MiniDMAIC: An Approach to Cause and Analysis Resolution in Software Project Development

Carla Ilane M. Bezerra, Adriano B. Albuquerque and Luiz Sérgio Plácido
Master Program in Applied Informatics, University of Fortaleza
Fortaleza, Brazil

1. Introduction

To achieve practices in CMMI a great amount of organizations are adopting Six Sigma as strategy. This methodology does not support practices of high levels of maturity but also of low levels (Siviy et al., 2005).
The Six Sigma and CMMI have compatible goals and the Six Sigma is, in most of the cases, extremely compatible with others quality initiatives that can be already implemented on the organization. The Six Sigma can be executed in macro and micro levels of the organization and can be successful either with elementary graphical tools or with advanced statistical tools (Dennis, 1994).
One of the fundamental aspects of the quality improvement is the analysis and resolution of problems. For this, a formal method of solving problems can be used, that may bring a lot of benefits, such as (Banas Qualidade, 2007):

- Prevent the problem solvers pass straight to the conclusion;
- Ensure the root-cause analysis;
- Demystify the process for solving problems;
- Establish analytical tools to use and determine when to use them.

In this context, the use of Six Sigma methodology's tools such as DMAIC, has been outstanding. Unlike other approaches to solve the problems, that focus only on eliminating the problem itself, the DMAIC methodology (Rath and Strong 2005) used by the Six Sigma comprises from the selection of issues that deserve a deeper treatment to the control of results obtained in the course of time.
The DMAIC method presents step by step how the problems should be addressed, grouping the aim quality tools, while establishing a standardized routine to solve problems with a proved efficient implementation in software organizations.
Although appropriate for the organizational level, the formal methods to solve problems can be not viable at projects level. A major challenge faced by companies that want the CMMI level 5 is exactly the implementation of the process area "Causal and Analysis Resolution - CAR" in the context of software projects, since they generally have very limited

resources. Thus, immediate actions are taken only to resolve problems and, in most of the cases, the same problems happen again.
Some works suggest approaches for analysis of causes focusing at the organizational level. However, it is often necessary to perform analysis of causes within the projects, quick and effective, attacking the root causes of the problem. In organizations that aim to achieve high levels on maturity mode, such as CMMI, this practice is required within the project to maintain adherence to the model.
Furthermore, none of the approaches investigated involving analysis and resolution of causes, is based on DMAIC. The proposed approach in this paper aims to make effective the root cause analysis in the context of projects providing a structured set of steps based on the DMAIC method, to be run in a simple way.
Despite all the benefits of using Six Sigma methodology in conjunction with the CMMI, the implementation of the process area "Causal Analysis and Resolution" in software projects often becomes impractical for the following reasons:

- DMAIC projects have duration between 3 to 6 months. However, projects require rapid resolution of their problems and cannot wait too long;
- Due to the great necessity of using statistical tools, the DMAIC can become excessively expensive, the savings may be less than the cost to achieve improvements, and the projects often have limited resources;
- The qualification level of the DMAIC team is quite strict, however, in the context of software development projects, other attributes such as business domain and project management can bring greater results than the fact of having a team with great knowledge in statistics.

Given this background, this work aims at developing an approach based on the DMAIC (Six Sigma), called MiniDMAIC, to address the process area "Causal an Analysis and Resolution" from CMMI, in software development projects, looking for reducing the disadvantages described above related to the use of DMAIC. It also aims to present the application of the methodology in software development projects in an organization using a workflow tool, which was implemented the practices of MiniDMAIC.
This work is organized into five sections, besides this introduction. In section 2, we present the theoretical basis related to Six Sigma and, more specifically, the DMAIC methodology. In Section 3, we discuss the CMMI process area "Causal Analysis and Resolution" pertaining to the maturity level 5. In section 4, we present the proposed approach, called MiniDMAIC. In sections 5 and 6, we present a mapping MiniDMAIC with the area of CAR and the DMAIC process, respectively. Aspects concerning the use of MiniDMAIC on real projects, and the obtained results are presented in section 7. In Section 8, contains papers relating to the preparation of the approach. Finally, in section 9, we present the final considerations and limitations of the proposed methodology.

2. The Six Sigma and the DMAIC Methodology

The Six Sigma é is a methodology that focuses on reducing or eliminating the incidence of errors, defects and failures in a process. The Six Sigma methodology also aims to reduce the process variability and can be applied in most of the sectors of the economic activity (Smith, 2000).

Achieving the Six Sigma means reducing defects, errors and failures[1] to zero and to achieve near the perfection in processes' performance. The methodology combines a rigorous statistical approach to an arsenal of tools that are employed in order to characterize the sources of variability to demonstrate how this knowledge can control and optimize the process results (Watson, 2001).
The Six Sigma methodology aims to define the obvious and not obvious cause that affect the process in order to eliminate or improve them and controlling them (Rotondaro 2002).
The Six Sigma presents some techniques to address problems and improvements, such as DMAIC (Define, Measure, Analyze, Improve and Control), DCOV (Define, Characterize, Optimize, Verify) and DFSS (Design For Six Sigma). In this work, the DMAIC methodology will be used.
The DMAIC methodology was created by General Electric and, according to Tayntor (2003), is the most used in companies that implement the Six Sigma, and also more suitable for software development.
The DMAIC methodology consists of five phases: define, measure, analyze, improve and control. In the phase "define" is necessary to identify the problem and then to define the existent opportunities to resolve it according to the customer requirements. In phase "measure", the current situation should be verified through quantitative measurements of the performance, so that subsequent decisions are based on facts. In phase "analyze", the achieved performance and their causes should be identified and the existent opportunities should be analyzed. After doing this analysis, it is possible to perceive points to improve the performance and to implement improvements in phase "improve." In phase "control" the improvement should be ensured, through the control of the deployed process performance.
Pande (2001) highlights that one cannot use the DMAIC for any improvement. A Six Sigma improvement project, according to the author, must have three qualifications:

- There is a gap between current performance and required/expected performance;
- The cause of the problem is not understood clearly;
- The solution is not predetermined, nor is the optimal apparent solution.

Besides, the viability criteria should be observed, such as: the necessary resources, available skills, the complexity, the probability of success and support and engagement of the team.

3. The CMMI and the Causal Analysis and Resolution

The Capability Maturity Model Integration (CMMI) (Chrissis, 2006) is a maturity model for the development of products developed by the Software Engineering Institute (SEI), which is increasingly being adopted by software organizations, since this model aims to guide organizations in implementing continuous improvements in their development process.

3.1 The Maturity Level 5

The focus of the maturity level 5 is the continuous improvement of processes. While level 4 focuses on the special causes of variation in the organization' process, level 5 tries to find common causes and address them, resulting in many improvements, which are

1 On methodology Six Sigma, the defects, errors and failures are any deviation of a characteristic that generate custome dissatisfaction (Blauth, 2003).

implemented in a disciplined manner. Measurements are used to select the improvements and estimate the costs and benefits to meet the proposed improvements. The same measurements can be used to justify efforts for further improvements (Kulpa, 2003).
The CMMI level 5 consists of two process areas: Organizational Innovation and Deployment - OID and Causal Analysis and Resolution – CAR. The latter is the focus of this work.
The goal of the Causal Analysis and Resolution - CAR is to identify causes of defects and other problems and take actions to prevent their occurrence in the future.
Table 2 shows the relationship of specific goals (SG) with their respective specific practices (SP) for this process area.

SG 1	**Determine Causes of Defects**	
	SP 1.1	Select Defect Data for Analysis
	SP 1.2	Analyze Causes
SG 2	**Address Causes of Defects**	
	SP 2.1	Implement the Action Proposals
	SP 2.2	Evaluate the Effect of Changes
	SP 2.3	Record Data

Table 1. Causal Analysis and Resolution in CMMI (Chrissis, 2006)

4. MiniDMAIC

The MiniDMAIC is a strategy that aims to simplify the DMAIC method in order to address the causes and resolution of problems in software development projects in a more practical and faster manner, with less risk and cost, preventing future recurrences, implementing improvements on the development process and thus, continually increasing the customer satisfaction (Gonçalves et al., 2008 and Bezerra et al., 2009).
This approach was originally defined in Gonçalves (2008a) and was applied in pilot projects in a software organization that was deploying the levels 4 and 5 of the CMMI model. During the implementation of the approach in the pilot projects some improvements to the approach were identified and so it was refined.
Based on the implemented improvements, the MiniDMAIC was executed in other software development projects and a second work has been published with case studies of some projects that implemented the refined approach (Bezerra et al., 2009). After this last work, improvements were added to the approach and were validated in a CMMI level 5 official assessment in the organization that was executed the MiniDMAIC. We can see that the approach presented in this work underwent for several validations and was refined and implemented in several software development projects, demonstrating effectiveness in the analysis and resolution of causes in the context of these projects.
The great difference between MiniDMAIC and DMAIC is that the DMAIC, from the analysis and resolution of the causes of the defined prolem defined, has the main objective the improvement of one of the organization’s standard processes, implementing the improvements in a controlled manner in the organization. The MiniDMAIC addresses the causes only in the project level and aims to prevent and treat the defined problems through the analysis and resolution of the problems root-causes. It can assist only in the organizational processes improvement (Bezerra et al., 2009).

Moreover, the DMAIC requires a statistical proof of the problems causes and achieved improvements, that is not required in MiniDMAIC, which identifies and prioritizes the causes using simpler tools such as : Ishikawa diagram and Pareto Charts, and analyzes the obtained improvements observing the progress of the project's indicators (Bezerra et al., 2009).

The main characteristics of MiniDMAIC are:

- Short duration;
- Need for basic knowledge of statistics;
- Linked to risks;
- Low cost when compared to DMAIC;
- Suitable for software development projects.

The problems that need to be addressed more careful by applying the MiniDMAIC approach can be defined at the organizational level (ex.: control limits, number of defects, etc.). However, it is important to clear that, to the project team, the difference between problems that require only simple and immediate actions, and those that require the treatment defined in MiniDMAIC. Simple actions are appropriate for treatment of simple improvement items which can be typically performed by a person with little effort and when the cause/solution is known or likely.

The execution of the MiniDMAIC in a software development project must also consider the size of the project and the frequency of the indicators collection in an organization. For organizations that collect monthly the indicators, the execution of the approach should consider that the project must have at least one month in duration. If the project has short iterations, the treatment of the problem by MiniDMAIC approach will be useful to prevent the problem does not occur in later iterations. For month-long projects the action's execution can end up at the end of the project. Although the action does not address the problem in time to present the effects of the improvements in the project, the execution of this action may have benefits that will help other organization's projects.

Examples of project's problems that deserve treatment by MiniDMAIC approach are:

- Out of control project, where the results of the indicators of statistically controlled processes do not satisfy the specification limits defined by the project or organizational baseline boundaries (e.g., productivity, delivery deviation, defect density, etc.);
- Recorrent problems in the project;
- High number of defects found in systemic tests;
- High number of defects found by the customer.

When the cause and defect analysis is performed, the selection of defects for analysis must take into account the following factors:

- Types of most common defects;
- Frequency of occurrence;
- Similarity between defects.

In this approach, defects are considered as failures, taking into account the defect, error and failure definitions presented in the IEEE 610.12-1990. We chose to use these concepts in a similar way, because the MiniDMAIC approach bases the phase "Measure" on the orthogonal defect classification (Chillarege et al., 1992), which uses the same definition.

As support to the approach, tools like: spreadsheets, project management tools, among others, may be used.

The items below describe the phases of MiniDMAIC, which uses the same phases of the DMAIC method, and a final phase that was included to provide the improvement opportunities, identified during the execution of the approach, to the organizational assets. The Figure 1 shows the sequence of steps of the approach.

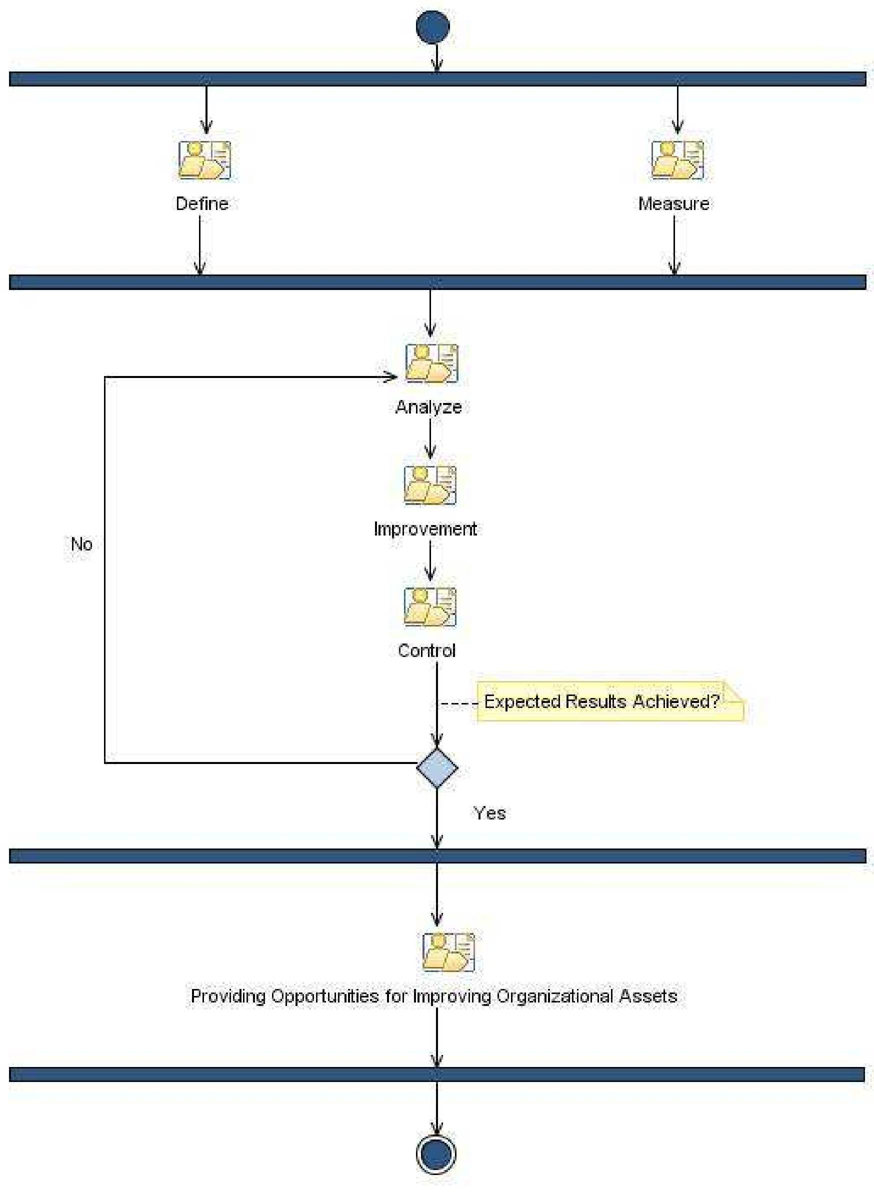

Define
Measure
Analyze
Improvement
Control
No
Expected Results Achieved?
Yes
Providing Opportunities for Improving Organizational Assets

Fig. 1. Phases of MiniDMAIC

4.1 Phase: Define

The phase "Define" is a phase of action planning and encompasses the definition of the problem, sources, impacted processes and subprocesses and expected results. Besides, the formation of the team (Table 2).

Define	Step 1 – Define the Problem	The problem trat will be adressed must be defined to be clear its importance and defined its objectives. A search should be made on the historical organizational base to look for similar problems that were treated in other projects using a MiniDMAIC action to help in defining and solving the problem's root-causes. It is important to describe the impact or consequences of the problem in the project. This description should be focused only on symptoms rather than in causes or solutions.
	Step 2 - Determine the Source of the Problem	This step should show what was the source who revealed the occurrence of the problem. Examples of sources of problems in software development projects are: • Project's indicators; • Report of systemic tests; • Results of integration tests; • Client's test report; • Problems identified in technical review that affect the requirements or the correct operation of the software; • Customer Complaints.
	Step 3 - Identify the Affected Processes	Identify which processes and subprocesses were affected by the defined problem. If the problem is the result of an out of control indicator, the baseline associated to the process should be identified associated with baseline. The process baselines selected by the project should consider the client's performance objectives.
	Passo 4 - Identify the Risks Related to do not Address the Problem	The risks related to do not address the problem can be identified by the project manager in order to treat and monitor them according to process defined to the process area Risk Management - RSKM of CMMI.
	Passo 5 – Define the Expected Results	In this step, the expected results to be achieved with the implementation of MiniDMAIC approach are defined aiming to address the problem. The expected results must be defined in a quantitative manner, and indicators associated with the defined problem can be used.
	Passo 6 – Forming the team and Estimating the Time of Execution	In this step the team that will participate in each phase of MiniDMAIC is formed and the time for implementing each one is estimated. In a MiniDMAIC action is not necessary to have Black Belts as leader. As they are simple and directly related to the project, the only need is a basic knowledge in Six Sigma and training in MiniDMAIC approach. The most important is the understanding of knowledge related to the project and management techniques and it is important that the Project Manager leads the MiniDMAIC. The MiniDMAIC team size may vary according to the needs of the problem. In situations that we may have just the project manager and a team member others collaborators can participate only in certain steps, for example, the support of a Green Belt leader (especially during the phases Measure and Analyze).

Table 2. Steps of the Phase "Define"

4.2 Phase: Measure

The phase "Measure" is the collection and analysis of measurements (existing or to be defined) related to the problem aiming to know the current situation of the project and the related processes, as shown in Table 3. This phase can be executed in parallel to the phase "Define", supporting the definition of the problem. If the results of the measurements are analyzed at the project level, the analysis must be verified in the report that comprises the collected data and the measurements' analysis. If the defined measurement is within the MiniDMAIC action, this should be collected and analyzed in the phase "Measure".

Measure	Step 1 – Plan the Measurements	In this step we should examine whether there is a need for a new measurement that provides more evidences for the problem at hand. In most situations, the measurements are already being conducted in accordance with the defined process that addresses the process area Measurement and Analysis - MA. A new measurement can also be planned to provide more evidences to consolidate and enlarge the understanding of the problem and its consequences.
	Step 2 – Measure the Current Situation	The measurements selected in the previous step must be executed according to the plan. It is necessary to collect information and measure the current situation of the project. Later, these same measures will be used to measure the obtained improvement. In case of collection of defects, it is recommended to use the template - Analysis of Causes provided by Bezerra (2009b), in order to prioritize the defects that deserve a more detailed analysis of the causes.

Table 3. Steps of the Phase "Measure"

4.3 Phase: Analyze

The phase "Analyze" encompasses the identification and prioritization of the problem's root causes using techniques to ensure that the root causes to be addressed are actually related to the problem and to the definition of possible actions to solve the problem, as we can see on Table 4.

Analyze	Step 1 - Determine the Problem's Causes	This is one of the most important steps of MiniDMAIC, since its purpose is to find out the problem's root cause. If this step is not done correctly, the result of MiniDMAIC may be compromised because all of the following activities will be based on the outcome of this step. So, it is important that the people who has knowledge related to the problem and can contribute with information about their causes. Examples of techniques to determine problem's causes are: brainstorming, five whys, cause and effect diagram (Ishikawa, 1985), among others. To execute this step the Template "Analysis of causes" provided by Bezerra (2009b) can be used. If defects are analyzed, the classification of defects to determine where the defects are more concentrated should be used as input for this phase.
	Step 2 – Priorityze the Problem's Causes	The prioritization of the problem's causes must be carried out in accordance with the process defined to the area Decision Analysis and Resolution - DAR. Another way to prioritize the causes is using the Pareto chart (Juran, 1991), where 20% of the causes can contribute to 80% of defects. If the Pareto chart is adopted, the causes can be grouped according to the level of criticism of the defects, the origin of the defects and the type of them. To execute this step, the Template - Analysis of causes provided by Bezerra (2009b) can be used.
	Step 3 – Define Candidate Actions	In this step, the possible actions to address the problem should be identified with the project team using the brainstorming technique. Every action should be linked to the related causes.

Table 4. Steps of the phase "Analyze"

4.4 Phase: Improve

The phase "Improve" comprises the definition and the analysis of feasibility of the proposed the working up and implementing of the action plan and the monitoring the obtained results (Table 5).

Improve	Step 1 – Prioritize the Actions	The candidates actions can be prioritized according to the process defined to the process area Decision Analysis and Resolution - DAR. A analysis of feasibility can also be carried out for the implementation of each action. Any priorityzed cause may have one or more actions, as well as an action can be addressing one or more causes prioritized in phase "Analyze". Besides, they should be traceable. The analysis of feasibility should verify aspects such as: complexity, time and cost to implement the action within the project.
	Step 2 – Prepare and Execute the Action Plan	An action plan for the implementation of the priority and approved actions should be worked up by the project manager to address and follow up the actions. This plan should contain the following information: • Tasks to be performed; • Responsible for executing the task; • Effort required to perform the task; • Deadline to complete the task. In the execution of the action plan, the tasks can be distributed to the project team.
	Step 3 – Monitor the Actions	In this step, the tasks should be monitored in order to know the progress of MiniDMAIC. These results should be followed up by the project manager according to the process area Project Monitoring and Control - PMC.

Table 5. Steps of the phase “Improve”

4.5 Phase: Control

The phase "Control" comprises the measurement, evaluation of obtained results and dissemination of results and lessons learned (Table 6).

Control	Step 1 – Measure the Results	After the implementation of the actions in the project, the project manager and its team should measure the results obtained in the period using the same indicators selected in phase "Measure" in order to verify if the quantitative result was achieved.
	Step 2 – Evaluate the Results	When the obtained results are evaluated, an analysis should ne carried out by the project manager and its team to verify if the expected results established in the phase "Control" have been achieved and whether there was an improvement when compared to what was collected in the phase "Measure" before of the problem’s treatment . This comparison will be useful as a basis to confirm if there was an improvement on the project and to verify if the problem was actually addressed.

	Step 3 – Publicize the Main Results and Lessons Learned	After the execution of MiniDMAIC, the results should be shared by the project throughout the organization, recording them in an organizational repository, accessible to all projects. Sharing this information can be useful to address similar problems in other projects, as well to improve the process at the organizational level. The way to publicize should follow the process defined to the process area Organizational Process Focus - OPF, which defines how the lessons learned must be shared by the organization. If possible improvements to organizational processes were identified, they should be sent to the Engineering Process Group - EPG to be analyzed and properly addressed.

Table 6. Steps to phase "Control"

4.6 Providing Improvement Opportunities for the Organizational Assets

The organizational historical base should include much information from the execution of MiniDMAICs projects. Considering data from more than one project, the engineering process group can analyze more data aiming to identify trends of problems in order to define improvements to be implemented in the processes and their assets at the organizational level. If the problem has already a known cause, or causes have just been identified within the projects, a single action organization must be defined.

Besides the MiniDMAIC, according to Albuquerque (2008), the following data sources may also help to identify recurrent problems in the organizational process assets: (i) evaluation of process suitability, (ii) evaluation process adherence; (iii) evaluation of the work products to the standards established in the organization, (iv) post-mortem analysis, (v) indicators for monitoring the processes, (vi) lessons learned (vii) request for exemption the execution of activities, (viii) guidelines, (ix) rationales to addapt the process and (x) requests to change the process.

It is important to highlight that some of these sources can be useful, also, in the context of the defects and problems.

Some information should be registered in the organizational historical base as: type of problem, problem's causes, actions taken to treat the causes and obtained improvements. These information are important to organize the problems identified in the projects using the approach MiniDMAIC in order to enable the identification of problems at the organizational level.

5. Dmaic x minidmaic

The MiniDMAIC is based on the steps of the DMAIC method defined by Tayntor (2003). Some steps have been suppressed due to the complexity of the used statistical techniques, for example, the step "Calculating the Current Sigma Level". And the steps related to customer requirements and changes in the standard processes were also removed, as illustrated in Table 7. The main goal of MiniDMAIC is to analyze and solve the causes of software development projects and does not focus on changes in the organization's standard process, which is the main goal of DMAIC.

Phase	DMAIC (Steps)	MiniDMAIC (Steps)	Rationales to Suppress the Steps
Define	Define the Problem	- Define the Problem	-
	Forming the team	- Forming the team and Estimating the Time of Execution	-
	Establish a Project Charter	- Determine the Source of the Problem - Identify the Risks Related to do not Address the Problem - Identify the Affected Processes - Define the Expected Results	-
	Prepare the Project Plan	-	- There is no need to have a lot of plans to analyze the causes in projects
	Identify the Customers	-	- The customer must be identified in the software project plan.
	Identify the Resulting Artifacts	- Throughout the implementation of MiniDMAIC	-
	Identify e Prioritize the Customer Requirements	-	- Customer requirements are not identified directly. They can be related to the step "Identify Affected Processes".
Measure	Define the Measurements	- Plan the Measurements	-
	Conduct Measurements	- Measure the Current Situation	-
	Calculate the Current Sigma Level	-	- A high knowledge in statistical techniques is necessary, which may be impractical in the context of projects.
	Determine the Process Capability	-	- A high knowledge in statistical techniques is necessary, which may be impractical in the context of projects.
	Carry out the Process Leaders Benchmark	-	- This Benchmark should be carried out at the organizational level, since there are no changes in the process executed at the project level.
Analyze	Determine the Causes of Variation	- Determine the Problem's Causes - Priorityze the Problem's Causes	-

	Carry out the Process Improvement Ideas Brainstorming	- Define Candidate Actions	-
	Determine the Improvements that have Major Impact on Customer Requirements	- Define Candidate Actions	-
	Prepare the Proposed process Map	-	- There is no need to provide changes in the process executed at the project level at the project level.
	Evaluate the Risks Associated with the Reviewd Process	-	- There is no need to provide changes in the process executed at the project level at the project level, because the risks can be addresses at organizational level.
Improve	Obtain the Approval of the Proposed Changes	- Prioritize the Actions	-
	Finalize the Implementation Plan	- Prepare and Execute the Action Plan	-
	Implement the Approved Changes	- Prepare and Execute the Action Plan - Monitor the Actions	-
Control	Establish the Key Metrics	- Measure the Results - Evaluate the Results	-
	Develop the control Strategy	-	- There is no need to provide changes in the process executed at the project level at the project level.
	Celebrate and Communicate the Success	- Publicize the Main Results and Lessons Learned	-
	Implement the Control Plan	-	- There is no need to provide changes in the process executed at the project level at the project level.
	Measure and Communicate the Improvements	- Measure the Results - Evaluate the Results - Publicize the Main Results and Lessons Learned	-

Table 7. Comparison of DMAIC Steps Defined by Tayntor (2003) and MiniDMAIC Approach Steps.

6. Minidmaic x CAR

For a better understanding of the relationship between MiniDMAIC and the Causal Analysis and Resolution (CAR) process area, a mapping was prepared to represent the relationship between the MiniDMAIC steps and the specific practices of CAR as we can see in Table 8. It is important to emphasize that such relationship does imply that the approach is covering the entire practice, since the process area is not related only to projects, but also has subpractices to the organizational level.

Phase	MiniDMAIC (Steps)	CAR (Specific Practices)	Observations
Define	Step 1 - Define the Problem	-	Related to Quantitative Project Management – QPM PA
	Step 2 - Determine the Source of the Problem	-	Related to Quantitative Project Management – QPM PA
	Step 3 - Identify the Affected Processes	-	Related to Quantitative Project Management – QPM PA
	Step 4 - Identify the Risks Related to do not Address the Problem	-	Related to Risk Management – RSKM PA
	Step 5 - Define the Expected Results	-	Related to Quantitative Project Management – QPM PA
	Step 6 - Forming the team and Estimating the Time of Execution	-	Relaletd to Project Monitoring and Control – PMC PA and GP 2.7 - Identify and Involve Relevant Stakeholders
Measure	Step 1 – Plan the Measurements	SP 1.1 - Select Defect Data for Analysis	-
	Step 2 – Measure the Current Situation	SP 1.1 - Select Defect Data for Analysis	-
Analyze	Step 1 - Determine the Problem's Causes	SP 1.2 - Analyze Causes	-
	Step 2 - Prioritize the Problem's Causes	SP 1.2 - Analyze Causes	-
	Step 3 - Define Candidate Actions	SP 1.2 - Analyze Causes	-
Improve	Step 1 – Prioritize the Actions	SP 2.1 - Implement the Action Proposals	Related to GP 2.10 - Review Status with Higher Level Management
	Step 2 - Prepare and Execute the Action Plan	SP 2.1 - Implement the Action Proposals	-

	Step 3 - Monitor the Actions	-	Related to Project Monitoring and Control – PMC PA
Control	Step 1 - Measure the Results	SP 2.2 - Evaluate the Effect of Changes	-
	Step 2 - Evaluate the Results	SP 2.2 - Evaluate the Effect of Changes	-
	Step 3 - Publicize the Main Results and Lessons Learned	SP 2.3 - Record Data	-

Table 8. Relationship between the MiniDMAIC Steps and Specific Practices of CAR

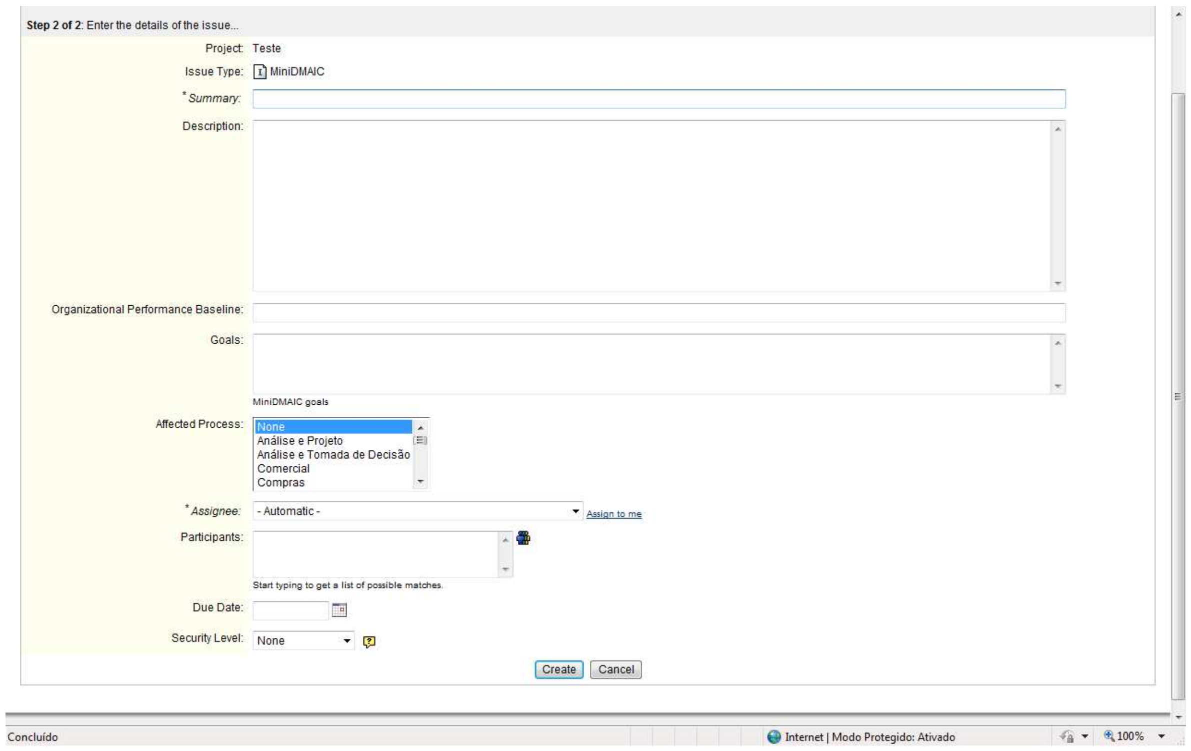
As can be observed, to be attend the process area Causal Analysis and Resolution – CAR, several steps defined in DMAIC were not necessary. The analysis of the DMAIC phases was the basis for defining the proposed approach.

7. Minidmaic Execution

The MiniDMAIC was and is still being executed in software development projects of Atlantic Institute, a software organization assessed at CMMI level 5 in August 2009 that achieved the highest level of maturity of this model. One of the factors that helped to be adherent to CMMI level 5 in relation to the process area Causal Analysis and Resolution, was the implementation of MiniDMAIC in the context of software projects. Four projects were assessed and all of them executed the MiniDMAIC approach for the analysis of causes and no weaknesses were found in any process area from levels 4 and 5 of CMMI during the official assessment.

During the execution of MiniDMAICs in the Atlantic's software development projects, the approach was being refined and better adequate to an analysis of causes more effectively and efficiently in the context of projects. Nevertheless, as the intent of the organization was to continuously improve their processes, the approach is being constantly improved for use in projects.

All the MiniDMAIC steps were implemented in the Jira, a commercial tool for workflow management that can be easily customized. The tool is already used in the organization to issue tracking, and other actions, and made possible to implement actions to causal analysis in projects in a simplest manner. Figure 2 shows the initial screen to create a MiniDMAIC action in the Jira tool.


Step 2 of 2: Enter the details of the issue...

Project:	Teste
Issue Type:	MiniDMAIC
*Summary:	
Description:	
Organizational Performance Baseline:	
Goals:	MiniDMAIC goals
Affected Process:	None Análise e Projeto Análise e Tomada de Decisão Comercial Compras
*Assignee:	- Automatic - Assign to me
Participants:	Start typing to get a list of possible matches.
Due Date:	
Security Level:	None

Create Cancel

Concluído Internet | Modo Protegido: Ativado 100%

Fig. 2. Initial Screen to Create a MiniDMAIC Action in Jira

7.1 Characterization of Organization to MiniDMAIC

Following the practices of the process area Causal Analysis and Resolution of CMMI, some criteria and conditions were defined by the organization to initialize a MiniDMAIC action on projects. The MiniDMAICs could be initialized for analyzing the causes of defects/problems or deviation in the indicators. In the management meetings, the project coordinator should analyze together with the manager the need to carry out a MiniDMAIC to the presented situation. The collaborator responsible for planning and monitoring a MiniDMAIC action should be the project coordinator.

The organization defined the following typical sources of defects/problems:

- Indicators of the project;
- Report of systemic tests;
- Results of integration tests;
- Report from the client tests;
- Problems found in the technical review that affect requirements or the proper execution of the application;
- Customer Complaints.

Moreover, the situations listed below may required the analysis of cause and defect using the MiniDMAIC:

- High value on the systemic tests indicators. For example, indicator above the project goal or out of the specified limits;
- Out of control project, where the results of the indicators with statistically controlled processes do not meet the limits defined by the project or the organizational baseline limits (e.g., productivity, deviation on delivery, defect density, etc.);

- High number of defects classified as critical and blocker in the systemic tests (according to the coordinator analysis);
- High number of defects found by the client (according to the coordinator analysis).
- Defects found in the first project's set of tests;
- Need to analyze the most common types of defects;
- Errors that occur so frequently in the various set of test.

When the analysis of cause and defect is performed, the selection of defects for analysis should consider the following factors:

- Types of the most common defects;
- Frequency of occurrence;
- Similarity between defects.

The organization has a well-defined testing process and to classify the defects should be considered: (i) criticality, (ii) the types of defects and (iii) the sources of defects in relation to the software development life cycle phases.
The level of criticality of the defects was based on the IEEE 1044 (1994) and has the following classification:

- Blocker: failure that causes the block of the main tested functionality or application, preventing the running of the tests. The cases that prevent the execution of other requirements are also considered;
- Critical: failure where the test case steps might be performed, however, they had a disastrous outcome. The cases where a secondary functionality could not be performed successfully are considered;
- Major: failure that has an incorrect results, but do not bring a high impact to the customer;
- Minor: failure in not essential requirements points;
- Trivial: Problems considered cosmetics / accessories that do not affect the functionality of the system.

The organization's types of defects were based on Orthogonal Defect Classification (Chillarege et al., 1992) that comprises the following types of defects:

- Interface;
- Function (functionality);
- Assembling / packaging / integration;
- Attribution;
- Documentation;
- Verification (field validation);
- Algorithm (internal logic);
- Time / serialization / performance.

The defects' sources also were based on Orthogonal Defect Classification (Chillarege et al., 1992), comprising the following sources:

- Requirements;
- A & D – Architecture;
- A & D – Design;
- Implementation;
- Testing.

7.2 Pilot Project Characterization

The organization's software development project selected as a pilot project was considered large, had short iterations and used the Scrum methodology (Schwaber, 2004). This project corresponded to the development of various sub-projects of experimental applications to mobile devices (cell phones). The general characterization of the sub-projects within the context of the project can be seen in Table 9.

Type	Embedded Software
Restrictions	Limited fixed price + Deadline + Flexible Scope
Duration	2 or 3 months. Sprints lasting 4 weeks
Estimate	Story Points + Use Case Points
Team Size	Small (up to 6 employees)
Product Line	Mobile
Stability Requirements	Small (Very volatile requirements)
Customer Engagement	Average
Design Complexity	Large

Table 9. Characterization of the Pilot Project's Sub-projects

In the next sections will show the execution of each phase of MiniDMAIC approach in the selected pilot project.

7.3 Performing the Phases "Define" and "Measure" of MiniDMAIC

All of the organization's software development projects collect and analyze, monthly, the project's indicators in the Project Performance Report. One of the indicators of the organization that has statistically managed processes and subprocesses is the indicator "Defect density".

In the project that MiniDMAIC was executed to this experience report, a great number of defects in systemic tests were identified and it was verified that the values of defect density in systemic tests indicator were above of the organizational baseline limits, as shown in the control chart (Figure 3).

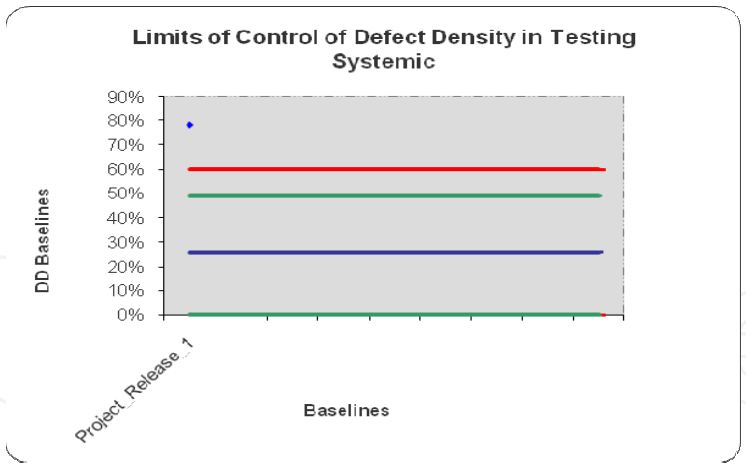


Fig. 3. Project's Control Chart for the Defect Density in Systemic Tests Baseline

Thus, we identified the need to open a MiniDMAIC action for the project in order to analyze the root cause of the project's defects.
The organization has a historical projects base located in a knowledge management tool, accessible to all employees of the organization. This historical base contains: general information from the projects, projects' indicators, lessons learned, risks and MiniDMAICs opened by the projects.
Initially, the organization's historical basis was analyzed to find MiniDMAICs related to the density of defects that have been executed in other projects. There were two MiniDMAICs related to this problem that were considered as a basis for a better execution and analysis of project' causes.
Analyzing the organization's performance baseline of the defect density in systemic tests was defined as the goal of the project, remain within the specified limits of the project (upper and lower target), reducing the density of defects in 81% to achieve the goal of defect density in systemic tests that had been established.
There was no need to identify a new measurement to measure the problem, since the problem was already characterized in the defect density in systemic tests indicator, which was already considered in the projects of the organization and that is statistically controlled.
In a spreadsheet, all defects related to the release's scope were collected and these defects were classified by criticality, source and type of defect, as shown in Figure 4. This classification helps to know the source of the defects according to its classification and to know which are the most recurrent. In the project's context, the largest number of defects was classified as major critical, the source in the implementation and the types of defects were: functionality and algorithm.

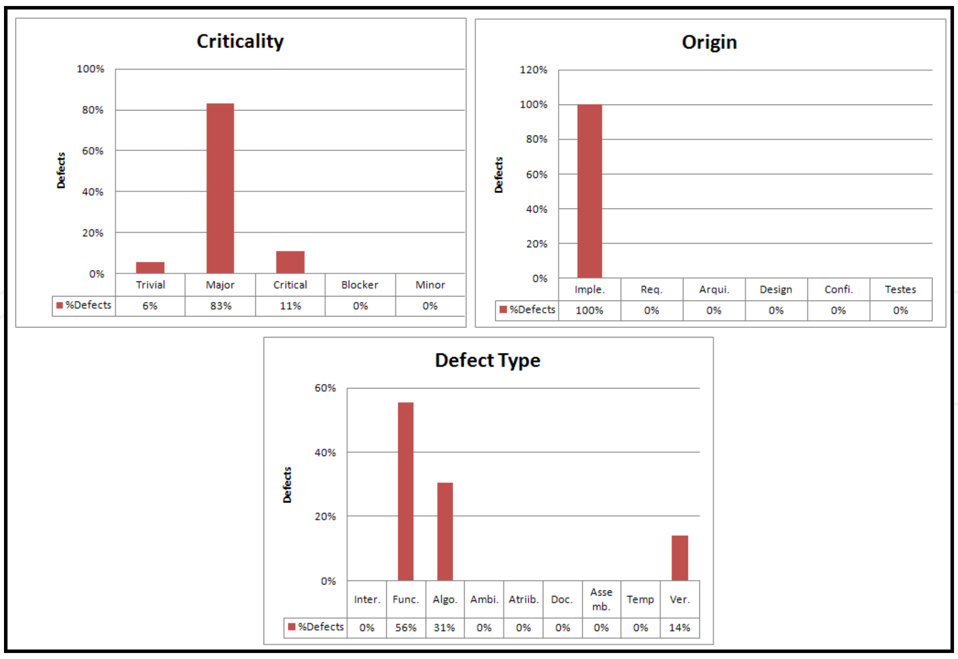


Fig. 4. Classification of the Defects Found in the Project's Systemic Tests

At this phase it was established the following:

- Goal: reduce the defect density in systemic tests in 81%, remaining within the specified limits of the project;
- Affected process (es): Implementation;
- Risks: No risks were identified related to the problem;
- Organizational Performance Baseline: defect density in systemic tests;
- Responsible for the phase: project coordinator, technical leader and Quality Assurance;
- Duration: 1 day.

During the execution of these two phases in parallel, there was only difficulty for classifying the defects, which required a great effort from the team to analyze them.

7.4 Performing the Phase "Analyze" of MiniDMAIC

At this stage, experts were allocated aiming to analyze the defects. In the case of the MiniDMAIC action on the pilot project, were allocated the following specialists: project coordinator, technical leader, Quality Assurance, developers, requirements analyst and test analyst.
Based on the defect classification of the phase "Measure" and grouping of the recurrent defects, a brainstorming meeting was held with the project team in order to find the root cause of defects. The brainstorming was organized in two meetings to identify and prioritize the causes of the problem. At the first meeting, the team had as input the defects collected in the phase "Measure" and their classification, and ideas of possible causes were collected without worrying whether those causes were actually the problem's root causes.
After identifying the causes, each defect were analyzed to know what the causes it was related. So, the most recurrent causes when they were consolidated by defects. Based on that consolidation, a second meeting was held with the project team and shown the consolidated causes to prioritize problem's root causes. The following causes were identified and prioritized by the team, with the help of Pareto charts:

- Cause 1: architectural components developed in parallel with use cases;
- Cause 2: baseline generated without testing in an environment similar to production;
- Cause 3: lack of understanding of requirements by developers;
- Cause 4: Sprint's scope badly estimated (estimation and sequence of the use cases development);
- Cause 5: architecture is not suitable for the concurrent development of the team.

Analyzing the identified and prioritized causes related to the found problems in the iteration was observed that:

- The planning was badly estimated. Many use cases were planned for a short time (fixed time of 4 weeks). Aiming to achieve the scope defined for the iteration, some activities essential to the quality of the final product were not performed in accordance to the planned estimation. Among them, the integration test and the testing on mobile device can be cited;
- The team did not have a full knowledge of the project requirements. It was the first sprint of the project and meetings or workshop were not held with the developers for sharing and discussing the requirements. The artifacts to define the requirements were defined, but they were not followed;
- The initial architecture was not mature, resulting in various problems and additional efforts for the development.

Then, a brainstorming was performed at a meeting to identify possible actions for addressing the causes. The following actions were identified:

- Action 1: perform integration tests before systemic tests;
- Action 2: held a requirement workshop for improving the understanding of the use cases by the project team;
- Action 3: carry out use case tests in an environment similar to the production environment;
- Action 4: define and communicate the concept of "done" to complete the implementation of the use case;
- Action 5: improve the planning to the next iterations, with the participation of the team (the planning should include the development and integration of architectural components before the development of the use cases);
- Action 6: perform the refactoring of architectural components.

In Table 10 we can observe the relationship between the identified causes and the prioritized actions for their treatment.

Causes	Action
Cause 1	Action 1, Action 3, Action 4
Cause 2	Action 1, Action 3, Action 4
Cause 3	Action 2
Cause 4	Action 5
Cause 5	Action 6

Table 10. Relationship Between the Causes and Actions Identified to Address the Defects' Causes

The phase "Analyze" of MiniDMAIC on the project was very detailed and all defects found to improve the effectiveness of the action were analyzed. In addition, we focus in the defects' root causes in order to do address wrong causes. The phase lasted two days. Nevertheless, the project team has difficult to understand what really was the defects' root cause, requiring the support of the Quality Assurance to guide the team and to focus on the causes of the problem.

7.5 Performing the Phase "Improve" of MiniDMAIC

All actions identified in the brainstorming were considered important to be implemented and were easy to implement. An action plan to implement the actions was defined on Jira and each action was inserted in MiniDMAIC action in the Jira MiniDMAIC as a sub-task of MiniDMAIC. For each action were assigned responsible to execute the action and defined a deadline to the action within the project. At this phase, all experts assigned on the phase "Analyze" played a role. Below are described the execution of the actions:

- Action 1: The team performed the integration tests in the sprints 2 and 3 before the systemic test. It was found that the development team identified virtually the same amount of problems that the systemic test team, proving the effectiveness of action.;
- Action 2: A requirements workshop was held in sprints 2 and 3 with the participation of requirements, IHC, testing and development teams. During the implementation of the action the understanding of the requirements was transferred by the requirements team for the rest of the team. The practice contributed a lot for leveling the understanding of the requirements and necessary changes in the requirements that had not previously been thought were highlighted;
- Action 3: In the first execution of this action there was an impediment. Because the use case tests had not been executed in an environment similar to the production environment, we found a bug that prevented the test. Moreover, some test team's members did not have mobile phones to execute the tests, which limited the execution of the action. The error that prevented the test was corrected and the use case tests began to be executed in sprints 2 and 3;
- Action 4: In the planning meeting of project's sprint 2, the concept of "done" has been defined together with the team and shared to all, through minutes and posters attached in the project's room. This practice was used during sprints 2 and 3. The concepts of "done" that were defined:
  - Requirements: use cases completed and reviewed with adjustments.
  - Analysis and Design: class diagram completed and reviewed with adjustments.
  - Coding: code generated and reviewed with adjustments and unit tests coded and documents with 75% of coverage.
- Action 5: Improve the planning of the next iterations with the participation of the team (the planning should include the development and integration of architectural components before the development of use cases). The planning improvements started in sprint 2 of the project. For this sprint was held a planning meeting with the project team, that was recorded in the minutes. In the planning, the development and integration of architectural components were planned to begin before the development of use cases. Furthermore, both the use cases refactoring activities as the activities for

understanding the implemented requirements in accordance with Action 3 were planned to be held initially. During the sprint 3, the same action was performed again;

- Action 6: this action was planned in the execution of Action 5 and the architectural component refactoring was performed by the project team, improving the application's maintainability.

The team had difficulty in deploying the action 3 due to the unavailability of an environment identical to the production environment for the whole team. The other actions were implemented more easily by the project team. On average, the implementation of the actions lasted two weeks.

7.6 Performing Phase "Control" of MiniDMAIC

After the implementation of the actions for addressing the causes of defects, the results were measured to analyze the achieved degree of effectiveness. In the project's second sprint the result was measured and we identified 38% of improvement in the systemic tests defect density indicator and that the result satisfied the project's limits. Nevertheless, the established of 81% was not achieved. So we decided to execute the phase "Improvement", implementing the same actions in the sprint 3, and measuring the results again to verify if the actions actually eliminated the root causes of defects.
In the sprint 3 was measured again the defect density in systemic tests indicator and was found a greater improvement, coming very close to the target defined to the project. Despite the goal was not achieved in sprint 3, the expected results were considered satisfactory and we could observe in two later sprints of the projects that the causes of defects were actually addressed. The improvement in the third sprint was 51%. The Figure 5 shows a control chart illustrating the improvement achieved by the project over the sprints.

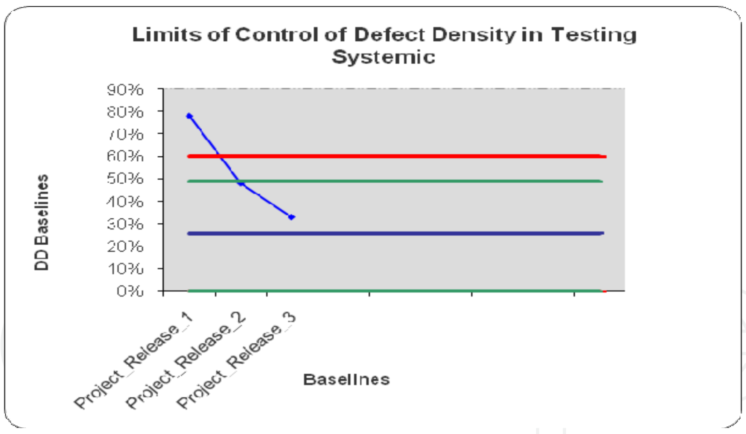


Fig. 5. Project's Control Chart for Defect Density in Systemic Tests Baseline with Final Results after the Execution of the MiniDMAIC Action

After the evidence of the implemented improvements, a meeting was held with the team to collect lessons learned and to close the action with the collected results. As the main lesson learned from the execution of cause analysis on the project, it was observed the importance, in the first sprint, to establish a minimum scope that would allow the architecture development and the knowledge of the team about application's business domain that was being developed.

After closing the action, the project coordinator sent the entire MiniDMAIC action execution's input for the organization's historical basis, through an action in Jira.
Due to the project has being returned to the phase "Improve" to perform the actions in project's sprint 3, the MiniDMAIC on the project had a longer duration, approximately 6 weeks. The strategy of re-performing the phase "Improve" on the next sprint of the project was chosen by the team to check if the actions were really effective and to eliminate the problem's root causes. If the project had obtained, actually, an improvement at the first moment, the duration of the MiniDMAIC action would be, on average, from two to three weeks.

7.7 Providing Improvement Opportunities for the Organizational Assets

All organization's MiniDMAIC actions are reviewed and consolidated by the process group and measurement and analysis group of the organization. The Jira tool generates a document, in Word format, for every execution of MiniDMAIC action that is sent to the historical basis by the project and published in a knowledge management tool, becoming able to be searched by all organization's projects.
To facilitate the monitoring of all MiniDMAIC actions by the process group, some information considered most important are consolidated into a spreadsheet. Table 11 presents the consolidated information including the MiniDMAIC executed on the project illustrated in this work.

Type of Problem	Problem's Causes	Actions Executed for Addressing the Cause	Achieved Improvement
High Defect Density in Systemic Tests	- Cause 1: architectural components developed in parallel with use cases. - Cause 2: baseline generated without testing in an environment similar to production environment. - Cause 3: lack of understanding of requirements by developers. - Cause 4: Sprint's scope badly estimated (estimation and sequence of use cases development). - Cause 5: architecture is not suitable for the concurrent development of the team.	- Action 1: perform integration tests before systemic tests. - Action 2: held a requirement workshop for improving the understanding of the use cases by the project team. - Action 3: carry out use case tests in an environment similar to the production environment. - Action 4: define and communicate the concept of "done" to complete the implementation of the use case. - Action 5: improve the planning to the next iterations, with the participation of the team (the planning should include the development and integration of architectural components before the development of the use cases). - Action 6: perform the refactoring of architectural components.	**Defect density reduction in 51%**

Table 11. Consolidated Information from MiniDMAICs

7.8 Benefits of the MiniDMAIC Approach

Some of the main benefits identified during the execution of MiniDMAIC actions in software development projects were:

- The execution of MiniDMAIC in the organization, reduced considerably, on the projects context, the defect density in systemic tests, as reported in Bezerra (2009b) and increased the productivity as described in Bezerra (2009a);
- The classification of defects used on the approach and adapted by the organization was essential for helping the projects to understanding the defects and to identify of root causes;
- The analysis of many MiniDMAIC is fundamental to identify improvement opportunities for the processes at the organizational level. Thus, we observed that, according to the organization's maturity level, new data sources can aggregate greatly

to the processes improvements. These new sources can be added to the list of data that can be analyzed, defined in Albuquerque (2008);
- The approach implemented in the Jira tool facilitated the use and increased the speed of MiniDMAIC execution, because this tool already contains all the required fields to perform each phase;
- Intensifying the use of the action in the projects an improvement was implemented, the execution of MiniDMAIC in the first set of tests of the projects to analyze the causes of defects. If the project has none actions to be executed to address the defects, the MiniDMAIC could be completed in phase "Analyze";
- The template for analyzing the causes of defects in systemic tests, available from the approach, was of great importance in facilitating the process of analysis and prioritization of the problem's root causes addressed in the projects;
- Integration of MiniDMAIC approach to the processes that deal with identifying and implementing process improvements at the organizational level.

8. Related Works

According to Kalinowski (2009), the first approach to analysis of causes found was described by Endres (1975), in IBM. This approach deals with individual analysis of software defects so that they can be categorized and their causes identified, allowing taking actions to prevent its occurrence in future projects, or at least ensuring its detection in these projects. The analysis of defects in this approach occurs occasionally, as well as corrective actions.

The technique RCA (Root Cause Analysis) (Ammerman, 1998), which is one of the techniques used to analyze the root cause of a problem, aims at formulating recommendations to eliminate or reduce the incidence of the most recurrent errors and hose with higher cost in organization's software development projects. According to Robitaille (2004), the RCA has the purpose of investigating the factors that are not so visible that has contributed to the identification of nonconformities or potential problems.

Triz (Altshuller, 1999) is another methodology developed for analysing causes. It is a systematic human-oriented approach and based on knowledge. His theory defines the problems where the solution raises new problems.

Card (2005) presents an approach for causal analysis of defects that is summarized in six steps: (i) select a sample of the defects, (ii) classify the selected defects, (iii) identify systematic errors, (iv) identify the main causes (V) develop action items, and (vi) record the results of the causal analysis meeting.

Kalinovski (2009) also describes an approach called DBPI (Defect Based Process Improvement), and is based on a rich systematic review for elaboration of the approach to organizational analysis of causes.

Gonçalves (2008b) proposes a causal analysis approach, developed based on the PDCA method, that applies the multicriteria decision support methodology, aiming to assist the analysis of causes form complex problems in the context of software organizations.

ISO / IEC 12207 (2008) describes a framework for problem-solving process to analyze and solve problems (including nonconformances) of any nature or source, that are discovered during the execution of the development, operation, maintenance or other processes.

Most of the research cited in this work proposes approaches for analysis of causes focusing on the organizational level. However, it is often necessary to perform analysis of causes within the projects that must be quick and effective. In organizations seeking high levels of maturity models of process improvement like CMMI, this practice has to be executed within the project to maintain the adherence to the model. Furthermore, from the investigated approaches involving analysis and resolution of causes, none is based on DMAIC method. The approach presented in this work has the main difference from other approaches the focus of causal analysis in the context of projects, providing a structured set of steps based on the DMAIC method, that are simple to execute.

9. Conclusion

The treatment of problems and defects found in software projects is still deficient in most organizations. The analysis, commonly, do not focus sufficiently on the problem and its possible sources, leading to wrong decisions, which will ultimately not solve the problem. It is also difficult to implement a causal analysis and resolution process (CAR) in projects, as prescribed by the CMMI level 5, due to limited resources which they have to work.

The approach presented in the work aims to minimize these difficulties by proposing a consistent approach to analysis and resolution of causes based on the DMAIC method, that is already consolidated in the market. This proposed approach is also adherent to the process area Causal Analysis and Resolution – CAR of CMMI. Moreover, the approach was implemented in a workflow tool, and has been executed in several software development projects in an organization assessed in level 5 of CMMI.

As the main limitation of the approach we have that the MiniDMAIC was defined in the context of organizations that are at least level 4 of CMMI maturity model, since the MiniDMAIC actions will have even better results, because several parameters to measure the projects' results will be already defined, and the use of statistical analysis tools will already be a common practice in the organization. However, it can be executed in less mature organizations, adapting the approach to the organization's reality, but some steps may not get the expected results.

10. References

Albuquerque, A. B. (2008). Evaluation and improvement of Organizational Processes Assets in Software Development Environment. D.Sc. Thesis, COPPE/UFRJ, Rio de Janeiro, RJ, Brazil.

Altshuller, G. (1999). Innovation Algorithm: TRIZ, systematic innovation and technical creativity. Technical Innovation Ctr.

Ammerman, M. (1998). The Root Cause Analysis Handbook: A Simplified Approach to Identifying, Correcting, and Reporting Workplace Errors. Productivity Press.

Banas Qualidade. (2007). "Continuous improvement – Soluctions to Problems", Quality News. São Paulo. Accessible in http://www.estatbrasil.com.br/PgQtN20030003.htm. Acessed in: 2007, Feb 22.

Bezerra, C. I. M.; Coelho, C.; Gonçalves, F. M.; Giovano, C.; Albuquerque, A. B. (2009a). MiniDMAIC: An Approach to Causal Analysis and Resolution in Software Development Projects. VIII Brazilian Simposium on Software Quality, Ouro Preto. Proceedings of the VIII Brazilian Simposium on Software Quality.

Bezerra, C. I. M. (2009b). MiniDMAIC: An Approach to Causal Analysis and Resolution in Software Development Projects. Master Dissertation, University of Fortaleza (UNIFOR), Fortaleza, Ceará, Brazil.

Blauth, Regis. (2003). Six Sigma: a strategy for improving results, FAE Business Journal, nº 5.

Card, D. N. (2005). Defect Analysis: Basic Techniques for Management and Learning, Advances in Computers 65.

Chillarege, R. et al. (1992). Orthogonal Defect Classification: a Concept for in-Process Measurements. IEEE Transactions on SE, v.18, n. 11, pp 943-956.

Chrissis, Mary B.; Konrad, Mike; Shrum, Sandy. (2006). CMMI: Guidelines for Process Integration and Product Improvement, 2nd edition, Boston, Addison Wesley.

Dennis, M. (1994). The Chaos Study, The Standish Group International.

Endres, A. (1975). An Analysis of Errors and Their Causes in Systems Programs, IEEE Transactions on Software Engineering, SE-1, 2, June 1975, pp. 140-149.

Gonçalves, F., Bezerra, C., Belchior, A., Coelho, C., Pires, C. (2008a). Implementing Causal Analysis and Resolution in Software Development Projects: The MiniDMAIC Approach, 19th Australian Conference on Software Engineering, pp. 112-119.

Gonçalves, F. (2008b) An Approach to Causal Analysis and Resolution of Problems Using Multicriteria. Master Dissertation, University of Fortaleza (UNIFOR), Fortaleza, Ceará, Brazil.

IEEE standard classification for software anomalies (1944). IEEE Std 1044-1993. 2 Jun 1994.

ISO/IEC 12207:1995/Amd 2:2008, (2008). Information Technology - Software Life Cycle Process, Amendment 2. Genebra: ISO.

Ishikawa, K. (1985). What is Total Quality Control? The Japanese Way. Prentice Hall.

Juran, J. M. (1991). Qualtiy Control, Handbook. J. M. Juran, Frank M. Gryna - São Paulo - Makron, McGraw-Hill.

Kalinowski, M. (2009) "DBPI: Approach to Prevent Defects in Software to Support the Improvement in Processes and Organizational Learning". Qualifying Exam, COPPE/UFRJ, Rio de Janeiro, RJ, Brazil.

Kulpa, Margaret K.; Johnson, Kent A. (2003). Interpreting the CMMI: a process improvent approach. Florida, Auerbach.

Pande, S. (2001). Six Sigma Strategy: how the GE, the Motorola and others big comnpanies are sharpening their performance. Rio de Janeiro, Qualitymark.

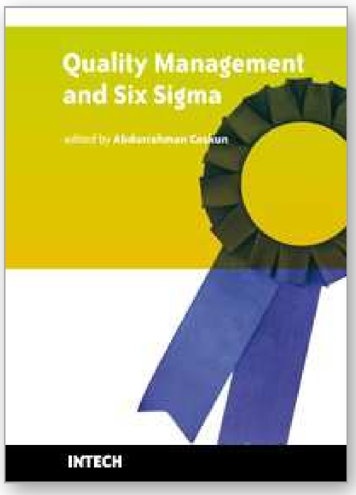
Rath and Strong. (2005). Six Sigma/DMAIC Road Map, 2nd edition.

Robitaille, D. (2004). Root Cause Analysis: Basic Tools and Techniques. Chico, CA: Paton Press.

Rotondaro, G. R; Ramos, A. W.; Ribeiro, C. O.; Miyake, D. I.; Nakano, D.; Laurindo, F. J. B; Ho, L. L.; Carvalho, M. M.; Braz, A. A.; Balestrassi, P. P. (2002). Six Sigma: Management Strategy for Improving Processes, Products and Services, São Paulo, Atlas.

Smith, B.; Adams, E. (2000). LeanSigma: advanced quality, Proc. 54th Annual Quality Congress of the American Society for Quality, Indianapolis, Indiana.

Siviy, J. M.; Penn, L. M.; Happer, E. (2005). Relationship Between CMMI and Six Sigma. Techical Note, CMU / SEI -2005-TN-005.
Tayntor, Christine B. (2003). Six Sigma Software Development, Flórida, Auerbach.
Watson, G. H. (2001). Cycles of learning: observations of Jack Welch, ASQ Publication.



Quality Management and Six Sigma
Edited by Abdurrahman Coskun

ISBN 978-953-307-130-5
Hard cover, 276 pages
Publisher Sciyo
Published online 16, August, 2010
Published in print edition August, 2010

If you do not measure, you do not know, and if you do not know, you cannot manage. Modern Quality Management and Six Sigma shows us how to measure and, consequently, how to manage the companies in business and industries. Six Sigma provides principles and tools that can be applied to any process as a means used to measure defects and/or error rates. In the new millennium thousands of people work in various companies that use Modern Quality Management and Six Sigma to reduce the cost of products and eliminate the defects. This book provides the necessary guidance for selecting, performing and evaluating various procedures of Quality Management and particularly Six Sigma. In the book you will see how to use data, i.e. plot, interpret and validate it for Six Sigma projects in business, industry and even in medical laboratories.

How to reference
In order to correctly reference this scholarly work, feel free to copy and paste the following:

Carla Ilane Moreira Bezerra Carla, Adriano Bessa Albuquerque Adriano, Luiz Sergio Placido Sergio and Marcia G. S. Goncalves Marcia (2010). MiniDMAIC: an Approach to Causal Analysis and Resolution in Software Development Projects, Quality Management and Six Sigma, Abdurrahman Coskun (Ed.), ISBN: 978-953-307-130-5, InTech, Available from: http://www.intechopen.com/books/quality-management-and-six-sigma/minidmaic-an-approach-to-causal-analysis-and-resolution-in-software-development-projects