Applying Six-Sigma Concepts to the Software Engineering: Myths and Facts

Conference Paper · February 2008

CITATIONS
5

READS
5,136

2 authors:

Rafa E. Al-Qutaish
64 PUBLICATIONS **576** CITATIONS
SEE PROFILE

Khalid T. Al-Sarayreh
Hashemite University
76 PUBLICATIONS **337** CITATIONS
SEE PROFILE

Applying Six-Sigma Concepts to the Software Engineering: Myths and Facts

RAFA E. AL-QUTAISH AND KHALID T. AL-SARAYREH
Faculty of Information Technology
Applied Science University
Amman 11931
JORDAN
rafa@asu.edu.jo, khalid_sar@asu.edu.jo
http://www.rafa-elayyan.net

Abstract: - Six-sigma has been invented by Motorola Company in 1986. Since that time, it has been widely used in the manufacturing. The success stories of applying six-sigma in manufacturing during the last years have encouraged the practitioners and researchers to explore the applicability of six-sigma in software industry. For the applicability of six-sigma in software engineering, some researchers came up with good results, while the others came up with a number of uncertainties on applying six-sigma to software engineering. In this paper, we will discuss these two opinions to present a clear view of both of them.

Key-Words: - Six-Sigma, DMAIC, DMADV, IDOV, Software Engineering, Software Process & Product.

1 Introduction

Six-sigma has been invented by Motorola Company in 1986 and defined as "a metric for measuring defects and improving quality" [1]. The term six-sigma has its origins in statistical process control, and recently it has become better known as the name of a wide ranging set of data based process improvement techniques [2]. In more details, six-sigma (6-σ) means six standard deviations. A standard deviation is a parameter that characterizes a set of measurements, just as the average can characterize such a set. One standard deviation is a value such that roughly two thirds of all values in a set fall within the range from one standard deviation below average to one standard deviation above average. Sets of values that can be characterized by the average and standard deviation can be modeled by the normal distribution, also known as the "bell-shaped curve". With a larger coefficient for sigma (1-σ, 2-σ, 3-σ, 4-σ, 5-σ, or 6-σ) more of the set is included, corresponding to a larger area under the bell curve [3]. Moreover, six-sigma is about measuring defects in value chain process in order to systematically reduce them and therefore the corresponding cost factors; where a process that executes on 6σ level will yield results within the tolerance interval with 99.99966% probability; which means that we could have only 3.4 defects per 1 million defect opportunities (DPMO) [4].

In the literature, there are many definitions of six-sigma. For examples, Brue [5] defined it as "a method for improving profitability and productivity; and it is a disciplined application of statistical problem solving tool to identify and quantify waste and indicate steps for improvement". On another words, six-sigma focuses on executive sponsorship, driving out defects, and measurable improvements [6]. While, Hefner and Sturgeon [7] defined it as "a management philosophy based on meeting business objectives by striving for perfection; and it is a disciplined data-driven methodology for decision making and process improvement". In addition, they identified the contents of the six-sigma to be five integrated methods, that is:

1- Process management.
2- Voice of the customer.
3- Change management,
4- Tools for measuring variation and change.
5- Business metrics.

Kumar [8] defined six-sigma as "a compelling method for breakthrough improvements for delivering world class processes with a defect rate of less than 3.2 parts per million". Whereas, Schofield [9] defined it as a disciplined, structured and data-driven methodology to solving problems. However, Motorola [10] identified six reasons why leaders love six-sigma, that is:

1- It impacts the bottom line:
2- It drives strategy execution.
3- It generates robust and flexible business processes.
4- It improves human performance across the enterprise.
5- It is highly scalable.
6- It is a low risk investment.

Since eighteens, six-sigma has been used to improve the process of producing the manufacturing products. Nowadays, there are a number of attempts to use the six-sigma in software engineering. In this paper, we will discuss these attempts along with the opposite opinions of some researchers who came up with a number of uncertainties on applying six-sigma to software engineering.

The rest of this paper is organized as follow: section 2 gives a general view of the six-sigma frameworks, section 3 discusses the use of six-sigma in software engineering, and section 4 introduces the uncertainty of the use of six-sigma in software engineering. Finally, section 5 concludes the paper.

2 Six-Sigma Frameworks

Six-sigma has different frameworks (methodologies) to be used within different circumstances (existing or new product/processes/service) [12], for example:

1- DMAIC: Define, Measure, Analyze, Improve and Control. It is used when a project's goal can be accomplished by improving an existing product, process, or service. It focuses on processes–including those supporting development –that persist from product to product.
2- DMADV: Define, Measure, Analyze, Design and Verify. It is used when the goal is the development of a new or radically redesigned product, process or service.
3- IDOV: Identify, Design, Optimize and Verify. It focuses on the requirements and plans for each new product–driving design improvements.
4- DCOV: Define, Characterize, Optimize and Verify. Ford uses this framework abbreviation for all categorize of its DFSS projects [13].
5- IIDOV: Invent, Innovate, Develop, Optimize and Verify.
6- CDOV: Concept development, Design development, Optimization and Verification.
7- IDEAS: Identify, Design, Evaluate, Assure and Scale-up.

To improve an existing project, the DMAIC six-sigma problem-solving framework should be used. For new projects, the most popular six-sigma framework used is DMADV which referred to as a DFSS framework (methodology).

2.1 DMAIC

The six-sigma process uses the DMAIC (Define, Measure, Analyze, Improve and Control) methodology as a roadmap for problem solving, as well as for product, process and business systems improvement [3]. The DMAIC framework consists of the following phases:

1- Define: In this phase, the six-sigma project team identifies a project for improvement based on business objectives and the needs and requirements of the customers.
2- Measure: In the Measure phase, the team begins with the proper metrics. Critical measures that are necessary to evaluate the success of the project are identified and determined. The initial capability and stability of the project is determined in order to establish a measurement baseline. Valid and reliable metrics to monitor the progress of the project are established during the Measure phase; input, process, and output indicators are identified.
3- Analyze: Through the Analyze phase, the team can determine the causes of the problem that needs improvement and how to eliminate the gap between existing performance and the desired level of performance.
4- Improve: The Improve phase is where the process transitions into solutions. Critical inputs have been verified and optimized toward nailing down the problem causes. Once problem causes are determined in the Analyze phase, the team finds, evaluates through testing, and selects creative new improvement solutions.
5- Control: Success in the Control phase depends upon how well the team did in the previous four phases. The keys are a solid monitoring plan with proper change management methods that identify key stakeholders. Lessons learned are now implemented and tools are put in place to ensure that the key variables remain within the acceptable ranges over time so that process improvement gains are maintained.

2.2 DMADV

It is used when the goal is the development of a new or radically redesigned product, process or service, and it's a DFSS framework. It consists of the following steps:

1- Define: This step spells out the project's purpose, importance, scope, deadlines, and resources.
2- Measure: The second step focuses on further defining and understanding the Voice of the Customer. This step integrates information from customer research, benchmarking, and technical research to arrive at a prioritized set of Critical-To-Quality (CTQ) Characteristics which the design must address.
3- Analyze: This step takes the CTQs from Step 2 and uses them to spark innovative ideas for a new design. These concepts are then evaluated and the

best concept is chosen for the next stage of design. Customer input is included in the evaluation.

4- Design: In this step several design sub-teams work in parallel developing, testing and building various design components. In general, the sub-teams develop and test a high level design, then make necessary modifications before developing and testing detailed designs.

5- Verify/Validate: In this step the team prepares to transition the product/service/process to process owners for ongoing maintenance and control. Teams typically deliver a detailed implementation plan which incorporates learnings from the full-scale pilot, documented work processes and methods, and a process management plan.

2.3 IDOV

The IDOV framework (methodology) focuses on the requirements and plans for each new product-driving design improvements. It consists of the following steps:

1- Identify: this phase contains the following steps: Translate customer qualities to system CTQ's, Perform CTQ flow-down/allocation, Verify measurement systems, and Create/validate system transfer functions.

2- Design: this phase contains the following steps: Formulate system design, Roll-up system capability, Compare capability flow-down & flow-up, and Identify gaps & trade-off lower level.

3- Optimize: this phase contains the following steps: Find the critical few X's, Assign robust targets & tolerances, and Generate manufacturing and purchase manufacturing.

4- Validate: this phase contains the following steps: Confirm predictions in pilot builds, Mistake-proof the process, Develop production control plan, Document the design effort and results.

3 Six-Sigma in Software Engineering

Six-sigma concepts are increasingly entering software engineering literature and practice [14]. However, today software practitioners are exploring ways to apply six-sigma techniques to improve software and systems development [15]. The essence of six-sigma for software is to prevent software from producing defectives in spite of their defects rather than to build software without defects [14].

Since six-sigma is new the software and systems development domains, many organizations are working hard to implant it; however, common questions from organizations include [15]:

– How does six-sigma compare with other improvement approaches, and how does it fit with my organization's other software process improvement initiatives?
– What evidence is there that six-sigma is applicable to software and systems engineering?
– What will it take for me to implement six-sigma in my organization, and how do I get started?
– How do I train software engineers in six-sigma methods when six-sigma training is largely focused on manufacturing?

In software engineering, controls can be implemented to take advantages of the improvement zone between 3σ and 6σ process performance; this can be done by building critical customers metrics into software solutions, for example: response times, cycle times, transaction rates, access frequencies, and user defined thresholds; then, these metrics can make the applications self-correcting by enabling specific actions when process defects surface within the improvement zone; these actions don't always need sophisticated technical solutions to be beneficial [14]. Controls can be as simple as an email notifying support personnel of defects above the 3σ level or a periodic report-highlighting activity in the 3σ to 6σ zone [14].

Fehlmann [4] identified a six-sigma approach to software development (by software development he not means only writing new software, but also software integration, deployment, and maintenance); that is:

1- Define: Set the goal.
2- Measure: Define the metrics.
3- Analyze: Measure where you go.
4- Improve: Improve your processes while you go.
5- Control: Act immediately if going the wrong path.

In addition, he mentioned three principles which based on the experience of implementing six-sigma for software:

– Principle #1: Measure customer related metrics only (use Combinatory Metrics to cover all topics).
– Principle #2: Adjust to moving targets (your goals may need change; accept change and manage it accordingly).
– Principle #3: Enforce measurement (Do not enforce meeting targets).

Siviy and Forrester [16] conducted a research project to investigate the use of six-sigma to

accelerate the adoption of CMMI. Thus, they concluded the following:

- Six-sigma helps integrate multiple improvement approaches to create a seamless, single solution.
- Rollouts of process improvement by six-sigma adopters are mission-focused as well as flexible and adaptive to changing organizational and technical situations.
- Six-sigma is frequently used as a mechanism to help sustain (and sometimes improve) performance in the midst of reorganizations and organizational acquisitions.
- Six-sigma adopters have a high comfort level with a variety of measurement and analysis methods.
- Six-sigma can accelerate the transition of CMMI:
  - Moving from CMMI ML 3 to 5 in 9 months, or from SW-CMM Level 1 to Level 5 in 3 years (the typical move taking 12-18 months per level)
  - Underlying reasons are strategic and tactical
- When six-sigma is used in an enabling, accelerating, or integrating capacity for improvement technologies, adopters report quantitative performance benefits, using measures they know are meaningful for their organizations and clients.

Moreover, Hefner and Sturgeon [17] compared CMM/CMMI to six-sigma, Table 1 shows the implementation differences between both of them.

Table 1: The Implementation Differences between Six-Sigma and CMM/CMMI [17]

	For an individual process	For the organizational infrastructure
Six-Sigma	Identifies how the activities might be improved.	Identifies what activities are used for improvement (DMAIC, DMADV).
CMM/CMMI	Identifies what activities are expected in the process.	Identifies how those activities might be implemented.

VanHilst *et al.* [18] proposed that the Global software Development Environments (GDEs) can be extended with a DMAIC framework (methodology) to interactively provide required metrics and analyses.

In the course of its work with hundreds of six-sigma companies and through its own experience, Microsoft has identified six key steps to ensure six-sigma success [19]; that is:

1- Establish Leadership Support and Engagement.
2- Align Goals with six-sigma Activities.
3- Establish six-sigma Infrastructures.
4- Identify Opportunities to Improve.
5- Match People with Projects.
6- Ensure Execution and Accountability.

Raytheon [20] started its six-sigma (also called R6S) in early 1999, as a result of their process improvement, they got $1.8 billion in gross financial benefits, $500 million improvement in operating profit, and generated $865 million in cash flow. This is an example of what others got by implementing six-sigma to software process improvement.

On contrast, Binder [21] stated that six-sigma does not make sense for software based on the following reasons:

1- Processes: Software processes are fuzzy. Every part of the software is produced by a process that defies the predictable mechanization assumed for physical parts.
2- Characteristics: Software characteristics of merit cannot be expressed accurately as ordinal tolerances; that which makes software correct or incorrect cannot usually be measured as simple distances, weights, or other physical units.
3- Uniqueness: Software is not mass produced. Even if software components could be designed to ordinal tolerances, they would still be one-off artifacts.

Head [22] used the cleanroom software engineering technique as a methodology to implement six-sigma to software in order to produce six-sigma quality software.

4 Uncertainty of Applying Six-Sigma to Software Engineering

The success stories of applying six-sigma in manufacturing during the last years have encouraged the practitioners and researchers to explore the applicability of six-sigma in software industry. Based on the results of this exploration, the researchers have been divided into two groups, that is, with or against this idea. However, some researchers came up with a number of uncertainties on applying six-sigma to software. Throughout this section, we will discuss in some details these uncertainties.

Binder [21, 23] stated that six-sigma does not make sense for software based on the following reasons:

1- Software processes are fuzzy. Every part of the software is produced by a process that defies the predictable mechanization assumed for physical parts.
2- Software characteristics of merit cannot be expressed accurately as ordinal tolerances; that which makes software correct or incorrect cannot usually be measured as simple distances, weights, or other physical units.
3- Software is not mass produced. Even if software components could be designed to ordinal tolerances, they would still be one-off artifacts.

Six-sigma software process could be interpreted as 3.4 failures per million lines of code; that is, 0.0034 failures per thousand lines of code, this would require a software process which is twice better than the current best practices [23]. It is difficult to imagine how this could be achieved since the average cost of such low rate of failures in the source code is reported to be $1000 per line [24].

Jacowski [25] mentioned that the big question is whether six-sigma can in fact be applied in the software industry as successfully as it was applied to manufacturing. However, this is still being argued. The real challenge is to see if it can be implemented to the software process without reinventing the wheel. There is also disagreement among leaders in the software industry about the need for six-sigma.

In Addition, a software development process - which is defined as a set of software engineering activities to transform user requirements into software product – is completely different than other types of processes such as manufacturing. The distinctiveness attributes for a software development process which are not available for other types of processes are as follows [26]:

1- Unlike other types of processes such as manufacturing, the software development process is not repetitiveness. However, each software product needs its own process. In addition, the software process produces one software product which could be duplicated with high precision. Then, once the software is duplicated, it produces the exact functionality as the original copy.
2- The inputs and outputs of the software development process are different in each instance of the process. It does not make sense to produce exactly the same piece of software twice. Therefore each instance of software process deals with one different set of user requirements, and outputs of different software modules form part of the final software product.
3- In contrast to a manufacturing process, each transformation of user requirement to a software module is cognition intensive; while, most of the manufacturing activities are targeted to minimize cognition.
4- As a result, software development is an intellectual process that needs visualization (e.g., documentation) before six-sigma implementation (Card, 2000). In software development, data relationships can be discovered via documentations, interviews and process mappings. Data flow diagrams, entity relationship diagrams, and object models are tools commonly used to represent the data relationship that the six-sigma approach needs for problem definition.
5- Software development is an intellectual process that needs visualization (e.g., documentation) before six-sigma implementation
6- Different sets of external factors affect the software development process, such as changes of developers, knowledge level, programming skills, and so on. Unlike the manufacturing process which is affected by many sources of variation such as temperature, raw materials, equipment wear, and human interaction. However, these factors are hardly valid for software process.

Moreover, Hong and Goh [26] stated that a misuse of six-sigma at the other extreme would be emphasizing the unique features of software process and never attempt to manage and improve the existing process. Some software engineering activities, such as the process life cycle models and quality measurements, are evidence of efforts towards producing stable software processes

5 Conclusion

Six-sigma has been invented by Motorola Company in 1986. Since that time, it has been used widely in the manufacturing. The success stories of applying six-sigma in manufacturing during the last years have encouraged the practitioners and researchers to explore the applicability of six-sigma in software industry. For the applicability of six-sigma in software engineering, some researchers came up with amazing results, while the others came up with a number of uncertainties on applying six-sigma to software engineering. In this paper, we have discussed these two opinions to present a clear view of both of them.

From our point of view, the six-sigma concepts could be applied to the software engineering, but it needs some customization. In addition, applying six-

sigma to software engineering process could be extended to the software product through the transformation of the product quality characteristics values into sigma values for all types of software product. Taking into account that different software product types may have different quality requirements since some software products are very sensitive to the quality, such as control systems, real-time systems, etc.

References:

[1] Motorola, "Six Sigma Dictionary," *http://www.motorola.com/content.jsp?globalObjectId=3074-5804*, Link tested on May 8th, 2006.

[2] Shelley, C. C., "Six Sigma and its Application to Software Development," *Oxford Software Engineering*, August 3rd, 2003.

[3] Breyfogle, F. W., *Implementing Six Sigma: Smarter Solution Using Statistical Methods*, 2nd ed, John Wiley & Sons Inc., New Jersey, USA, 2003.

[4] Fehlmann, T. M., "Six Sigma for Software," in *Proceedings of the 1st Software Measurement European Forum (SMEF'04)*, Rome, Italy, 2004.

[5] Brue, G., *Six Sigma for Managers*, McGraw-Hill, New York, USA, 2005.

[6] Pickerill, J., "Implementing CMMI in a Six Sigma World," Presented at *the 17th Software Engineering Process Group Conference (SEPG'05)*, Seattle, USA, March 7th-10th, 2005.

[7] Hefner, R. and Sturgeon, M., "Optimize Your Solution: Integrating Six Sigma and CMM/CMMI-Based Process Improvement," Presented at *the Software Technology Conference*, April 29th - May 2nd, 2002.

[8] Kumar, G. P., "Software Process Improvement - TRIZ and Six Sigma (Using Contradiction Matrix and 40 Principles)," *TRIZ Journal*, April, 2005.

[9] Schofield, J., "When Did Six Sigma Stop Being a Statistical Measure?" *CrossTalk: The Journal of defense Software Engineering*, April, 2006.

[10] Motorola, "six reasons why leaders love six-sigma," *https://mu.motorola.com/six_sigma_lessons/contemplate/assembler.asp?page=leaders_title*, Link tested on May 8th, 2006.

[11] Mazur, G. H., "QFD in support of design for six sigma," in *Proceedings of 8th International Conference on ISO and TQM*, Montreal, Canada, April, 2003.

[12] Brue, G., *Design for Six Sigma*, McGraw-Hill, New York, USA, 2003.

[13] Soderborg, N. R., "Design for Six Sigma at Ford," *Six Sigma Forum Magazine*, November, 2004, pp. 15-22.

[14] Biehl, R. E., "Six Sigma for Software," *IEEE Software*, Vol. 21, No. 2, 2004, pp. 68-70.

[15] Heinz, L., "Using Six Sigma in Software Development," *News@SEI*, Number 1, 2004.

[16] Siviy, J. M. and Forrester, E. C., "Using Six Sigma to Accelerate the Adoption of CMMI for Optimal Results," Presented at *the Six Sigma for Software Development Conference*, October, 2004.

[17] Hefner, R. and Sturgeon, M., "Optimize Your Solution: Integrating Six Sigma and CMM/CMMI-Based Process Improvement," Presented at *the 14th Annual Software Technology Conference*, Salt Lake City, USA, April 29th – May 2nd, 2002.

[18] VanHilst, M., Garg, P. K., and Lo, C., "Repository Mining and Six Sigma for Process Improvement," *ACM SIGSOFT Software Engineering Notes*, Vol. 30, No. 4, July, 2005, pp. 1-4.

[19] Microsoft, "Six Sigma: High Quality Can Lower Costs and Raise Customer Satisfaction," *http://www.microsoft.com/office/business/articles/sixsigma.mspx*, Link tested on May 8th, 2006.

[20] Hayes, B., "Introduction to Six Sigma for Software: The Third Wave," Presented at *the Six Sigma for Software Development Conference*, October, 2003.

[21] Binder, R. V., "Can a Manufacturing Quality Model Work for Software?," *IEEE Software*, Vol. 14, No. 5, 1997, pp. 101-105.

[22] Head, G. E., "Six-Sigma Software Using Cleanroom Software Engineering Techniques," *Hewlett-Packard Journal*, Vol. 45, No. 3, June, 1994, pp. 40-50.

[23] Binder, R. V., "Six Sigma: Hardware Si, Software No!, *http://www.rbsc.com/pages/sixsig.html*, Link tested on June 21st, 2006.

[24] Joyce, E., "Is Error-Free Software Possible?," Datamation, Feb 18, 1989.

[25] Jacowski, T., "Six Sigma in The Software Industry," *Ezine Articles, http://ezinearticles.com/?Six-Sigma-In-The-Software-Industry&id=198268*, Link tested on June 21st, 2006.

[26] Hong, G. Y. and Goh, T. N., "Six Sigma in Software Quality," *TQM Magazine*, Vol. 15, No. 6, 2003, pp. 364-373.