Agile and Six Sigma, how do they mix together?

MOHAMED AMR, Software Engineering Competence Center (SECC)
AYMAN KHALIFA, Diagnosoft

This experience report is about how agile software development and Six Sigma can be adopted together. It explores how both can benefit from each other and the gaps that are filled by mixing both of them together. We tried to cover these aspects through telling our story in adopting both together as well as pointing out what we have learned, how did we overcome the challenges we faced and eventually what we concluded after adopting both successfully.

1. INTRODUCTION AND BACKGROUND

Agile methods have been dominating the software development domain in the last decade, and they have proven to be successful for managing and executing software development projects, which are considered as innovative knowledge work as opposed to task work projects founds at other domains such as manufacturing.

Agile teams usually focus on quick and short-term improvements identified either through daily collaborations or through periodic vehicles like retrospectives. Sometimes agile teams lack the bird-eye/strategic approach to process improvement or problem solving. This is where Six Sigma comes along.

Six Sigma is a philosophy, a measure and a methodology for problem solving and process improvement. It provides a set of tools, phases and roadmaps, which can be used by an organization to reduce variation and improve the capability and quality of its processes, products and services.

In this experience report, we will share our experience in combining Six Sigma and its 'DMAIC' roadmap as a statistical and analytical problem solving approach with agile software development (particularly Scrum). We will discuss how we benefitted from applying the DMAIC roadmap with an agile team who was adopting Scrum. We will also discuss how we used agile methods in managing the Six Sigma project itself by adopting an agile-based iterative and incremental method called "Process Increments". In addition, we will try to bridge the gap between the 2 approaches and methodologies, and focus on the areas where both can meet and benefit from each other.

2. SIX SIGMA IN SOFTWARE DEVELOPMENT

2.1 What is Six Sigma?

Six Sigma is a structured methodology for process improvement and problem solving. It offers a rich set of tools, techniques and roadmaps, which aim to reducing variation, improving the quality of production processes by decreasing the number of defects and improving the capability of the processes, products and services.

Motorola developed Six Sigma in 1986. And since its adoption at General Electric in the 1990's, it has been widely adopted by many organizations in different domains and industries. Six Sigma gained popularity mainly in the manufacturing domain, mainly because of its origin and the fact that Six Sigma often related to an objective of reducing variation in production processes, and since reducing variation in a repetitive production process in manufacturing has been always a desirable goal to achieve, it has led Six Sigma to be a successful approach and methodology for improving processes, products quality and accordingly the business.

In parallel to this movement, the Lean movement was rising as well. Basically, the Lean philosophy in production considered the expenditure of resources for any goal other than the creation of value for the end customer to be wasteful, and thus a target for elimination [1]. Both Lean and Six Sigma were coupled eventually to produce what was called Lean Six Sigma, which is a systematic approach for process improvement and problem solving aiming to reduce variation, remove waste and increase capability.

First Author: Mohamed Amr Abdel Kader, Cairo-Egypt, email: mohamed.amr@gmail.com
Second Author: Ayman Khalifa, Cairo-Egypt; email: ayman@diagnosoft.com

2.2 Six Sigma in Software

A software development project is a typical knowledge work project, which involves a lot of innovation and creativity and relies heavily on the skills of individuals and teams. It also relies heavily on frequent and high bandwidth communication to continually manage uncertainty on the way to develop the final outcome, which almost in all cases, is intangible and not clear.

Adopting Six Sigma to produce software products or improve existing software development processes hasn't been that popular mainly because Six Sigma has always been understood as a structured methodology for reducing variation in a repetitive production process as opposed to a human-centric and innovation focused process such as software development. Also considering the value from the Agile Manifesto "*Individuals and Interactions Over Processes and Tools*", one would argue that you'll always find variation in a software development process because it heavily relies on the skills of individuals and teams which makes them the primary source of variation. Also variation from this perspective is not perceived as problem in the software development because different engagements are not perceived as repetitive.

However in order to assess the benefit of adopting Six Sigma to a software development team, Six Sigma should be looked at from a different perspective. A typical objective of using Six Sigma for software development teams would not be to reduce variation, but to remove waste and eliminate or reduce defects. Lean Six Sigma is actually a philosophy and a structured methodology for problem solving and process improvement, which provides a very rich and useful set of tools, roadmaps and techniques for this purpose. It depends on using the data along with the right set of tools and through asking the right set of questions to the right stakeholders to focus on the right problems. Also Six Sigma is considered as a pragmatic approach to empirical and experimental process improvement. It's simply about solving problems, by focusing on the right problems, which impacts the business and introduce costs to the organization. Six Sigma also, is about investigating problems to the level of their root causes and finding appropriate solutions rather than introducing a catalogue of pre-defined solutions.

2.3 Six Sigma and Agile software development

The 12th principle of the Agile Manifesto states the following: "*At regular intervals, the team reflects on how to become more effective, then tunes and adjusts its behavior accordingly*". This principle relates to the fact that most agile teams focus on continuous improvement and adaptation to change in the form of Inspect and Adapt cycles which happens either through daily collaboration (e.g. in daily standup meetings) or through periodic vehicles such as retrospective meetings happening at the end of an iteration. During these checkpoints, most agile teams would focus on soft observations and short-term improvement actions rather than adopt a bird-eye view or approach to problem solving and improvement. Also informed decision making at this stage sometimes is based only on soft observations not solid data and measures. And this is where Lean Six Sigma comes along.

For an agile team, Six Sigma would provide them with a structured approach for empirical problem solving. It would also provide them with a set of tools and techniques, which can be used to address the root causes of the problems impacting their performance, quality and customer satisfaction.

2.4 DMAIC roadmap for problem solving and process improvement

DMAIC is a data-driven roadmap for improving existing processes in a way, which removes waste, eliminate defects and increase performance and capability. Six Sigma projects adopt the DMAIC as their roadmap for improving an existing process or solving a problem.

The different phases of the DMAIC roadmap are [Define – Measure – Analyze – Improve – Control], as shown in figure 1.

***Define**: Define the problem, guided by the voice of the customer, business, or current process. Define the goal to achieve and formulate a business case for the Sigma Six project*

***Measure:** Measure the key parts of the existing process, and collect data about factors contributing to the problem in order to establish a current baseline for performance from a quantitative and qualitative perspective*

***Analyze:** Analyze the factors contributing to the problem in order to identify the root causes for the critical factors impacting performance and prioritize those root causes*

***Improve:** Identify, implement and evaluate improvement solutions to eliminate root causes of the problem in part or in whole*

***Control:** Sustain the improvements achieved and monitor them to ensure continued and sustainable success*

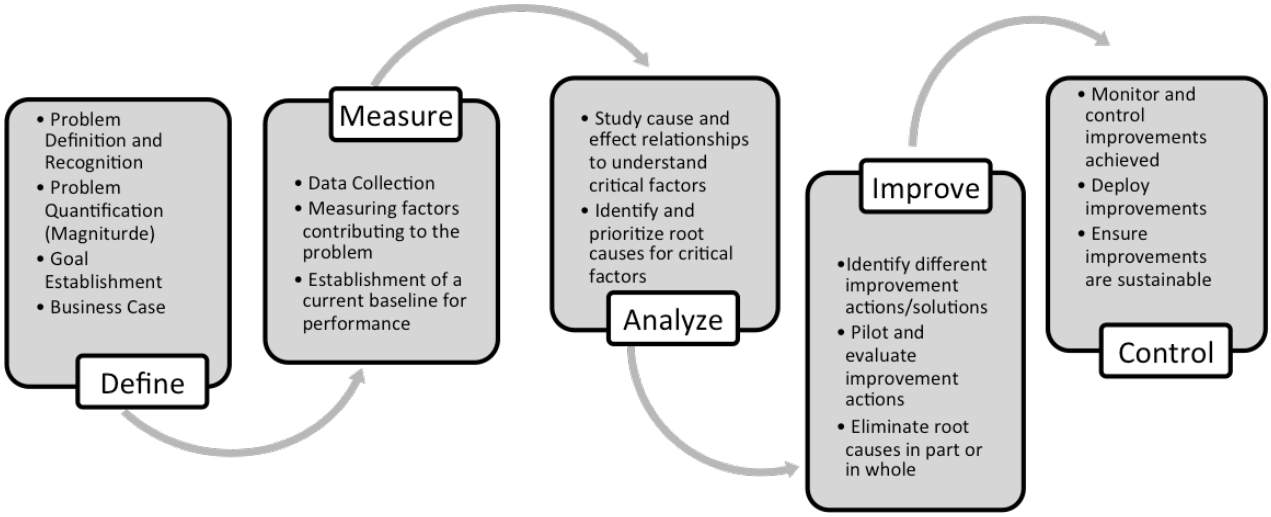

Fig. 1. A typical DMAIC cycle for a Six Sigma project

3. PROBLEM CONTEXT AND BACKGROUND

3.1 Project Background:

Our experience mainly involved an agile team developing a medical imaging tool designed for the viewing, quantification and analysis of cardiac magnetic resonance images (MRI). The first version of that product was launched in 2006, and since then there has been continuous development to the product to provide new features and enhancements as well as fixing issues and bugs. Due to the long period of development in addition to the complexity of the domain and technologies used, the code base was growing in size year after year eventually exceeding 400 thousand lines of code and ending up with legacy code in many modules.
In addition to the large legacy code base, the product had more than 1300 manual test cases, which constituted the full regression test suite.
The team had been adopting Scrum for about a year when we started our Six Sigma project together. The team size most of the time varied between 7-10 technical members.

3.2 Problem Context:

During the recent product releases, the team became more overwhelmed with new feature requests and changes/enhancements to existing features in addition to the technical issues they were facing in development and testing. This has led the team to overlook the need to work on the core problems affecting their quality, productivity and throughput and instead focus only on delivering new features to the business and providing business value as quickly as possible.
Ironically, those problems eventually impacted the business value offered to their customers by delaying their release dates, and impacting the business as well by affecting the time to market.

The trigger which made us think of adopting the DMAIC improvement roadmap to the team came after I had a meeting with the Product and Development Manager. I got from him that they were facing some problems in their recent product releases, which were costing them more money and impacting their customers and the time to market the new product releases.

Basically, the problem was; '*Accumulated leaked bugs as well as delayed regression and integration tests results in consuming a large stabilization phase at the end of each production release. This leads to delays in the release delivery as a result of the large stabilization phase, which impacts the product sales and clients. It also leads sometimes to adding additional sprints in the next release as a result of bugs being transferred from one release to another*.'
We realized that this problem was an excellent candidate problem for a Six Sigma improvement project following the DMAIC improvement roadmap.

3.3 Define phase:

The Define phase in a typical DMAIC project is mainly about defining the problem. Several factors and inputs contribute to the problem definition. These inputs are collected as:

(1) Voice of the Customer (VOC): Represents the input from the perspective of the Customer. It refers to criteria impacting the customer satisfaction like defects, cost, expectation, value delivery…etc.

(2) Voice of the Process (VOP): Represents the input coming from the process targeted to improvement. It refers to understanding the process activities, inputs, outputs, roles and responsibilities involved and the boundaries.

In our project the voice of the customer (VOC) was obtained from the perspective of the clients awaiting new versions and product releases. The value to these clients was impacted mainly by the delay in releasing the new versions (as a result of large stabilization phases). The problem was also impacting the product sales and delaying the revenues.

As for the voice of the process (VOP), it basically represented the approach the team was adopting at this point for product development, which was basically Scrum. It also referred to the problems and issues they were facing related to the nature of their product and technologies used.

The next step was to quantify our problem statement. We needed to know the magnitude of the problem we were facing at this point. We started to ask the following questions regarding the problem statement identified above:

- What's the % of the time allocated to the stabilization sprints/activities vs. the total duration of the product release?
- What are the figures about the numbers of leaked bugs accumulated at the end of each product release? (Before entering the stabilization phase)

The quantified problem definition became:

"Accumulated leaked bugs (reached an average of 100-140 bugs before entering the stabilization phase) as well as delayed regression and integration tests, results in consuming a large stabilization phase at the end of each production release. This leads to delays in the release delivery as a result of the large stabilization phase, which impacts the product sales and clients. The % of time allocated for the stabilization activities reached more than 35% of the total time of the product release"

Usually the problem in a DMAIC Six Sigma project is referred to statistically as the response (Y) which is a function of X's which are the factors contributing to this (Y). Usually we start discussing the X's during the Define phase, but only during the Analyze phase we are able to find the critical X's and the root causes behind them.

The next step was to gather input from the product development team as well as the product owner who represented the value team. We had a brainstorming meeting, which resulted in a fishbone diagram shown as figure 2. It included the main factors that the team thought was impacting the main problem (Y).

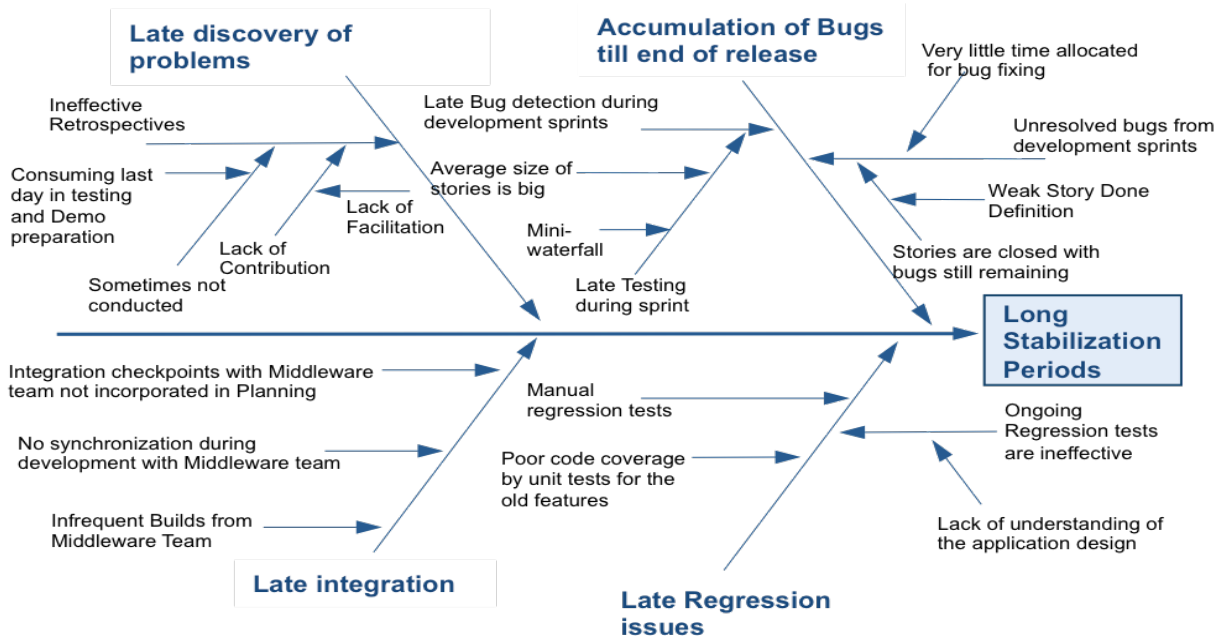

Fig. 2. The Fishbone diagram, which was formulated during the Define phase.

Considering the factors identified in the fishbone diagram, which captured the current issues the team had at that time, we decided to come up with the following goals for our Six Sigma initiative.

- Decrease the time allocated for Stabilization activities from 35% to somewhere between 15% and 20% (only 1 stabilization sprint every 6 development sprints) in order to release faster
- Decrease the number of leaked bugs which consumes a lot of time at the end of each product release
- Produce a potentially shippable product increment at the end of each sprint

After defining the problem and establishing the goal for the Six Sigma project, we formulated our business case. The business case was basically a cost-benefit analysis comparing the cost to implement the project (at this stage the cost mainly was represented by the effort of the team involved in the Six Sigma project) against the benefits achieved (Saving the effort wasted in stabilization activities + better product quality and faster time to market).

3.4 Measure phase:

During the Measure phase, it was the time to start taking a closer look at the factors (X's) impacting our main problem. Our main input was the fishbone diagram (Figure 2) created during the Define phase.
The first thing to do was to classify the factors as quantitative or qualitative.

For the quantitative factors, we also listed the required additional measures, which needed to be collected to give a better understanding of those factors. Eventually we were able to come up with the following list of quantitative factors and measures as shown in Table 1

Factor (X)	Type	Baseline Value	Remarks
Total number of accumulated bugs for product release (before entering the stabilization)	QUANTITATIVE	Avg. of 140+ bugs per product release	This is the main indicator for the leaked bugs which is the main cause of huge stabilization periods
The % of regression testing execution effort vs. the total testing execution effort	QUANTITATIVE	More than 70%	Indicator for the cost of regression tests (which was manual and not optimized)
Per Sprint → The % of leaked bugs vs. the total detected bugs	QUANTITATIVE	An avg. of 73%	This measure was an indicator for the bugs which are leaked from one development sprint to another
Per Sprint → The % of bugs detected and resolved in the same sprint	QUANTITATIVE	An avg. of 50%	This measure related specifically to new bugs detected in a development sprint and whether they were resolved in the same sprint or not
The % of leaked bugs vs. the total number of detected bugs in a production release	QUANTITATIVE	An avg. of 82%	The same measure as above but applied to the whole release
Average size of user stories per release	QUANTITATIVE	An avg. of 3 points	This was an indicator for the size of user stories and whether this size affected the progress and produce more bugs or not
Average time to complete the development of a single user story in a development sprint	QUANTITATIVE	8-10 days	This measure indicated that although the Definition of Done on the story level was weak, still the team took the whole sprint (10 working days) to complete any story because stories were always finished in the last one or 2 days of the sprint
Code coverage % of unit tests	QUANTITATIVE	Below 10%	At this point the code coverage of unit tests to the legacy code part was very poor and less than 10% which indicated the huge number of regression bugs and the repetition as well
Code metrics	QUANTITATIVE	NA	These were some code metrics related to the legacy code part. They were collected to help come up with refactoring opportunities

As for the qualitative factors, we came up with the following list:

- Very weak Done definition (on story level) which resulted in high counts of leaked bugs
- Ineffective mechanisms to gather feedback and contribution from team members during retrospectives which resulted in ineffective feedback channels between the team and the lack of value perception from conducting retrospectives
- Lack of understanding of the product design and architecture by the testing team (which led to consuming a lot of time in useless regression cycles)
- Mini-waterfalls iterations as the team tended to work on all user stories in parallel during the development sprints
- Manual regression tests (a large part of the regression test suite was still done manually which consumed a lot of time)
- Synchronization issues with another Middleware development team which resulted in late integration and build issues
- Lack of skills for the new team members in unit testing
- Legacy code challenges which prevented the team from creating isolated unit testing for old features

Those factors (quantitative and qualitative) constituted our current performance baseline at that time. It represented a snapshot of the current issues and problems, which needed to be addressed. This was the basis for improvement as well, and whatever improvement actions were to be taken later on were expected to eventually contribute to improving those factors.
Luckily most of the contributing factors (X's) identified during the Define phase, were controllable factors as they were related to our team and inside our boundaries of control. The exception to this was the factors related to the infrequent builds from the other middleware product team, which resulted some times in late integration issue with our team.

3.5 Analyze phase:

During the analyze phase, we studied the factors identified at the previous section, through:

- Interviewing different team members to obtain their input regarding the weight of each of the identified factors. For example, we met with the testing team and discussed with them the challenges they faced in regression testing, and the effort consumed in this activity. We also discussed with them the reasons for delaying testing activities inside a development sprint (mini-waterfall iterations) and the challenges they faced in creating efficient regression tests while having poor understanding of the product design
- Investigated some artifacts such as the team Done definition, regression test scenarios and scripts samples, and unit test coverage for new features vs. old features from previous releases
- We investigated the measurement baselines for bugs which included data about bugs detected during the development sprints, % of leaked bugs from each sprint and the % of regression bugs at the end of each release
- We studied some correlation relations between the size of user stories, and the amount of bugs detected during development and the amount of bugs leaked afterwards

Throughout the Analyze phase, we were able after investigating and examining the root causes to come up with a prioritized list of root causes for which improvement actions were to be identified. Usually at this point, a Six Sigma team should think about what we call the low hanging fruits or the quick wins. When we started our analysis, we began to consider the X's, which were considered as quick wins. Working on those X's and improving them was expected to result in quick improvements to our main problem (Y), which was concerned with decreasing the time spent in stabilization activities through decreasing the number of accumulated leaked bugs.
At this point we categorized our X's and root causes in 4 main areas. Each area had a set of both quantitative and qualitative X's as well.

- Late discovery of problems and ineffective retrospectives:
    - Ineffective mechanisms to gather feedback and contribution from team members during retrospectives which resulted in ineffective feedback channels between the team and the lack of value perception from conducting retrospectives

- Accumulation of bugs resulting from late detection and resolution:
    - For each development sprint:
        - The % of leaked bugs vs. total detected bugs during the sprint
        - The % of bugs detected and resolved in the same sprint
    - The number and % of leaked bugs vs. the total number of detected bugs in a release
    - Average size of user stories per release
    - Average time to complete the development of a single user stories in a development sprint
    - Very weak Done definition (on story level) which resulted in high counts of leaked bugs
    - Mini-waterfalls iterations as the team tended to work on all user stories in parallel during the development sprints
    - The % of regression testing execution effort vs. the total execution effort of the testing team during a development release (was huge due to the late involvement of testing execution activities in a development sprint)
- Regression test effectiveness issue:
    - Lack of understanding of the product design and architecture by the testing team (which led to consuming a lot of time in useless regression cycles)
- Late Integration issues:
    - Synchronization issues with another Middleware development team which resulted in late integration and build issues

Other factors related to unit test coverage, and legacy code refactoring opportunities and regression test automation were candidates for bigger six sigma projects as they involved different challenges and working on them needed a huge investment at that time so they needed to be dealt with differently.

3.6 Improve phase:

At this stage, we had our list of root causes, and through our interviews and brainstorming meetings we were able to come up with a list of improvement actions to start implementing and piloting across future sprints.
The improvement actions categorized by the areas identified in the Analyze phase were:

- Regression test effectiveness issues:
    - Established the habit of close communication between testing and development members regarding the product design and architectural details which were useful for the testing team to know in order to make their regression suites more effective and efficient
- Late Integration issues:
    - Worked on establishing solid contact with other teams and synchronize with them early checkpoints for integration rather than late ones at the end of the release
    - Worked with the Product Owner and the team to identify stories which had strong emphasis on integration with other products and prioritizing those stories early during the release
- Late discovery of problems and ineffective retrospectives:
    - Coaching the team during the retrospective, and introducing new techniques for them by facilitation in order to encourage contribution of all team members and enhance the problem/impediment detection and removal cycle which the team clearly had an issue with during the previous releases
- Accumulation of bugs resulting from late detection and resolution:
    - Revisiting and strengthening the Done definition on the story level to avoid having bugs leaked from one sprint to another
    - Coaching the team to come up with more slicing guidelines for the large user stories which caused utilization issues during the sprint and led to late testing and bug leakage
    - Worked with the team to establish a better-streamlined model for managing the workflow inside a sprint rather a waterfall-based model. The team was encouraged to work on less number of stories at a time by limiting the work in progress (WIP) to decrease testing bottlenecks
    - Worked on improving some unit testing practices (to increase the coverage of unit tests and decrease regression bugs)

These improvement actions were introduced gradually through multiple iterations and with the introduction of these actions, improvement were eventually measured.

During the next production release we started to sense improvements one sprint after another and by the end of the release we were able to show some improvements statistically.

Eventually we are able to measure improvements in different factors (X's), which contributed to an improvement in our main problem (Y), as shown in Table 2, which includes the new baselines obtained after improvement.

Table 2: New performance baselines obtained after improvement

Factor (X)	Value before	Value after	Remarks
The % of regression testing execution effort vs. the total testing execution effort	More than 70%	Between 50% - 55%	We measured improvement in this factor as a result of optimizing the testing activities, but still it was minor, because investing in automating their regression suite (more than 1000 test cases) was a candidate for a separate DMAIC project
Per Sprint → The % of leaked bugs vs. the total detected bugs	Median = 73%	Median = 32%	We measured great improvement in this factors as a result of working on the Done definition and optimizing the team activities by limiting the work in progress during their development sprint
Per Sprint → The % of bugs detected and resolved in the same sprint	An avg. of 50%	100%	
The % of leaked bugs vs. the total number of detected bugs in a production release	An avg. of 82%	An avg. of 41%	The same measure as above but applied to the whole release
Average size of user stories per release	An avg. of 3 points	An avg. of 2 points	At this point, we found that the problem wasn't mainly in the avg. story size, but in the slicing of the stories. Some stories were still big and needed slicing even though their size estimates were not big relative to other stories. So improvement came from mainly adopting better slicing techniques in the subsequent releases
Average time to complete the development of a single user story in a development sprint	8-10 days	3-5 days	This measure reflected the improvement after streamlining the development inside the iteration and the early involvement of the testing team accordingly
Code coverage % of unit tests	Below 10%	Below 10%	At this point we hadn't worked on the unit test coverage as it was a candidate for a separate Six Sigma DMAIC project
Total number of accumulated bugs for product release (before entering the stabilization) **(Main X)**	Avg. of 100+ bugs for a 6 sprint product release	Avg. of 25 bugs for a 6 sprint product release	This was the indicator for improvement in the main X contributing to our problem which was a result of the improvement of all other X's
The % of stabilization sprints from the total product release duration **(Y)**	An avg. of 36%	An avg. of 10%	This was the main indicator for improvement in our main problem (Y)

Eventually, as a result of improving those factors we were able to improve the whole response (Y) of our project, which was the number of leaked bugs at the end of the release and the % of stabilization phase vs. the total duration of the release.

3.7 Control phase:

Basically, out Control phase was about making sure that the improvements achieved were sustainable. This was achieved through deploying the improvement actions made throughout all future releases and incorporating new practices in the team ongoing process.

For example, at this point (1) the new Done Definition replaced the old one and was communicated to the management and the product management team as well. (2) New team members were trained on unit testing, and the team formulated a process for new comers where they can learn unit testing through pairing with older team members. (3) Strict Integration checkpoints were established between Virtue team and the Middleware development team, which guaranteed that integration, would not be delayed till the stabilization phase.

4. ADOPTING AGILE IN MANAGING A DMAIC-BASED SIX SIGMA PROJECT:

4.1 Introduction to Process Increments:

'Process Increments' is an agile-based method for managing process improvement projects. This method basically builds on agile values and principles and leverages of the power agile methods such as Scrum and techniques such as User Stories.

Basically, this method partitions the scope of a process improvement project to user story-like increments and populate a backlog of this increments which is implemented through improvement sprints and are managed using agile project management techniques.

Typically, a process increment is a process improvement chunk, which can be implemented in a relatively small time (1-2 weeks) and should provide value to the organization.

A process increment has the following attributes:

- **Name**: The name of the process increment which describe the title and objective of the increment
- **Conditions of Satisfaction:** These are the acceptance criteria or conditions which when achieved imply that the increment is fulfilled
- **Size Estimate**: This is the relative size estimate of the increment compared to other process increments in the project. Process increments are estimated relatively in points such as user stories, and usually the relative size estimate indicate the time needed to implement such increment relative to other increments considering other factors such as complexity or the difficulty of implementation.
- **Process Area:** This is the theme or category of the process increment.

For example, if we use process increments to manage a Six Sigma DMAIC-based project, we would partition the scope of the project into increments where each increment will resemble a user story with the attributes mentioned above.

An example of a process increment from the Define phase is shown in figure 3.

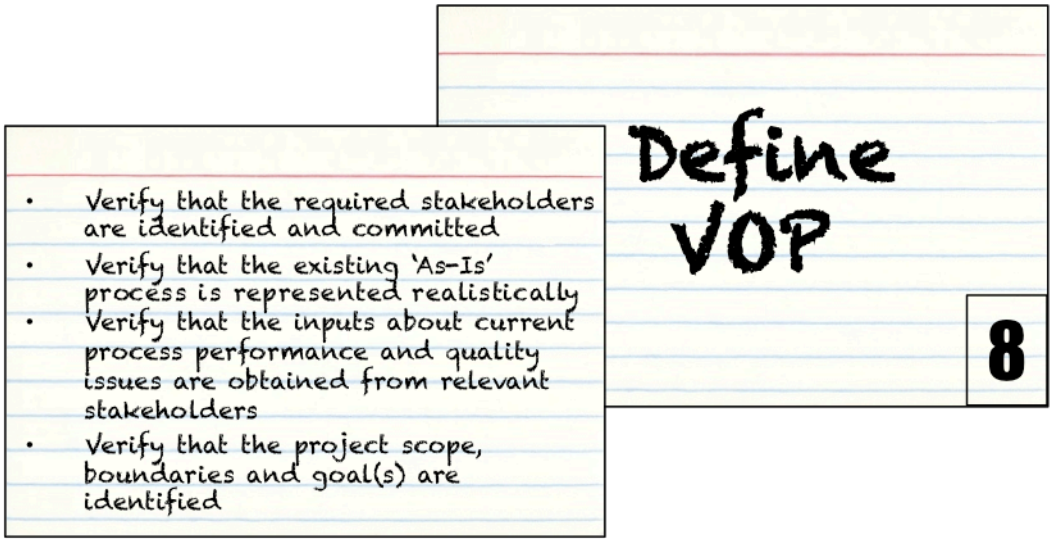

Fig. 3. A typical process increment from a DMAIC-based Six Sigma project. The increment belongs to the "Define" area or theme, and basically the scope of the increment is to define the Voice of the Process (VOP), which is a key part of the Define phase.

4.2 How Process increments were used to manage our DMAIC project:

In principle, Six Sigma is about process improvement and problem solving. A DMAIC-based project specifically is about improving an existing process and solving problems, which results in introducing changes, which can be big and usually accompanied by many challenges and risks. Also there's a large degree of uncertainty and scope vagueness in a DMAIC project, and progress sometimes is not visible across all stakeholders.

If we consider a DMAIC-based project; applied to a software development team; as a typical project then there's no better alternative to managing its uncertainty and big changes than with agile methods, and this where process increments as a method comes in handy.

The typical DMAIC roadmap implies a staged model for managing Six Sigma projects, and this can challenging depending on the scope of the project and how big is the change to be introduced. Also, depending on the magnitude of the problem, and the factors contributing to this problem, a Six Sigma team may need to divide their Measure, Analyze, and Improve phases into smaller cycles where in each cycle some of the contributing factors are measured, analyzed and then improvement actions can be piloted incrementally and iteratively with their effectiveness evaluated through successive pilots rather than a big bang implementation approach.

A software development project following traditional methods (e.g. waterfall) would probably suffer in implementing Six Sigma techniques because it's an environment which impedes changes and provide less number of feedback cycles which results in poor performance. On the other hand, agile projects offer the perfect environment for experimentation needed to deploy process changes incrementally and iteratively while managing the transition risks and challenges in an effective manner as well.
In our project, we realized that the DMAIC cycle was better to be managed in an iterative and incremental fashion. We adopted the "Process Increments" with the following simple steps:

- First, we began to establish our DMAIC project backlog. This backlog included the process increments which were the building blocks which represented all the units of valuable work which needed to be covered in our Six Sigma project
- The structure of the improvement backlog is shown in Table 3.
- Increments were implemented across improvement sprints. The length of an improvement sprint was typically 2 weeks, and the team ran those improvement sprints parallel to their normal development sprints

Eventually the DMAIC roadmap went from being a typical staged roadmap to an iterative and increment one. Our new roadmap became something like: **[D]** -->[M --> A --> I --> C] --> [M --> A --> I --> C] -->[...] where we first identify the problem (Y) and the main contributing factors (X's) in an initial Define phase, and then we undergo small cycles of measuring those factors, analyzing their root causes, piloting specific improvement actions across improvement sprints and then eventually defining criteria for maintaining the improvements achieved.

Table 3: A snapshot of the improvement backlog populated with process increments

THEME / PROCESS AREA	PROCESS INCREMENT	SIZE	CONDITIONS OF SATISFACTION
DEFINE	Define VOP	8	• Verify that the required stakeholders are identified and committed • Verify that the existing 'As-Is' process is represented realistically • Verify that the input about current process performance and quality issues is obtained from relevant stakeholders • Verify that the project scope and boundaries, and goal(s) are identified
DEFINE	Create an initial Business Case	5	• Verify that a Problem statement is defined • Verify that the magnitude of the problem is assessed and defined • Verify that the initial CoPQ (Cost of Poor Quality) is assessed and calculated • Verify that the initial improvement goals are identified • Verify that the initial Cost-Benefit Analysis is conducted

MEASURE	Collect and gather data about X's	5	• Verify that the list of X's contributing to the main problem is completed • Verify that X's are classified as (quantitative vs. qualitative) • Verify that X's are classified as (controllable vs. uncontrollable) • Verify that required information and facts about X's are available in order to measure X's • Verify that major X's are measured and baselined as the basis for further improvement

The team implemented process increments across a number of successive improvement sprints. Those sprints were parallel to their product development sprints, and their improvement velocity was measured to indicate progress in the Six Sigma project.

Our Done definition for the process increments was as follows:

- All conditions of satisfaction are satisfied and validated
- All required stakeholders have been involved in implementing the increment
- Necessary knowledge about the process increment activities has been acquired by the team

During the Improve phase, the process increments identified at this point were related to implementing and adopting new improvement actions, so they shared some additional criteria to their Done definition such as:

- Process increment should be implemented at least on 3 sprints before evaluation
- Current processes, tools and guidelines should be updated after implementing the process increment

5. CONCLUSION

5.1 Problems faced by agile teams:

Many agile teams depend on daily collaborations and periodic vehicles such as retrospectives to identify and work on the problems and issues they face. During those vehicles, they would typically focus on soft observations and short-term issues. Any decision making at this stage is mainly based on perceived symptoms of short term and visible problems.

Sometimes, many agile teams would be so neck-deep in their daily development and challenges that they overlook the hidden problems or impediments, which impacts their performance and quality on the long run.

In addition to the issues described above, many teams would find it hard to express those types of issues and propose improvement actions to their management, because there would be a communication gap. In order for the management to take informed decisions, they need more than simple observations or high level improvement initiatives. They need deeper insight into the problems, cost-benefit analysis figures, ROI data or any other form of empirical information based on solid data. They need to eventually understand the magnitude of the problems and issues they are facing and relate those problems to an impact on the business that they can perceive well.

5.2 Solution offered by Six Sigma:

Six Sigma offers agile teams the roadmap (DMAIC), approach and techniques, which can help them effectively define and work on the problems and issues affecting their quality and performance.

We have found that agile teams could benefit from Six Sigma in many ways such as:

- An analytical problem solving approach
- Having a very rich set of tools and techniques and approaches for long-term and strategic process improvement and problem solving
- Shifting their focus from just acknowledging or observing problems to effectively defining those problems along with their quantitative magnitude and impact on the business
- Shifting their focus from discussing symptoms of issues, to measuring and analyzing root causes of problems and eventually eliminating them
- Focusing on using quantitative data along with the right set of tools and asking the right set of questions to the right stakeholders in order to focus on the right problems

- Shifting their focus to be problem driven rather than solution driven. The team should strive to solve the right problem by eliminating its root causes rather than providing a catalogue of pre-defined solutions
- Unifying the team towards a specific goal for problem solving and improvement, and relating this to a business case, which relates better to customers and managers.
- Obtaining the buy-in and support from management, because it related better to them through effective problem definitions, and business cases for the solutions

5.3 Benefits of adopting agile practices and methods to manage DMAIC-based Six Sigma Project:

Adopting the 'Process Increments' method to manage the Six Sigma has resulted in many benefits such as:

- Gave us a powerful and structured yet lean approach to manage our project
- Managing the DMAIC project in an iterative and incremental fashion mitigates the risks associated with introducing, measuring and evaluating bulky changes
- Forming an improvement backlog of relevant process increments which can be reused in other projects and contexts
- Scope and Progress visibility for the DMAIC project which leads to more involvement and buy-in from different stakeholders specially management
- The involvement of the whole team in the journey to improve their performance and quality, with the support and buy-in of the management

During our experience, we were able to create an effective mix of the 2 philosophies and approaches. We have realized benefits in adopting Six Sigma DMAIC roadmap as an analytical problem solving method to our agile team, and we benefitted as well from adopting agile practices and techniques to manage our Six Sigma project itself using the agile-based method "Process Increments".

6. ACKNOWLEDGEMENTS

We would like to sincerely thank Radouane Oudrhiri for his recognizable work and key insights about Six Sigma and its application in information technology and software development specifically. Mohamed had an amazing opportunity to learn about advanced Six Sigma Black Belt topics and techniques from Radouane, in addition to being mentored and coached by him during 2010. We would also want to thank Tim O'connor for his review and insights about the paper, and we couldn't have done it without his support.

REFERENCES

1) Oudrhiri, Radouane. "Six Sigma and DFSS for IT and Software Engineering Position Paper", http://fmisociety.org/ITLeadersAcademy//lectures/13_1_presentation1.pdf
2) Amr Noaman and Mohamed Amr. "Process Increments – An Agile Approach to Software Process Improvement", IEEE Xplore Digital Library-Agile Conference (AGILE), 2011, E-ISBN: 978-0-7695-4370-3