THE APPLICATIONS OF SCRUM AND SIX SIGMA TO THE SOFTWARE INDUSTRY

Kari Sanford

Fall 2017

TABLE OF CONTENTS

ABSTRACT

SCRUM is an Agile method developed to better manage engineering processes in the ever-changing, unpredictable software market (Donizetti Zorzo, 2013). “It embeds nicely into an organization’s existing release cycle, deriving from that process’ discovery and elaboration step an initial project backlog” (Hughes, 2012). SCRUM follows a five-step process: planning, development, design review, demonstrations, reflection. This structured process fosters creativity and productivity within a team, while using time and task management strategies to ensure the project is completed on time. SCRUM places special emphasis on communication in each step, whether it be internal of the company, or between the company and the customer. Communication of project requirements, progress, and completion allows the customer to have input during the entire development process, and an above-satisfactory end result.

Six Sigma is a set of management tools that improve business and engineering processes. Jerome Blakeslee Jr. describes Six Sigma as “a high-performance data-driven approach… to align an organization keenly to its marketplace and deliver real improvements (and dollars) to the bottom line” (Card, 2000). When applied to the software industry, Six Sigma impacts the “software process, the software product, and to balancing the ‘voice of the customer’ and the ‘voice of the business’ to maximise overall business value resulting from processes and products” (Mahanti, 2005). If the customer requirements, development schedule, and total costs are not clearly identified, the team will have great difficulty creating software that meets expectations. Practicing Six Sigma methods will help engineers develop in ways that decrease software defects.

SCRUM: THE CONCEPT

SCRUM is a widely-used, streamlined approach for product development in the software industry. It uses sprints and daily record-keeping to monitor tasks being performed. As the team develops the software, the stakeholders [or customer] might request a status update. The team can deliver an update to the customer via the user demo step in the SCRUM process. The demonstration is at the end of each build or development phase. All stories, or tasks, are recorded on the cumulative project backlog. These stories are marked as accepted, rejected, under development, or waiting. Sharing the backlog with the customer is another way to communicate project progress. Daily updates are recorded on the task board and burndown charts. These tools organize task information by user stories, story points, and task cards. The task board displays progress via task cards. The burndown chart illuminates progress of all the stories on the backlog. The team might displays the task board and burndown charts in a room that the customers have access to, which serves as a method of visual communication between the business and the stakeholders (Rodriguez *et al.*, 2013). Project management and communication are built each step of the scrum process, aligning teams for success.

SCRUM: TOOLS AND TECHNIQUES

User Stories

A user story is an entry to the product backlog. Pertinent data fields of a user story can include but are not limited to: an ID number, user type, action, reason, notes, priority, time estimate, and sprint number. The ID number is used to identify the user story. The user type, action, reason, and notes identify what task is being requested, by/for whom, and why it needs to be done. Based on this information, the team can determine the priority of the task, or story. In turn, the priority of the task influences which sprint the story is schedule to be completed in

(Rodriguez *et al.*, 2013). Depending on the project process, higher priority tasks may be scheduled in the first few sprints.

Story Points

In the Virtual SCRUM simulation at UNICEF University, story points represented a working day consisting of six hours of uninterrupted work (Rodriguez *et al.*, 2013). Story points are estimates of time to completion for a user story [or task].

Product Backlog

The product backlog is a spreadsheet that stores the backlog items. Each row corresponds to a user story. The spreadsheet is a historic document that serves as an archive for completed stories, as well as a place to record current and future stories. The product backlog is shared and edited by the Scrum Team, and can be easily understood by the product owner and other stakeholders (Rodriguez *et al.*, 2013).

Task Card

A task card contains the task name, user story details, story points assigned, and person responsible (Rodriguez *et al.*, 2013). The team may also add notes to each task card, for the sake of keeping information in one place. Notes pertinent to the project might include issues related to a task, lessons learned, and secondary contact information.

Task Board

User story requirements can be traced on task boards, which illustrate team progress and performance metrics. “As the development of the tasks evolves, the tasks move along the DOING list on the Task Board and the team members are able to load hours worked into the assigned tasks” (Rodriguez *et al.*, 2013). After a task is completed, the task is moved to the DONE list. If a team member is idle, they can selected a new task from the TO DO section. The task board is a tool that visually sorts tasks to show the team what they have accomplished, which can motivate them to continue to accomplish tasks. It embodies the agile values: transparency, collaboration, focus, and self-organization (Rodriguez *et al.*, 2013).

Burndown Chart

A burndown chart graphs effort against the elapsed time of a sprint. Estimated story points are ideal effort, and burnt story points are team effort (Rodriguez *et al.*, 2013). The team can refer to this chart to project when future user stories in the sprint backlog should be completed. A project manager could use the burndown chart to monitor the team’s progress, and identify when the team’s effort falls below the ideal effort.

SCRUM: ACCEPTED BY THE SOFTWARE INDUSTRY

Saddington writes that Agile Teams are more effective if they base their communication standards on Daily SCRUM. “This is a daily morning meeting that facilitates team communication, fosters teamwork, discloses details of the project work, and highlights items for review, action, or execution” (Saddington, 2012). All members of the Agile Team must participate, since they share their accomplished and current work, as well as obstacles they are

facing. There is no hierarchy in this meeting; everyone has the same level of power. This Daily SCRUM is a short meeting, between 15 and 20 minutes. All detailed discussions typically happen outside this meeting, where team members can solicit input from subject matter experts.

SCRUM Implemented at Amazon.com

Alan Atlas was the first full-time Agile trainer and coach at Amazon.com. He connected the how, what and why of SCRUM to the Amazon business plan. He shares that Agile practices were first introduced to Amazon in 1999, but the adoption of SCRUM did not happen until 2004. "Amazon.com's unplanned, decentralized SCRUM transformation is of interest because it is different from the current orthodoxy regarding enterprise SCRUM transitions, and its strengths and weaknesses reveal some fundamental lessons that can be applied to other enterprise SCRUM transitions" (Atlas, 2009). Elements that aided in the full transition to SCRUM methods were permission, teams, knowledge, and energy.

Teams at Amazon are given the power to resolve problems without waiting for outside help, or upper management approval (within reason). At Amazon, the culture focuses on "creating, delivering, and operating excellent software in as streamlined and process-light a way as possible" (Atlas, 2009). All teams [including local management] have permission to choose to implement SCRUM; it is not a requirement.

Atlas observed that the corporate culture at Amazon.com has consistently used Agile practices. Amazon.com practices the 2 Pizza Team concept: "If you can't feed a team with two pizzas, it's too large" (Jeff Bezos). This limits team sizes to five to seven people, allowing for smaller, specialized teams who complete projects in chunks. These 2 Pizza Teams are stable and

long-lived; similar to SCRUM teams, but without the daily communication and documentation of stories and tasks.

Some team leaders were hesitant to choose SCRUM because they did not know enough about the method. As a coach, Atlas spent time teaching leaders and their teams about the SCRUM process, and how to customize the sprint setup and documentation to the Amazon.com standards. After these workshops, leaders felt more confident in their choice to adopt SCRUM methods.

As Atlas spent more time with Amazon.com, employees formed an email-based SCRUM community. The employees trained each other to become SCRUM Masters, on an *ad hoc* and voluntary basis. Eventually, enough teams had adopted SCRUM and were continuously soliciting guidance. Atlas boarded Amazon.com as a full-time Trainer and Coach and assisted with smoothing out the SCRUM implementation process for teams.

Higher level engagement with SCRUM was absent at the organization and enterprise levels (Atlas, 2009). Executive leaders did not show support for SCRUM, so the transition to this new method exists only at lower team levels. While transactional leadership aids the adoption and standardization of early methods, it appeared that the teams were doing just fine making independent, informed decisions to solve their problems. Atlas recognizes this as "a pull-based incremental approach and not a plan-driven, command and control, push-based approach" (Atlas, 2009).

Atlas has sparse reports to demonstrate that Amazon.com benefited from SCRUM methods. "The Amazon S3 project, which not only delivered on time after 15 months of work, but nearly achieved the unusual result of having the entire development team take a week's vacation leading up to the launch day" (Atlas, 2009). Typically, teams that work on projects of

this scope and complexity will wait until the last minute to put in maximum effort, only to fail to produce the deliverables. Another team, infamous for failing to deliver any software after 8 months, created software a month later at the end of their first sprint (Atlas, 2009). If more projects were documented, and these reports better archived in a database, perhaps there would be more examples of SCRUM success at Amazon.com.

SCRUM Implemented at Microsoft

David Treadwell, the corporate vice president of the .Net Developer Platform group at Microsoft announced, "we're not mandating them [SCRUM], but we're encouraging them. So SCRUM is one process—the idea that teams meet once a day for half an hour, figure out what they're going to do then go off and do their work very quickly" in 2005. He mentions that two brains are better than one, which is why many programmers work in a minimum of pairs now at Microsoft. Treadwell admits that the software practices used in the mid-90s do not scale to today's problems [referencing the SQL Server 2005 project]. "As the projects got larger and larger, we introduced too many complex interdependencies on early software, more than we could really digest through the system" (Treadwell, 2005). These complexities can now be divided into smaller, simpler sprints, which are reduced to user stories and tasks. At Microsoft, SCRUM is used for developing applications, but also for testing applications for quality assurance.

Rob Caron, a developer at Microsoft, expressed worry that people were jumping on the bandwagon of adopting SCRUM. He identified success factors for his team implementing SCRUM, two of the most significance being training and management support. Most people on Caron's team have a copy of *Agile Project Management with Scrum*. Caron also ensures that

each team member has an opportunity to be the ScrumMaster role, to better understand the position and the process. He believes “the better the team understands SCRUM, the better your chances of success” (Caron, 2005). The other success factor to Caron’s SCRUM implementation is that his manager supports the new process. His manager holds the team accountable to keeping the backlog updated with the most recent statuses for each item. The team is also expected to solicit help or additional resources when needed. Caron closes with the comments “it’s very important on day one to communicate to management why you are implementing SCRUM, how they are involved, and what they should expect. Once management buys into SCRUM, I think you have a much better chance of succeeding” (Caron, 2005).

SCRUM: REJECTED BY THE SOFTWARE INDUSTRY

SCRUM Rejected by Middle and Upper Management at Amazon.com

During implementation of SCRUM at Amazon.com, Atlas observed that the transition stalled at the team level “due to failure to engage either middle or upper management in a meaningful way” (Atlas, 2009). Both of these groups must provide their support and execution of the transition in order for changes to truly take effect. Training for middle managers is necessary, but it is unlikely that they would attend unless executive management does, too. A Transition Team, with a subject matter expert, could speed up this process by garnering support from the upper levels of management first. The executive team could pass the baton of newfound knowledge and a vision of change to the middle managers, who in turn can inspire the lower level team leaders.

SCRUM Rejected by Panda Strike

Panda Strike is a software company that has recently rejected Agile and SCRUM methods. Since the company's employees are spread out around the country and world, they disagree with the Agile Manifesto that advocates for face-to-face conversation. Giles Bowkett, one of the Panda Strike employees, writes, "Face-to-face communication has a lot of virtues, and there are certainly times when it's necessary, but it's not designed to facilitate extremely detailed changes in extremely large code bases, and tools which are designed for that purpose are often superior for the task." Bowkett's team primarily uses Github diffs to demonstrate the edits to the code. Team members can leave comments to ask or answer questions on the code. Each team member can even have their own branch, distinct to their individual contributions to the project. Bowkett believes the SCRUM meetings are simply recaps geared for a non-technical audience, which wastes valuable time that could be spent developing. His distributed team values open source, open feedback tools to create software.

From personal experience, Bowkett finds, "You won't be a productive developer if you're not fluent with modern tools. It doesn't matter how good you are with individuals and interactions if everybody's texting and you're still leaving voicemails." If the tools and process change, the interactions must follow suit. Bowkett observes a fluid boundary between tools and interactions; the tools are used to create more effective, meaningful communication interactions. The source code host is the process, and medium is the message.

Bowkett disagrees with the daily aspect of SCRUM, since some tasks or user stories may take a few days or even weeks to complete. Having a daily meeting with no update is pointless to him. Bowkett also disapproves of SCRUM terminology. He prefers to use *iteration* instead of *sprint*. Since SCRUM is an Agile Process, he would rather use Agile terminology. Bowkett also

finds contradictions between SCRUM and Agile language. “Agile processes promote sustainable development… [The team] should… maintain a constant pace indefinitely. Contrast this with the term ‘sprint,’ which suggests a short burst, an unsustainable pace” (Bowkett, 2015). Lastly, Bowkett has identified negative connotations with SCRUM language, too. A team could be starting a project on the first day, and already have a *backlog*. This implies that the project is behind schedule, and could be discouraging to employees. *Sprints* and *spikes* imply urgency, and could rush employees to complete tasks, potentially decreasing end quality.

SIX SIGMA: THE CONCEPT

Before Lean and Agile methods, engineers performed quality assurance by testing their software after it was completely developed (Hong, 2003). Often, defects found using this approach are resolved by reworking the entire software package, which can add months to the timeline. This is inefficient in time and cost; software companies could be using these resources on other products.

The quality of a software product is determined by how well the design and development processes operate. Six Sigma increases product quality by reducing variability, or inconsistencies (Biehl, 2004). The sigma number is a measure of process variation.

Process capability is defined as the full range of normal process variation for a product feature. Statistical calculations will use μ (mean) to represent the average of all values in a population and σ (sigma) to represent the standard deviation of the distance from the population’s mean value (Hong, 2003). The maximum range of variation will determine the design tolerance on the desired mean value. When the process capability value is greater than the

calculated upper and lower tolerance limits (S_U and S_L), defects or errors occur. The relationship of tolerance to process capability is the design margin, or capability index:

$$C_p = \frac{\text{Design Tolerance (or Specification Width)}}{\text{Process Capability (or Total Process Variation)}} = \frac{S_U - S_L}{6\sigma}$$

(Hong, 2003).

Sigma levels, or numbers, are metrics quantified by the process capability index, C_p. "In order to design the products and processes so defects virtually never occur, the specification width (or design tolerance) should be twice as large as the true process capability" (Hong, 2003). Per this requirement, an appropriate Six Sigma capability would be $C_p \geq 2.0$. Most organizations adopt the limit of 3 sigma, to control design processes with the goal of creating products that satisfy customers.

SIX SIGMA: MANAGEMENT TOOLS & TECHNIQUES

Prominent Six Sigma management tools and techniques that can be applied to the software industry are discussed below.

Check Sheet: This tool is used in the measure phase of Six Sigma methodology. The check sheet gathers "data of the desired characteristics of a process that should be improved" (Kumari, 2011). This tool is a method of control; management can use the data collected to identify areas of improvement, and then direct engineers to make the appropriate adjustments.

Pareto Chart: Should several quality problems arise at once, the Pareto chart assists management in prioritizing which problem should be solved first (Kumari, 2011). The Pareto chart is related to the 80/20 rule: "80% of problems are because of 20% of causes" (Kumari, 2011). This tool

appears in the define phase of Six Sigma, as it identifies one problem to solve at a given time. In the analysis phase of Six Sigma, the Pareto chart can be used to uncover the causes of each problem. This tool helps organization leaders perform all four functions of management more efficiently:

- Plan: identify a problem to solve.
- Organize: allocate resources, or people, to solve the problem.
- Direct: instruct workers assigned to the problem.
- Control: determine root causes of the problem.

Cause and Effect Diagram: This diagram lists all possible causes of an effect (problem). The Cause and Effect Diagram is also called the Fishbone Diagram because when written out, it resembles the skeleton of a fish. “The main problem is the head of a fish, the main causes are ribs and the detailed causes are the small bones” (Kumari, 2011). In Six Sigma, this diagram is used in the design and analyze phases. Management would benefit from this tool to determine causes of the problem (control), and to draft a resolution strategy (planning).

Control Chart: This tool is also known as Statistical Process Control. In Six Sigma, it is used in the analysis, improve, and control phases (Kumari, 2011). The control chart determines if a process has outcomes that are predictable or not. It can also identify a cause of process variation, and confirm that a process performance improved, or has smaller variation (Kumari, 2011). Management can use this tool to create a design process that has consistent variation according to Six Sigma standards (± 3 sigma). Management can also use the control chart to “detect the change in the process i.e. alteration in mean value” (Kumari, 2011).

Other Six Sigma management tools and techniques include, but are not limited to: histogram, stratification, scatter plot, project management methods, FEMA (Failure, Effect and Mode Analysis), stakeholders analysis, process documentation, ANOVA (Analysis of Variance), correlation and regression, and design of experiments (Kumari, 2011).

SIX SIGMA: ACCEPTED BY THE SOFTWARE INDUSTRY

Within the past two decades, software variability has increased due to user skills failing to keep up with updated software features, increased network loads and slower response times, and failure to store and retrieve data from databases (Biehl, 2004). Variability jumped from 3σ to 6σ. While new processes were performing at acceptable quality levels within this new variability scope, quality engineers noted, "The point isn't to build software without defects but to prevent software from producing defectives in spite of their defects" (Biehl, 2004).

Six Sigma helps software design processes, and management, improve in two key areas. It encourages management to automate correction processes by using systems that measure program performances (Biehl, 2004). Six Sigma also encourages the implementation of reaction procedures for when key metrics fall into the predetermined improvement zone (Biehl, 2004). Reaction procedures could vary from a predetermined adjustment in the program execution, to a notification sent to management, with the associated performance data attached to the message. An example of a reaction procedure would be "a data display application might temporarily reduce the amount of detail it displays per screen if it notes that data access or response times are rising above their control limits" (Biehl, 2004).

Mahanti presents results of a Six Sigma pilot survey of the Indian software industry in the 2009 paper, "Six Sigma in the Indian Software Industry." 100 questionnaires were sent to

software companies, with a response rate of 20 percent. It was discovered that only 12 companies were actively using the Six Sigma method. Those surveyed used the following metrics to measure service performance in the software industry: number of customer complaints, defect (bug) rate, cost of poor quality, defect per million opportunity, process capability index, and access time (Mahanti, 2009). All agreed that it is important to lower negative performance metrics. Of the master black belts, green belts, black belts, project managers, and others surveyed, it was found that Six Sigma will improve the development process by "reducing process deviation, centering, making the process mean match with the process target" (Mahanti, 2009). "The respondents were asked to rank the 19 Critical Success Factors [for implementation of Six Sigma in software industry] on a scale of 1 to 19 (1 being the most important and 19 being the least important" (Mahanti, 2009). Management commitment, linking Six Sigma to business strategies, and project planning were the top three ranked factors. Documentation management and suppliers involvement were identified as the least important factors (Mahanti, 2009). These results illustrate the importance of management endorsement. If senior management actively supports the implementation of Six Sigma methods, it is likely that the procedure will be used by all tiers of the organization.

SIX SIGMA: REJECTED BY THE SOFTWARE INDUSTRY

Some leaders in the software industry have outright rejected Six Sigma methodology, in preference for other modes such as the Capability Maturity Model. A primary reason for Six Sigma rejection is that management cannot visualize the software design process unless it is documented (Card, 2000). Six Sigma does not address the inherent need for software documentation. Another weakness of the Six Sigma method is that it "relies on trained personnel

to find processes that need improvement and act accordingly" (Card, 2000). If there is only one senior software engineer in the department that can identify bugs, it is likely that key software features will be overlooked, as this senior engineer will be overbooked with project tasks, and pressured to meet deadlines. In order for Six Sigma to be successful, there must be several people who are trained to identify problems in the code, and implement solutions. Furthermore, the Six Sigma method "focuses internally on learning from current experience. It does not consider, systematically, how externally developed technology might revolutionize or even eliminate a process" (Card, 2000). To survive in an ever-changing industry, software leaders must specialize in a particular technology and consistently meet customer demands, thus following the Capability Maturity Method.

Other reasons for Six Sigma rejection revolve around the calculations of variance. The design tolerance, or maximum range of variation, of software characteristics cannot be measured on a continuous numerical scale. "A software output either conforms or does not conform to the specifications" (Hong, 2003). Software testers will use binary notation when evaluating output: 1 for pass, and 0 for fail. With respect to the software industry, upper and lower boundaries of expected output do not exist. "Fault tolerance is often used to increase software reliability by putting in redundant modules" (Hong, 2003). However, the software design process hardly repeats itself. Each software test uses a different set of inputs to yield a unique output. "Since process capability represents the full range of normal process variations for a given characteristic," process capabilities are not defined clearly in the software process (Hong, 2003). If the values used to calculate variance using Six Sigma methods are not compatible with the performance values that the software industry creates, management will be less likely to use Six Sigma methods.

COMPARISON: SCRUM AND SIX SIGMA

SCRUM and Six Sigma use similar processes to manage long-term projects. Both methods have a planning phase, where the team identifies the problem, understands customer requirements, and drafts a scope statement. In SCRUM, planning occurs in its own sprint, a prequel to the work discussed and executed in the other sprints. Six Sigma planning occurs before any work is started, and usually requires approved documents signed by executive management and the customer. The next step is to organize the project by delegating resources and people to the project. In SCRUM, the Scrum Master typically organizes the project by setting up the documentation [backlog, task board]. In Six Sigma, the Project Manager requests funds from upper management, and determines which team will work on the project. Following organization is the direct phase, or execution. SCRUM uses sprints for execution, dividing the work into smaller pieces that are completed in about 30 days each. It is unique in that demonstrations and reflections are conducted at the end of each sprint. Demos exist to prove that the work is being completed to the customer's liking. Reflections are used to report lessons learned to management, and used for reference for similar work in the future. Six Sigma might implement these procedures, but they are not required in the process. Six Sigma directs specific team members to the work, which might be broken down into phases [such as time study, propose recommendations, analyze recommendations, implement recommendations]. Different than SCRUM, Six Sigma might use other Agile tools to manage the project schedule, such as Work Breakdown Structures and Gantt Charts. Both SCRUM and Six Sigma use meetings and documentation as methods of control, to ensure the project is completed according to previously identified cost, schedule and customer specifications.

CONCLUSION

SCRUM makes large complex projects more manageable by reducing them to smaller parts, or sprints. These sprints have tasks that are well-documented. The team decides who is responsible for which task at daily team meetings, and shares status updates. SCRUM places special emphasis of communication, not only at meetings, but as work is being performed. In-person collaboration is the key to success with SCRUM; teammates can bounce ideas off of each other, and learn new skillsets along the way.

The Six Sigma methods will only benefit a software organization and its management if the development process is documented. Six Sigma can improve the management and execution of project requirements, schedule, and cost by implementing procedure to reduce product performance variability. Automating performance data collection and creating reaction procedures for low-performing software components are two examples of Six Sigma implementation. However, standardizing and enforcing new procedures is only achieved by communication. You can write a new procedure, but if no one reads it or talks about it, it will never make a difference.

I believe that the best method for software development is a tailored version of SCRUM. SCRUM is a more viable option than Six Sigma because it integrates communication and other methods of reporting into its process. The planning phase involves the customer, to remove any ambiguity about product requirements. The development phase is less daunting in smaller pieces; it reduces employee stress levels and maintains high motivation to complete work. The design review serves as stage gate reviews for executive management, and the customer if they wish to attend. Demonstrations serve as another confirmation mode for management and the client that work is being completed. Demonstrations also provide public speaking opportunities for

employees seeking extra professional development. Reflection should always be required for engineering projects. It forces the team to assess their performance, identify areas of improvement, and record tips and tricks for future relevant projects.

WORKS CITED

Arul, Krishna and Harsh Kohli. 2004. "Six Sigma for Software Application of Hypothesis Tests to Software Data." *Software Quality Journal* 12 (1): 29-42.

Atlas, Alan. "Scrum at Amazon – Guest Post by Alan Atlas." *The Agile Executive.* Wordpress.com, 20 July 2009. Web. 26 November 2017 Accessed. https://theagileexecutive.com/2009/07/20/scrum-at-amazon-guest-post-by-alan-atlas/

Biehl, Richard E. 2004. "Six Sigma for Software." *IEEE Software* 21 (2): 68-70. doi:http://dx.doi.org.ezproxy1.lib.asu.edu/10.1109/MS.2004.1270765.

Bowkett, Giles. "Flaws in Scrum and Agile." *PandaStrike.* PandaStrike, LLC, 4 March 2015. Web. 23 November 2017 Accessed. https://www.pandastrike.com/posts/20150304-agile

Card, David N. 2000. "Sorting Out Six Sigma and the CMM." *IEEE Software* 17 (3): 11-13. doi:http://dx.doi.org.ezproxy1.lib.asu.edu/10.1109/MS.2000.10019.

Caron, Rob. "Critical Scrum Success Factors." *Microsoft | Developer.* Microsoft, 1 April 2005. Web. 22 November 2017 Accessed. https://blogs.msdn.microsoft.com/robcaron/2005/04/01/critical-scrum-success-factors/

Donizetti Zorzo, Sergio, et al. "Using Scrum to Teach Software Engineering: A Case Study." *Frontiers in Education Conference, 2013 IEEE*, 2013, pp. 455–461.

Hong, G. Y. and T. N. Goh. 2003. "Six Sigma in Software Quality." *The TQM Magazine* 15 (6): 364-373.

Hughes, Ralph. *Agile Data Warehousing Project Management Business Intelligence Systems*

Using Scrum. Burlington, Elsevier Science, 2012.

Kumari, Neeraj. 2011. "Applying Six Sigma in Software Companies for Process Improvement." *Review of Management* 1 (2): 21-33.

Mahanti, Rupa and Jiju Antony. 2005. "Confluence of Six Sigma, Simulation and Software Development." *Managerial Auditing Journal* 20 (7): 739-762.

Mahanti, Rupa and Jiju Antony. 2009. “Six Sigma in the Indian Software Industry: Some Observations and Results From a Pilot Survey.” *The TQM Journal* 21:549-564.

Rodriguez, Guillermo, et al. “Virtual Scrum: A Teaching Aid to Introduce Undergraduate Software Engineering Students to Scrum.” *Computer Applications in Engineering Education*, 2013, pp. Computer Applications in Engineering Education, 2013.

Saddington, Peter. *The Agile Pocket Guide A Quick Start to Making Your Business Agile Using Scrum and Beyond*. New York, Wiley, 2012.

Selby, Paige C., and Selby, Richard W. “4.4.2 Measurement-Driven Systems Engineering Using Six Sigma Techniques to Improve Software Defect Detection.” *INCOSE International Symposium* 17:640-651.

Taft, Darryl. Treadwell, David. “Microsoft Lauds Scrum Method for Software Projects.” *eWeek.* QuinStreet Inc, 11 November 2005. Web. 22 November 2017 Accessed. http://www.eweek.com/it-management/microsoft-lauds-scrum-method-for-software-projects