 hours/KLOC to 0.4 Hours/KLOC (KLOC is Kilo Lines of Code, a measure for software size). The results were amazing as 5 times improvement was seen in the process.

Case 3: Customer Report Preparation
This case illustrates how the customer report preparation time was reduced drastically using TRIZ and Six Sigma in Independent Testing projects. In the independent testing projects, Test Management Software is used for testing process management. However the reports are prepared using MS excel software, the process is manual and takes almost 30 % of the team effort for documenting and report preparation.

Stage 1: Use of Six Sigma Methodology to find out the Root cause
A Six Sigma project was taken up to reduce this report preparation time in all independent software testing projects. It was observed that manual process of taking out data from Test Management Software and copying in excel is the root cause for lot of time.

Step 2: Establish Contradiction and select principles
The team decided to go for automating the process and took the help of contradiction matrix since the team wanted to maximize the productivity while automating. It was assumed that if increase in automation would result in a decrease in productivity. The principles selected to be applied were such that productivity increases while automation increases.

The principles from Contradiction Matrix are numbers 1. Merging 2. Equipotentiality 3. Parameter changes 4. Copying

Step 3: Apply principles to develop the solution
Team used different principles in the following manner

5. Merging
12. Equipotentiality
35. Parameter changes
26. Copying

Principle: Merging
What it says: Bring closer together (or merge) identical or similar objects, assemble identical or similar parts to perform parallel operations.
What was done: A number of manual steps like copying pasting, counting, verifying etc. being done sequentially was combined. Downloading a detailed report, which Test Management Software develops in MS Word format, enables this

Principle: Parameter changes

What it says: Change an object's physical state (e.g. to a gas, liquid, or solid.)
What was done: Number of challenges came to fore while trying to do automation. They were

- Test Management Software supports detailed Report Generation of Test Cases in Word Format and not in other applications
- Customization of the Report Generation is not supported in Test Management Software.
- Summarized Report cannot be directly generated from Test Management Software.

Using the parameter change principle, the team creatively generated a report from Test Management Software to MS Word (Test Management Software can generate MS Word file) and then from MS Word it was converted to Excel file (MS Word and MS Excel are compatible).

Summary
Report preparation time has been reduced from 30% of total time to 10% of total time. The results are telling and definitely Contradiction Matrix/40 Principles gives a direction towards solution.

CONCLUSION
This paper proposes a new way of applying TRIZ with Six Sigma to the software process improvement. However the classical TRIZ orientation is very much towards the technical systems. More research is required towards bringing out database of solutions and principles application in software process improvement. This will help application of TRIZ maturing for process improvement. Also, This paper demonstrates use of Six Sigma along with TRIZ for better results.

Acknowledgements: The author thanks all the project teams worked with him, viz: Alapati Nagalakhmi, Vinay Polepally, Girish deshpande, Mitesh Patel and Mynampati Pavan Kumar. Author also thanks Venkat Balan, Six Sigma Master Black Belt for his critical comments and useful suggestions on the draft of this paper.

REFERENCES:

1. **http://**software**.isixsigma.com for Six Sigma in Software development**
2. www.triz-journal.com **for Contradiction Matrix and 40 principles**

Author Biography: ***Pavan Kumar Garikapati*** is presently working as Six Sigma Black Belt in the Corporate Quality and Delivery Innovation department of Patni, India's sixth largest software services player.
Pavan has about 10 years of consultancy and training experience in manufacturing, process, service and IT industries with expertise in installing and implementing Quality management systems and continuously improving on them using Six Sigma, TQM and ISO 9000 QMS.
A Certified Six Sigma Black Belt and certified Lead auditor for ISO 9000 QMS, Pavan has given consultancy to over 20 organizations including various sectors of economy. He has also attended International seminar on 'TQM in Service sector' conducted at Bangkok as Indian representative sponsored by Asian Productivity Organisation, Japan.
In his present role as Six Sigma Black belt, he has successfully implemented number of Six Sigma Projects in Software processes like Software development and Software Testing and support services like Marketing, helpdesk etc. He is also a certified Software Quality Analyst (CSQA).
A prolific trainer, he has logged more than 800 person hours training in Quality management and related fields, he is certified for Direct trainers skills and design of training by Govt. of India. He has also authored technical and research papers in Quality management.
A Mechanical Engineering graduate with a Masters in Manufacturing Engineering and Post graduation in Industrial Engineering, he has authored technical and research papers in Quality management.