PROCESS IMPROVEMENTS FOR ESTABLISHING CMMI CERTIFICATION AT COMPANY XYZ

A Senior Project submitted

in partial fulfillment

of the requirements for the degree of

Bachelor of Science in Industrial Engineering

California Polytechnic State University

San Luis Obispo

By

Megan Farrell

December 2012

Graded by: _____ Date of Submission:_____

Checked by:_____ Approved by:_____

Abstract

CMMI stands for Capability Maturity Model Integration and is an enterprise-wide Process-Improvement model that provides a set of best practices that addresses productivity, performance, costs, and stakeholder satisfaction. This improvement model is generally used in computer software companied and not only does it improve their processes, but many contracts will only be awarded with those companies who are CMMI certified, so it is very important for Company XYZ. This goal of this project was to help improve the CMMI efforts at Company XYZ in 4 of the 20 distinct processes areas to attain a CMMI Maturity Level of 3. Improvement to these processes was done so through analyzing the current processes, surveys, management suggestions, and successful CMMI suggestions. In the end, Requirement Definition and Requirements Management were achieved through creating standardization, visualization, and traceability to these areas. The verification and validation process areas were also achieved through improving the testing software division in various ways including better visualization. CMMI is a model to optimize the development activity in every stage and once accomplished, Company XYZ is expected to see improvements in quality, growth increase, cost variance, delivery time, productivity, and customer satisfaction.

Table of Contents

List of Tables

List of Figures

Introduction

Company XYZ is a computer programming company in which most of its contracts come from the government, specifically the Department of Defense. With the competition rapidly growing of computer software companies, it is essential that Company XYZ becomes competitively advantageous. One of the ways to do this is to become CMMI Certified. CMMI is the abbreviation of Capability Maturity Model Integration and CMMI certification is a specific software product quality management and quality assurance standards. By becoming CMMI Certified, not only is the company implementing a product improvement initiative for higher quality products, but it is enabling itself to be awarded more contracts in the future, ones that could only be awarded with this certification.

Becoming CMMI Certified is a long process and is far beyond the level of complexity and scope for a senior project. The initial stages of becoming certified, which is what Company XYZ is currently in, contains a lot of documentation and widespread company participation. The focus of this project is to assist in fulfilling the initial stages of this project and to help spread awareness to the company about what this project really is and how it will affect the different areas of the organization. The deliverables and objectives for this project are to:

- Define measurable and quantifiable metrics for all departments within the company
- Formulate suggestions and improvements to the current computer development cycle in order to improve those metrics
- Create a brief document (5-10 pages) describing and explaining the actions of the CMMI Certification Process in all areas of the company and how this new process will affect each area.
- Construct an economic justification of implementing the CMMI process changes and getting CMMI Certified.

In order to complete the above objectives the following courses will be used: Project Organization and Management, Manufacturing Organizations, Quality Engineering, Engineering Economics and Process Improvement Fundamentals. The organization of this report will begin with the literature review about what CMMI is and how to integrate it with an Agile Software Organization. The literature review will then be followed by the design portion of the report then followed by the methodology to increase improvements within the organization. Methodology will then be followed by the results and economic justification and immediately followed by the conclusion. The report will then be completed with the addition of the Bibliography and Appendixes.

Background and Literature Review

CMMI, or Capability Maturity Model Integration, is a process improvement approach that provides organizations with the essential elements of effective processes, which will improve their performance. CMMI-based process improvement includes identifying your organization's process strengths and weaknesses and making process changes to turn weaknesses into strengths [1]. Integrating CMMI best practices into an agile development organization, may seem contradictory, however research from SEI Learn suggests just the opposite [2]. Agile's values and practices ensure critical, long-term process success, which can make it an ideal partner of the CMMI framework, which delivers a robust infrastructure of organization-wide, broadly inculcated continuous improvement and optimization [3]. It has been argued that CMMI left out some of the basic elements critical to long-term process success that Agile values and practices supply [4]. Together, Agile and CMMI complete each other's capabilities and can lead to fast, affordable, visible, and long-term benefits [3].

Capability Maturity Model Background

The Software Engineering Institute (SEU) is a research and development center sponsored by the U.S. Department of Defense and operated by the Carnegie Mellon University [5]. CMMI is a framework, or process improvement approach, which shows the evolutionary path covering the entire product life cycle from requirements definition through delivery and maintenance, providing organizations with the essential elements of effective processes [6]. It can be used to guide process improvement across a project, a division, or an entire organization. CMMI helps integrate traditionally separate organizational functions, set process improvement goals and priorities, provide guidance for quality processes, and provide a point of reference for appraising current processes [7].

The CMMI path (staged representation) defines five levels of process maturity. Each maturity level builds upon key elements, called Process Areas, of the previous level, as seen in Table 1 [1]. These maturity levels can also be commonly referred to Initial, Repeatable, Defined, Managed, and Optimizing, each a layer in the foundation for ongoing process improvement [8]. At maturity level 1, processes are usually ad hoc and chaotic, and usually don't provide a stable environment. At this level organizations often produce products and services that work; however, they frequently exceed the budget and schedule of their projects. At maturity level 2, the projects of the organization have ensured that requirements are managed and that processes are planned, performed, measured, and controlled; whereas at maturity level 3, processes are well characterized and understood, and are described in standards, procedures, tools, and methods, and usually possess an integrated software management, focus, and coordination. At maturity level 4, organizations are controlled using statistical and other quantitative techniques possessing quantitative process management; and finally at maturity level 5, an organization has achieved all the specific goals of the process areas assigned to maturity levels 2, 3, 4, and 5 and processes are continually improved based on a quantitative understanding of the common causes of variation inherent in processes.

Table 1: Process Areas by Maturity Level (staged representation)

Level	Process Area
Level 1	(not applicable)
Level 2	Configuration Management Measurement and Analysis Product Monitoring and Control Project Planning Process and Product Quality Assurance Requirements Management Supplier Agreement Management
Level 3	Decision Analysis and Resolution Integrated Teaming Integrated Product Management Organizational Environment for Integration Organizational Process Definition Organizational Process Focus Organizational Training Product Integration Requirements Development Risk Management Technical Solution Validation Verification
Level 4	Organizational Process Performance Quantitative Project Management
Level 5	Causal Analysis and Resolution Organizational Innovation and Deployment

Organizations are assessed and rated according to the level for which they satisfy all the Process Areas for that level and the levels leading up to that level. For example, a level 3 organization satisfies all Process Areas for level 3 in addition to levels one and two. An organization cannot become level 3 without first satisfying the Process Areas for level one and two [9]. Figure 1 shows the percentage of levels attained by organizations assessed against the CMM criteria for software. Note that only about 7% of the assessed organizations attain a level 4 or 5 [8].

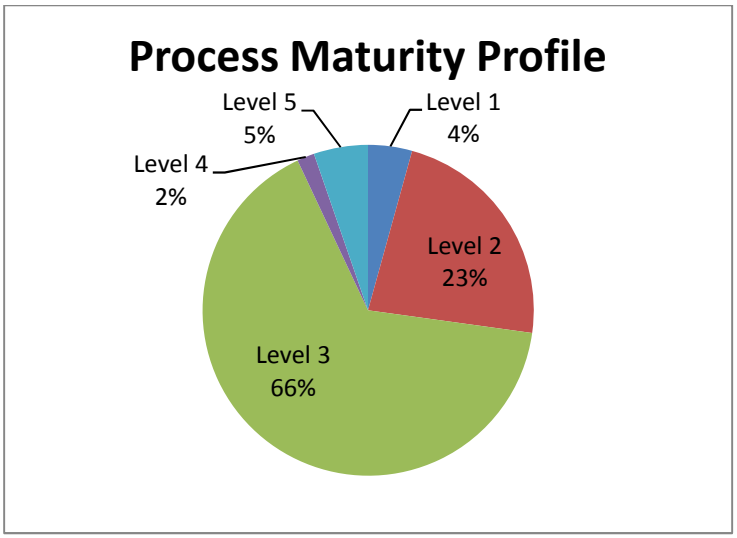


Figure 1: Process Maturity Profile of the Software Community 2011 [8]

CMMI is a continuously evolving application of process management and quality improvement concepts. It is based on a commonsense and broad-based consensus of the product and service development communities. The goal of CMMI is to promote consistent and predictable engineering and management practices. As process capabilities evolve, organizations are able to reduce development cycle times, eliminate defects, predict process performance, and adapt products and services to better meet the needs of their end customer. Therefore, organizations can better plan and manage their development and maintenance while engineering products to better meet specifications.

CMMI techniques apply to teams, work groups, projects, divisions, and entire organizations. CMMI models, moreover, are simply collections of best practices that help organizations to dramatically improve effectiveness, efficiency, and quality. These products, or CMMI solutions, consist of practices. Practices cover topics that include causal analysis; configuration management; quality assurance; verification and validation; risk management; requirements management; supplier management; project management; interface compatibility; make, buy, or reuse analysis; capacity management; availability management; disaster recovery, data collection, process performance; and many more [1].

In addition, organizations should not confuse CMMI goals with business goals. Achieving CMMI maturity does not guarantee an organization will achieve their strategic business goals. However, CMMI provides a powerful tool to move them in the right direction. Successfully adopting CMMI has shown that it can yield a solid, high-return investment for an organization as it ensures long-term enduring results. The business benefits experienced by organizations using CMMI in their process improvement programs include, but are not limited to, improved customer satisfaction, increased quality, more accurate, schedules, lower development costs, substantial return on investment, improved employee morale, and reduced turnover [10].

CMMI generally applies to the same process management concepts as Six-Sigma and Total Quality Management (TQM). Six-sigma is a business initiative which seeks to nearly eliminate product defects, thus improving customer satisfaction. Technically, six-sigma means having no more than 3.4 defects per million opportunities with a product or service [11]. TQM is a tool used to better meet customer needs by improving the supplier inputs and the process of an organization. CMMI, Six-Sigma, and TQM all recognize the disciplines of statistical problem solving and quality tools. Utilizing the CMMI Level 4 statistical techniques in software applications is very important as it aids in predicting the number of production defects based on the current and previous months' defects discovered in production, unit testing, system testing, and regression testing, as well as the cross-sectional project factors. However, the approach may just as easily be used to predict other measures of quality, assess the quality impacts of any evolution activity of potential concern, and account for other recognized cross-sectional variations, such as among teams, geographical locations, methods, or tools [12].

Integrating CMMI into an Agile Software Organization

There have been many misunderstandings about the effectiveness and possibility of integrating and using both CMMI and Agile practices, however, both Agile and CMMI together have shown to

benefit project and organizational performance [3]. At the project level, CMMI focuses at a high level of abstraction of *what* projects do, not on what development methodology is used, while Agile methods focus on *how* projects develop products [13]. There can be much value gained from Agile and CMMI synergies as many CMMI-adopting organizations have Agile development teams. Conversely, CMMI can be effectively introduced in an Agile setting where an iterative, time-boxed approach is used, which is perfectly compatible with CMMI [13].

CMMI and Agile can complement each other by creating synergies that benefit the organization using them. Agile methods provide software development how-to's that are missing from CMMI best practices that work well, especially with small, co-located project teams [14]. CMMI provides the systems engineering practices that help enable an Agile approach on large projects. CMMI also provides the process management and support practices that help deploy, sustain, and continuously improve the development of an Agile approach in any organization and integrating CMMI and Agile Development can be divided into five parts [15].

- Part One begins with concise primers and refreshers on both CMMI and Agile, explaining why they are far more compatible than many practitioners realize [15].
- Part Two introduces specific, proven techniques to help CMMI "process-mature" organizations increase their agility [15].
- Part Three demonstrates how successful Agile organizations can increase their CMMI process maturity without compromising the agility that has brought them success [15].
- Part Four shows how the CMMI can help organizations that are attempting to be agile, but are missing key ingredients of true agility [15].
- Part Five introduces exclusive CMMI/Agile-based techniques for achieving substantial performance gains by focusing on the "personal" side of process improvement [15].

Although integrating CMMI and Agile together can operate synergistically to enhance the other's performance, speed to deployment, and acculturation, organizations should be well advised and educated about integrating both practices [3]. According to a survey ran in 2008 exploring the success rate and effectiveness various process frameworks, the success rate for CMMI-compliant traditional projects was 57.6% while the CMMI-compliant Agile projects was 53.4% [16]. In addition, the success rate for non-CMMI traditional projects was 54.8%, while the success rate for non-CMMI Agile projects was 56.8% [16]. The success rates among the various process frameworks are surprisingly close, perhaps due to the misunderstandings or misuses or these processes, and further investigation needs to be performed in order to conclude the true effectiveness of the integration of both CMMI and Agile.

Design and Methodology

Overall Approach

Due to the enormous complexity and time required to establish a CMMI Level 3 certification from a CMMI level of 1, the overall approach for this project was to focus on improvements for only a subset of the 20 Process Areas, detailed in Table 1, associated with doing so. These Process Areas include: Requirements Management, Requirements Development, Verification, and Validation. Both the Requirements Management and Requirements Development process areas fall under the "Requirements" initial stages of a project process design for a computer software company, while Verification and Validation fall under the "Test" latter stages of a project process design, as seen in Figure 2 below. The overall approach of this project was to focus on improving and standardizing the Requirements and Testing Processes in a CMMI way.

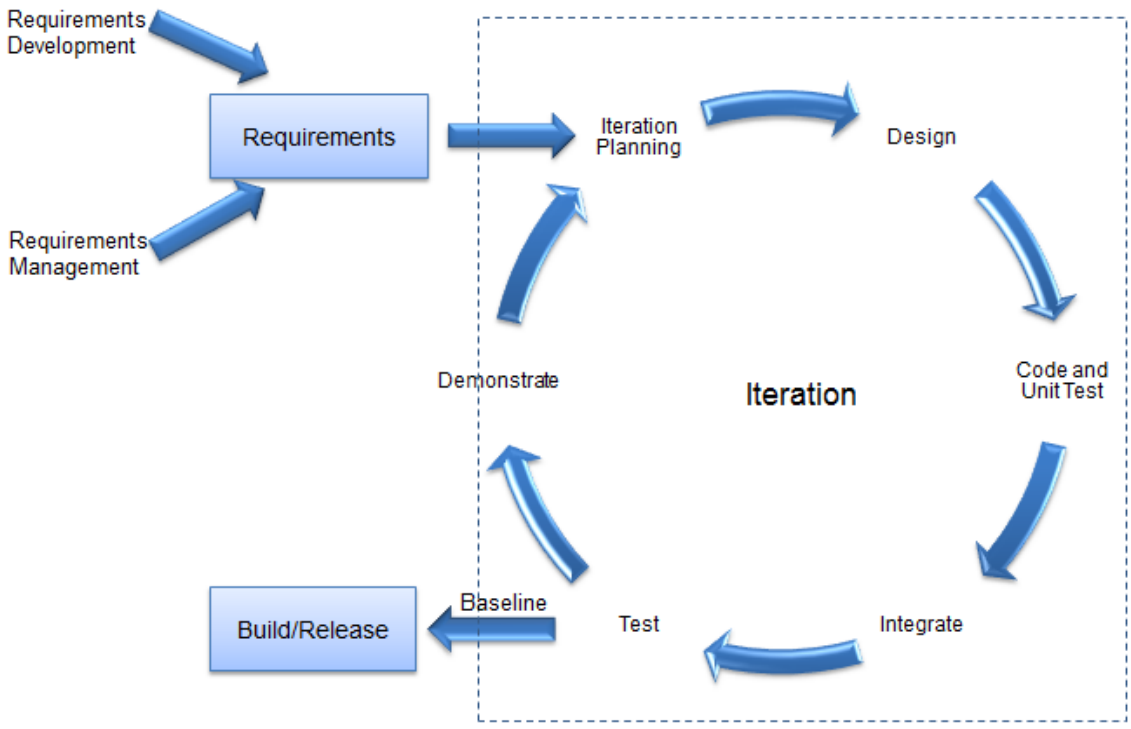


Figure 2: Company XYZ's Iterations and Processes

Requirements Processes

Requirements are the foundation of every software project and for the basis for system and software design, development, verification and operation. It is essential to understand the customer needs and develop a set of requirements early on to review with the customer and manage throughout the system development lifecycle. This stage consists of the Requirements Development and Requirements Management Process Areas and is crucial to develop a strong, robust set of processes to establish a CMMI Certification.

Prior Requirements Workflow Process

As part of the CMMI process, a baseline appraisal is conducted to determine the different states of the process areas the company is currently in. The appraisal for the Requirements Processes was conducted much earlier to the start of this project. The results from the baseline appraisal "requirements" process areas showed that many areas were lacking and needed improvement to attain their respective maturity levels. These results include:

- No formal or standard Requirements Development or Requirements Management process in place.
- Different projects within the company did these processes differently.
- No standard Requirements Management tool in place; Excel was used on some projects.
- Bidirectional requirement traceability not established.
- Traceability of requirements to design/test was not evident.

A lack of standard processes and tools may cause requirements creep and errors which increase customer concerns and project risks [17]. This can be especially troublesome to a computer software company when it is not properly managed, due to the detrimental impact such changes may have on cost, resources, quality, or the ability to deliver a system that incorporates the new requirements on time. While it can be argued that the majority of software applications have unstable requirements and

that some degree of requirements creep is observed in all requirements methodologies, it is very important to implement standard tools and processes to minimize those risks.

Requirements Workflow Re-Design

From the baseline appraisal, it was determined that a company-wide standard tool was needed in order to properly define and manage requirements throughout the company. The selected tool to perform said function is called JIRA. JIRA is currently used by the majority of the company as a tool for managing bugs, tasks and projects, although it has the capabilities for managing requirements and supporting the overall System Development Life Cycle throughout the company. After extensive collaboration, the newly re-designed JIRA now has the capabilities to perform all of the requirements development and management required to become CMMI certified. These new capabilities within JIRA include:

- Eliciting, analyzing, deriving, allocating, and documenting all customer requirements
- Developing lower-level requirements for subsystems and configuration items and trace to source requirements
- Allocating requirements to elements of functional and physical architecture
- Linking requirements to all design and test artifacts
- Managing requirements changes
- Maintaining bidirectional traceability of requirements
- Generating various reports and documents

The process for the newly defined requirements process contains six distinct steps to go from creation to closure. First, new requirements are created or imported from the excel worksheet, traceability to parent/child requirements are established, requirements are approved, and then assigned to versions or sprints. Second, the development team will create features, tasks, and subtasks

for each requirement, all linked to the parent requirement. Third, the test and evaluation team will create and link all test cases to the requirements to ensure all the new requirement functionality is working as intended. Fourth, the development team will code and develop the requirements and mark them as "resolved" when they believe the functionality is fully fulfilled. Fifth, the test and evaluation team fully test the requirement by running through the test cases created in the third step and link any bugs found to the requirement. If any bugs are found, the state of the requirement becomes "re-opened" and the requirements go back and forth between steps four and five until no bugs involving the requirement exist. Not until all bugs have been resolved and the requirement's functionally has been fully developed and tested, does the requirement reach a closed state. Figure 3 shows the requirements traceability within JIRA by illustrating the developed requirement, located in the middle, visibly linked to all design and test artifacts to ensure completion of all customer requirements without errors.

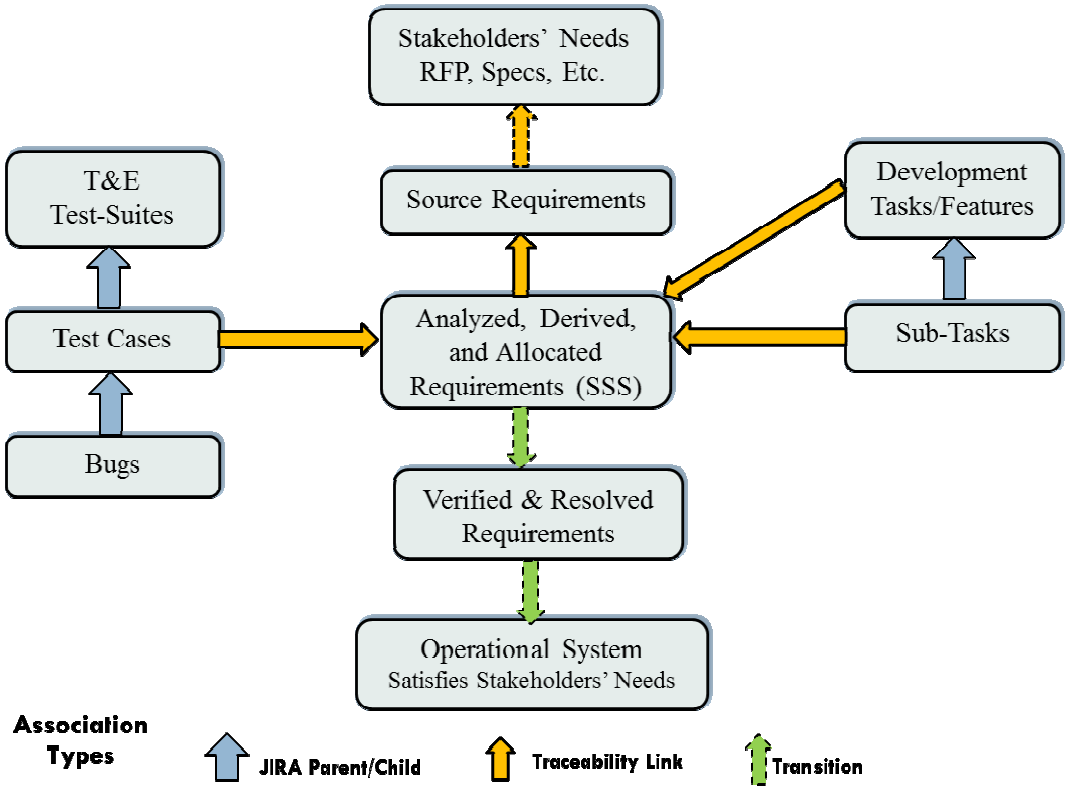


Figure 3: Requirements Traceability within JIRA

In order to use JIRA as a tool to perform these essential functions, a lot of collaboration, back-end coding, creating excel templates, preparation, testing, and support needed to be conducted, created, and maintained. This project included creating the needed excel templates in order to import the requirements according to set of attributes, found in the Appendix A, and establishing links and back-end coding to allow the program to function as desired along with testing all developed functionality to make sure it all works as intended. In addition, a matrix was created to outline the responsibilities for the different groups within the different requirements states for training and clear understanding purposes, which can be found in Appendix A.

Testing Processes

The second main part of the project was to focus on creating improvements to the current testing processes for the overall CMMI improvements efforts. The testing stage in a computer development cycle is one of the final stages of the development process, and is comprised of the Validation and Verification Process Areas. Although commonly overlooked, the testing processes are crucial in establishing and maintaining software quality. The focus of Software Quality Assurance, or testing processes, is to monitor continuously throughout the Software Development Life Cycle to ensure the quality of the delivered products. This requires monitoring both the processes and the products. In process assurance, Software Quality Assurance provides management with objective feedback regarding compliance to approved plans, procedures, standards, and analyses. Product assurance activities focus on the changing level of product quality within each phase of the life cycle, such as the requirements, design, code, and test plan. The objective is to identify and eliminate defects throughout the life cycle as early as possible, thus reducing test and maintenance costs [18].

The key roles this project played in improving the Test Group CMMI efforts include: creating a document and understanding the current testing process that was used for a baseline appraisal,

improving the current Test Group Processes by creating dashboards in the bug and issue tracking software, JIRA, for testers and management in order to create visibility of work performance and tasks needing improvement, and lastly creating a training document along with PowerPoint slides describing the new testing process changes which was used to train all the part-time testers on the new Dashboards and Requirements integration.

Current Testing Process Document

Assessing the current practice is an important step in the process improvement cycle as assessment procedures give organizations a systematic and thorough way to examine and document the current status of their software development process, and is needed for a baseline appraisal to become certified in CMMI. The first step, therefore, was to prepare a detailed written description of the process as it currently is practiced. As part of this project, I helped in the creation of this 30 page current test plan document, detailing every step of the current processes. The template for this document can be found in Appendix B and a brief current testing overview can be seen in Figure 4 below.

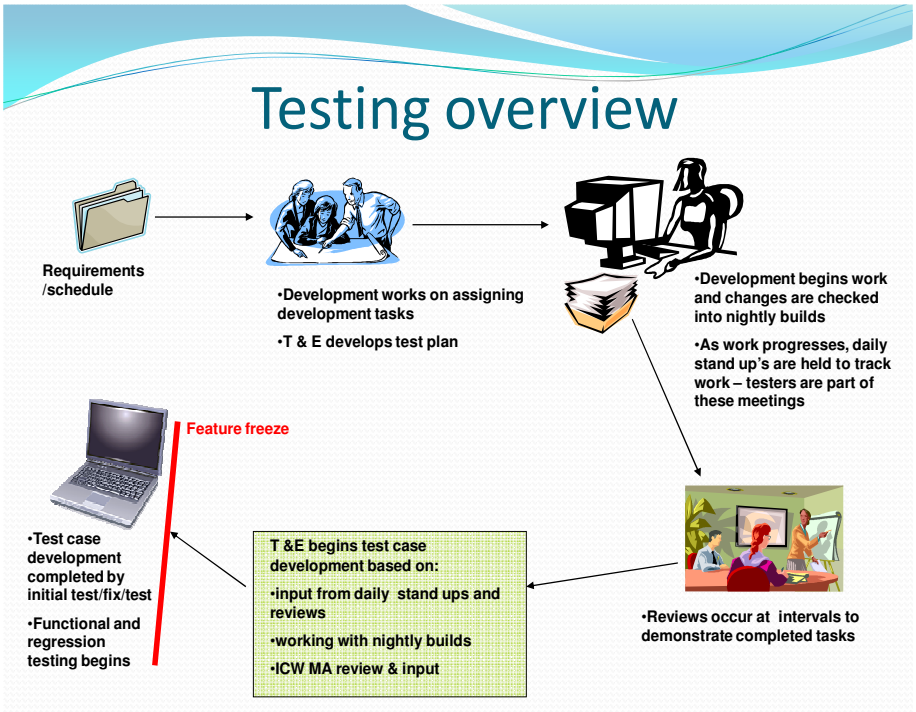


Figure 4: Current Test Workflow

Home and Project dashboard creation in JIRA

Identification of Need

After creating the baseline testing processes document, it is import to describe the desired processes and to constantly generate improvements to that model. To understand what areas of the testing processes needed improvement, a survey was conducted and evaluated for all of both the part-time testers and full-time management. Based on the results of the surveys (seen in Appendix Z), It was discovered that one of the main areas needing improvement was visibility of work performance for management and visibility of tasks for the part-time testers.

Creating the Solution: Dashboards

Better insight and visibility about individual roles and current project status's were determined to be the biggest areas needing improvement. This was done by creating "Home" and "Project-based" dashboards within the current bug and issue tracking software, called JIRA. "Home" Dashboards are based on the user's individual role, tester and management, and contain key elements requested and needed for each role to perform optimally. "Project-based" dashboards show visibility of all key testing metrics and statistics, outlined in the "Test Execution Metrics" in Table 2, which apply to each project. Examples of each Dashboard can be shown in Figure 6 in the following section.

It is not just management that needs to know the status of test planning and preparation to properly gauge the readiness of the test team to test the software. Testers also need to be able to see and work with this information, found in the project-based dashboards and to understand the parts that are relevant to their own work, found in the home dashboards. With adequate information, testers are better able to plan and perform their tasks more efficiently, and to get an accurate picture of the system [19]. While testing is in progress, results should be measured in terms of software functionality tested, degree of code coverage, progress against schedule, and problems found. Usually these factors provide

a much clearer picture of the status of the testing (and the overall quality of the software) than how long the testing has been running.

Compiling and communicating complete information about test activities for each project dashboard provides a common way for everyone on the project to track test activity progress. Testing status needs to be presented to each member of the team in a way that is understandable and that lets them make the best decisions possible. The project manager needs to have a comprehensive view of the progress of test preparation and execution, understanding at a global level what tests are being designed, what tests were run against different versions of the software, what areas of functionality are affected by significant bugs, and what parts of the code will be affected by the bugs that have been found. Testers need to have a different, but similarly detailed view of the system. In short, each person on the project has a different role, and so each needs information that is appropriate to their function, as found through the "Home" dashboards. Table 2 below summarizes commonly used testing metrics and where in the testing process they apply.

Table 2: Commonly Used Testing Metrics [20]

Metric Type	Test Development Metrics	Test Execution Metrics
Functional Metric	• Number of requirements allocated by test • % of requirements by test development phase	• Number of requirements verified • % of requirements tested by version • % of requirements tested by major software component • Stability of server/platform per user
Code Metric	• % of code covered per test • % of code coverage per major software component	• Code coverage of tests completed for each version under test
Problem Metrics	• Problems tested for in regression tests • Extreme conditions tested for in functional tests	• Problems found per version tested • Problems found per software component • Number of critical/high problems found per version
Schedule Metrics	• % completion of functional test requirements by testing phase • Weighted functional requirement completion	• Tests completed per version • Estimated number of days to complete • Test cycle completion time • Time to complete testing per functional area

Revised Testing Process Training

Once improvements have been made to the baseline processes, it is essential to provide adequate training to ensure all designs and improvements work as anticipated. Tester training was created, prepared, and managed to include how the new JIRA Dashboards functionality worked and how the new Requirements within JIRA worked specifically geared to the tester's viewpoint. A PowerPoint Training Presentation was created using the cloud-based presentation software, Prezi, describing these 2 new functionalities and how they pertained to the new test group processes. The training slide for using the new dashboards can be seen in Figure 6, below, while the training slide for understanding the new Requirements processes can be seen in Figure 7 on the next page.

Figure 5: Training Slide for using the new Dashboards

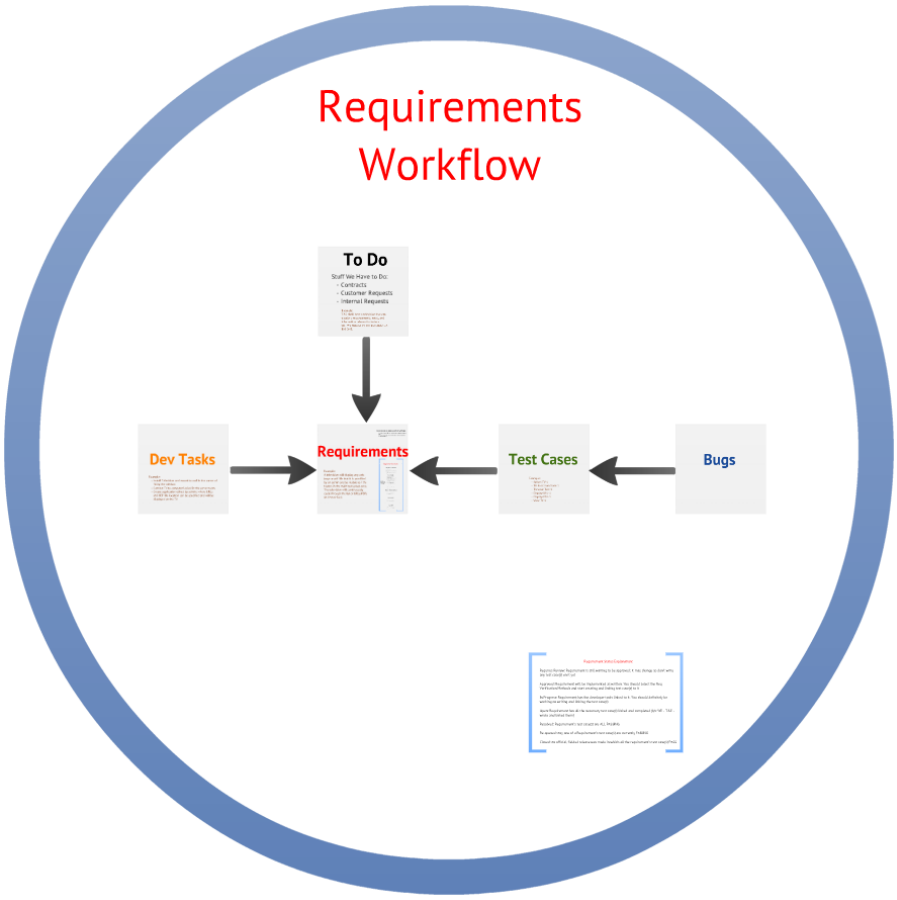


Figure 6: Training Slide for understanding Requirements Processes using Prezi

Once the training sessions ended, a brief test was handed out to gauge the tester understanding of these new processes and functionalities and helped to understand which areas needed to be further explained. Upon completion of all testing training sessions and collection of feedback, the “New Tester Training” documentation was edited to incorporate these new processes. All of the presentation material was composed into 2 training document slides, complete with diagrams and screenshots. Both of these 2 Training guides, without the screenshots, can be found in Appendix C.

Results and Discussion

Although the total CMMI efforts for Company XYZ are only around 75% complete, all of the CMMI efforts for this project have been completed and implemented. The results for the CMMI efforts relating to this project are as follows:

- The newly re-designed JIRA requirements functionality have been put into company-wide use for almost 2 months and with the learning-curve and transition, it is just starting to become beneficial, although it is expected to become much more effective in time.
- All testers have been trained in the new CMMI requirements processes and dashboards.
- In 3 months, there has been a 150% improvement in part-time tester work performance, according to assessment from the test group manager.
- All test group employees are utilizing the new dashboard tools and visuals in some way.

The big picture and main results from these efforts and implementations set forth by this project will become more prominent when the company fully reaches a CMMI Maturity Level of 3. Once accomplished, the company is expected to see a 50% growth increase, 25% improvement in cost variance, more efficient use of personnel, improved on-time delivery, accuracy of estimates, quality, productivity, and customer satisfaction [21].

Cost Analysis

The cost to implement a CMMI maturity level of 3, from a maturity level of 1 varies greatly based on numerous factors. These factors include: the size of the company, the companies starting point, the scope of improvement, the current measurement system, employee and management commitment, etc. The range of costs and benefits in establishing a CMMI certification can vary greatly depending on the company and can be almost impossible to determine without any financial data.

Due to the lack of data or forecasts, the cost analysis in this report is determined based on statistical or median data values to represent a medium sized software company, such as Company XYZ. Internal costs, such as process definition and documentation, process implementation, process adherence, and assessment preparation, can cost anywhere from $500,000 to 2 billion dollars, and External costs, such as: Assessment costs and consulting fees, can cost anywhere from $160,000 to $460,000. This brings the Total Cost to Implement a CMMI process for a medium sized software company to anywhere from $660,000 to almost 2.5 billion dollars.

The estimated savings, on the other hand, can range anywhere from 5.7 billion to 12.2 billion dollars. This, however, is based on the improved processes, and does not take into account the benefit of potentially being awarded more government contracts. Although both the estimated costs and estimated savings are comprised of wide spreads of varying data, most of the research found stated that the median return on investment for establishing a CMMI maturity level 3 certification for a medium-sized software is generally about 4:1. An example of a medium sized software company's costs and savings can be seen in Table 3 and Table 4 respectively.

Table 3: Example CMMI Costs for a medium sized software company [22]

	Total Hours	Wage/hour	Total Amount
CMMI Policies and Procedures	5,862	$ 50	$ 293,100
CMMI Evidence of Use	30,837	$ 50	$ 1,541,850
CMMI Implementation Cost			**$ 1,834,950**
Assessment Preparation Costs	4,800	$ 50	$ 240,000
Assessment Costs	3520	$ 50	$ 176,000
Assessment Fee			$ 64,615
CMMI Assessment Costs			**$ 480,615**
Total CMMI Cost			**$ 2,315,565**

Table 4: Example CMMI Savings for a medium sized software company [22]

	Total Hours	Wage/hour	Total Amount
Maintenance savings	222,932	$ 50	$ 11,146,600
Development savings	20,352	$ 50	$ 1,017,600
Total CMMI Savings			**$12,164,200**

Based on these sample costs and savings figures for a similar medium-sized software company, the Benefit-Cost Ratio and Return on Investment index measures can be calculated. The Benefit-Cost Ratio is simply the benefits divided by the costs which is: $2,513,565 divided by $12,164,200, or 5:1. The Return on Investment, or ROI, is benefits minus costs divided by costs, or 12,164,200 minus 2,315,565 divided by 2,315,565, which is 4.25 or 425%. Although these metrics are specific to the company used in this sample, all figures researched have shown similar results and can give a fairly accurate cost analysis for implementing and establishing a CMMI Maturity Level 3 certification to medium-sized software company.

Summary and Conclusions

Implementing a CMMI maturity level rating of 3 into a medium sized software company is very time and resource consuming, although with enough time and dedication, it is determined that the value of the CMMI Level 3 outweighs the cost of achieving it. With the scope of this project, Company XYZ has successfully improved their computer and testing processes in the following ways:

- Increased standardization and visualization through incorporating requirements into the JIRA workflow
- Increased traceability of requirements to design and test artifacts
- Increased tester work efficiency though visualization of individual-based tasks and project-based tasks

Managing a successful testing effort is very important and requires knowing all about project schedules. As we have seen, treating testing as a black box at the end of the project schedule invites failure. Testing and test preparation need to be open, managed, and manageable processes that everyone on the project team can understand and work with. Having insight into all testing requirements and desired outcomes is an absolute must for effectively managing a successful testing effort. In addition, having current information available for testing is important to prepare and manage the project for testing. Testing progress needs to be measured in terms of schedule, functionality, code coverage, and problems. These four measures of progress need to be available in a form that allows each member of the team to understand the current status of testing, what needs to be done, and how that will affect their part of the project. Using that knowledge, the team can ensure the success of testing and remove one of the major stumbling blocks to a successful project and a successful product.

Bibliography

[1] Carnegie Mellon University, "Software Engineering Institute," 2012. [Online]. Available: http://sei.cmu.edu/cmmi. [Accessed 5 May 2012].

[2] H. Schneider, "CMMI and Agile: Opposites Attract," February 2009. [Online]. Available: http://www.executivebrief.com/cmmi/agile-cmmi/. [Accessed 5 May 2012].

[3] H. Glazer, "Love and Marriage: CMMI and Agile Need Each Other," *CrossTalk: The Journal of Defense Software Engineering,* 2010.

[4] B. Boehm and R. Turner, Balancing Agility and Discipline: A Guide for the Perplexed, Boston: Addison-Wesley Professional, 3002.

[5] Carnegie Mellon University, The Capability Maturity Model: Guidelines For Improving The Software Process, MA: Addison Wesley Longman, Inc., 1994.

[6] L. Zhang and D. Shao, "Software Process Improvement for Small and Medium Organizations Based on CMMI," in *Artificial Intelligence, Management Science and Electronic Commerce*, Jiaozuo, 2011.

[7] C. Shelton, "Agile and CMMI: Better Together," Scrum Alliance, 2008.

[8] Software Engineering Institute Carnegie Mellon University, "CMMI for SCAMPI Class A Appraisal Results 2011 End-Year Update," Pittsburgh, 2012.

[9] M. J. Miller, D. M. Ferrin and F. Pulgar-Vida, "Achieving Higher Levels of CMMI Maturity Using Simulation," in *Simulation Conference, 2002. Proceedings of the Winter*, 2002.

[10] B. P. Gallagher, M. Phillips, K. Richter and S. Shrum, CMMI for Acquisition: Guidelines for Improving the Acquisition of Products and Services, Second Edition, Addison-Wesley Professional, 2011.

[11] P. Buss and N. Ivey, "Dow Chemical Design for Six Sigma Rail Delivery Prohect," in *Proceedings of the 2001 Winter Sirnulation Conference*, 2001.

[12] J. E. Hale and D. P. Hale, "Evaluating testing effectiveness during software evolution: a time-series cross-section approach," *Journal of Software: Evolution and Process,* vol. 24, no. 1, pp. 35-49, 2012.

[13] H. Glazer, J. Dalton, D. Anderson, M. Konrad and S. Shrum, "CMMI of Agile: Why Not Embrace Both!," Canregie Mellon University, Hanscom, 2008.

[14] N. Davis, "Secure Software Development Life Cycle Processes: A Technology Scouting Report," Carnegie Mellon University, Pittsburgh, 2005.

[15] P. E. McMahon, Integrating CMMI and Agile Development: Case Studies and Proven Techniques for Faster Performance Improvement, Addison-Wesley Professional, 2011.

[16] S. W. Ambler, "Dr. Dobb's Agile Newsletter 02/08," *Dr. Dobb's The World of Software Development,* 2008.

[17] C. Joned, "Strategies for Managing Requirements Creep," *IEEE Computer,* vol. 26, no. 6, pp. 92-94, June 1996.

[18] A. Aurum, R. Jeffery, C. Wohlin and M. Handzic, Managing Software Engineering Knowledge, Springer Verlag, 2003.

[19] P. Fowler and S. Rifkin, "Software Engineering Process Group Guide," Pittsburgh, September 1990.

[20] S. C. Shimeall, "Managing the Testing Process: Opening the Black Box," 2012.

[21] D. Goldenson and D. Gibson, "Demonstrating the Impact and Benefits of CMMI: An Update and Preliminary Results," Carnegie Mellon University, Pittsburgh , 2003.

[22] D. F. Rico, "Software process improvement: Modeling return on investment (ROI)," in *Software Engineering Process Group Conference* , Washington D.C., 2002.

Appendix A (Figures)

Requirements Worksheet Template

Table 5: Excerpt from the Requirements Worksheet Template

Summary	Req-ID	Para-ID	Req-Title	Requirement Text	Component	Issue Type	Req-Type	Req-Source	Reporter	Assignee	Status	Priority
GEN-00 : 1.0 : Introduction	GEN-00	1.0	Introduction	**1.0 Introduction**	System	Requirement	Header				Requires Review	None (Unknown/Unassigned)
GEN-01 : 1.0.1 :	GEN-01	1.0.1			System	Requirement					Requires Review	None (Unknown/Unassigned)
					System	Requirement					Requires Review	None (Unknown/Unassigned)
GEN-02 : 2.0 : Reference Docs	GEN-02	2.0	Reference Docs	**2.0 Reference Documents**	System	Requirement	Header				Requires Review	None (Unknown/Unassigned)
GEN-03 : 2.0.1 :	GEN-03	2.0.1			System	Requirement					Requires Review	None (Unknown/Unassigned)
					System	Requirement					Requires Review	None (Unknown/Unassigned)
GEN-04 : 3.0 : Requirements	GEN-04	3.0	Requirements	**3.0 Requirements**	System	Requirement	Header				Requires Review	None (Unknown/Unassigned)
GEN-05 : 3.1 : System Reqs	GEN-05	3.1	System Reqs	**3.1 System Requirements**	System	Requirement	Header				Requires Review	None (Unknown/Unassigned)
GEN-06 : 3.1.1 : General Reqs	GEN-06	3.1.1	General Reqs	**3.1.1 General System Requirements**	System	Requirement	Header				Requires Review	None (Unknown/Unassigned)
GEN-07 : 3.1.1.1 :	GEN-07	3.1.1.1			System	Requirement					Requires Review	None (Unknown/Unassigned)
					System	Requirement					Requires Review	None (Unknown/Unassigned)
GEN-08 : 3.1.2 : States & Modes	GEN-08	3.1.2	States & Modes	**3.1.2 Required State and Modes**	System	Requirement	Header				Requires Review	None (Unknown/Unassigned)

Table 6: Requirements Worksheet Attributes Definition

Attribute	Description of Responsibilities
Summary	This code is automatically generated by this template by combining two other fields: <*Req-ID*> : <*Para-ID*>: <Req-Title>. It should not be altered manually.
Req-ID	This is a sequentially-assigned ID for the requirement that is unique within the project. It is a capitalized short prefix, then a dash (-), then a zero-padded sequence number. "GEN" is the prefix for most high-level system requirements and each system feature/function usually gets its own prefix that is unique within the project. The sequence number starts at "00" within each prefix. As the first line of every component feature, the main header row would always be assigned "00." *Ex. GEN-00 (Introduction), MAP-08 (Eighth requirement under the Map component's header)*
Para-ID	This is a series of numbers separated by periods (.). It represents a numeric outline of the system functional hierarchy and capabilities, which is resulted from the functional analysis and allocation process. All related requirements will follow the numeric outline started in this template. High-level function headers would get Para-IDs starting with 3.2.1, 3.2.2, 3.2.3, ... related requirements get an added period followed by a number starting from 1. *Ex. 3.2.2.1 (First requirement under the second component's header), 3.2.2.1.2 (Second child of previous example)*
Req-Title	This a brief descriptive title for the requirement which will be used as part of the "Summary" field and will be displayed in high-level reports.

	Ex. Mission editability
Requirement Text	This is the full description of the requirement. For header requirements, this has the form: <*Para-ID*> <*Section title*>. Info requirements would include free-format supplemental information. Most other type of requirements would include a full "shall-statement" specification of the requirement.
Component	This is the major component group to which the requirement will be allocated to. The component list is usually identified when physical architecture of the system is developed and the functional analysis and allocation is performed (allocating functions and requirements to components of the system). The high-level system requirements would generally use "System." Other feature components need to be determined and added to this worksheet and JIRA on a project-by-project basis. "Other" is generally used for anything that does not clearly fit into other components.
Issue Type	Every line will have an issue type of "Requirement", because this is a requirements document. It is needed for JIRA import purposes.
Req-Type	This describes the nature of the requirement and determines how it should be used for analysis. There are currently eight types: Constraint, Data, Functional, Header, Info, Interface, Non-Functional, and Other. Header is a special type that is just used for structural labeling of the requirements. Info requirements provide supplementary information and do not need to be followed. This attribute with enumeration values exist in JIRA.
Req-Source	This is the source of the requirement; where it came from. There are currently seven sources: Oper. Concept, Customer Input, Derived, Design Doc., Legacy, Sys. Engineer, Domain Expert. Header requirements leave this blank because they are inherent to the structure of all projects. This attribute with enumeration values exist in JIRA.
Reporter	This is the JIRA username of the person, usually a Systems Engineer, who originally added this requirement. Additional usernames may need to be added on a project-by-project basis.
Assignee	This is the JIRA username of the person, usually a Systems Engineer, who currently manages this requirement and who would answer questions regarding it. Additional usernames may need to be added on a project-by-project basis.
Status	Every line will have a status of "Requires Review", because all requirements from this document will require initial review. It is needed for JIRA import purposes.
Priority	Every line will have a priority of "None (Unknown/Unassigned)", because no requirements from this document will initially have a special priority. It is needed for JIRA import purposes.

Requirements Responsibilities during JIRA Processes

State	SE	T & E	PM/LPE	EA/Dev	CM/QA
Pre-JIRA	• Elicit requirements • Gather requirements in the Excel template • Perform initial review & requirements analysis • *Create the HLDD* • **Give reviewed requirements to CM for import →**	• *Review requirements and the HLDD; provide feedback*	• *Review requirements and the HLDD; provide feedback*	• *Review requirements and the HLDD; provide feedback* • *Provide input and support the HLDD development* • Create features, tasks (product backlog)	• Produce, review, and control the Initial Requirements Baseline (IRB) • *Control and manage the HLDD* • **Import requirements worksheet into JIRA** ***Import***
JIRA Requires review	• Refine, derive and logically group requirements. • *If required link child-parent requirements* • Reject/close reqs. that will not be implemented • **Accept requirements** ***Promote***	• Review requirements • Create test suites	• Review requirements • Prioritize features, tasks, and requirements • Assign tasks/subtasks for development	• Review requirements • Create subtasks	• Produce, review, and control the initial Functional Requirements Baseline(FRB)
JIRA Approved	• Generate requirements traceability matrix (RTM) • Allocate requirements to system components • *Reject reqs. that will not be implemented* • **Verify subtasks are linked to requirements** ***Promote***	• Create test cases • Create test suites • Assign Verification Methods for each requirement	• Review test cases and test suites • Review the RTM • *Reject requirements that will not be implemented*	• Link subtasks to requirements	• Produce, review, and control the initial Allocated Requirements Baseline (ARB) • Review and maintain the RTM
JIRA In progress	• Review test cases and provide feedback to T&E • *Reject reqs. that will not be implemented* • Revise requirements and approve by CCB • **Verify test cases are linked to requirements** ***Promote***	• Create test cases • Link test cases to requirements. • Revise test cases and update links as needed • Review requirements changes and approve	• *Reject requirements that will not be implemented* • Review test cases and provide feedback to T&E • Review requirements changes and approve	• Develop subtasks • Review requirements changes and approve • Update requirements links as needed	• Review requirements changes and approve
JIRA Open	• Generate RTVM and SSS as required • Revise requirements and approve by CCB • Review and monitor test results • *Reject requirements that will not be implemented*	• Execute test cases and record the results • *Create and link bugs* • **Verify that test cases pass** ***Promote***	• Review and monitor test results • *Reject requirements that will not be implemented*	• Develop subtasks • *Fix bugs*	• Produce, review, and control the Test Requirements Baseline TRB) • Review and maintain the RTVM and SSS • Generate nightly builds
JIRA Resolved	• Run functional tests to verify requirements and validate the system • Report any issues or system malfunctions to the PM.PE and T&E • Review test reports	• When all requirements are Resolved, conduct a full acceptance test • Generate test reports • *Create and link bugs and re-open related requirements*	• Review test reports • **When the full acceptance test passes, transition the requirements to the "Closed" state** ***Promote***	• *Fix bugs* • Review test reports	• Generate nightly builds • Review and control the final test reports • *Generate, review and control the requirements change history and audit reports*
JIRA **Closed**			• Review satisfied and rejected requirements with stakeholders and decide on follow-on work		• Generate the final build • Produce, review, and control the final products and deliverables for the Customer

Appendix B: Project Test Plan Template

Project Test Plans for Live Programs projects are driven by CDRL DIDs and currently follow MIL-STD-498-STP DID. Project test plans on the MODSIM and other groups employ a subset of these data elements. The corporate software management plan will address the specific deliverables for the Live Programs and MODSIM testing efforts.Section summaries are provided for guidance. Refer to MIL-STD-498-STP DID for specific details.

1 Scope.

1.1 Identification. This paragraph shall contain a full identification of the system and the software to which this document applies.

1.2 System overview. This paragraph shall briefly state the purpose of the system and the software to which this document applies. It shall describe the general nature of the system and software.

1.3 Document overview. This paragraph shall summarize the purpose and contents of this document and shall describe any security or privacy considerations associated with its use.

1.4 Relationship to other plans. This paragraph shall describe the relationship, if any, of the STP to related project management plans.

2. Referenced documents. This section shall list the number, title, revision, and date of all documents referenced in this plan.

3. Software test environment. This section shall be divided into the following paragraphs to describe the software test environment at each intended test site.

3.x (Name of test site(s)). This paragraph shall identify one or more test sites to be used for the testing, and shall be divided into the following subparagraphs to describe the software test environment at the site(s).

3.x.1 Software items. This paragraph shall identify by name, number, and version, as applicable, the software items necessary to perform the planned testing activities at the test site(s).

3.x.2 Hardware and firmware items. This paragraph shall identify by name, number, and version, as applicable, the computer hardware items that will be used in the software test environment at the test site(s).

3.x.3 Other materials. This paragraph shall identify and describe any other materials needed for the testing at the test site(s).

3.x.4 Proprietary nature, acquirer's rights, and licensing. This paragraph shall identify the proprietary nature, acquirer's rights, and licensing issues associated with each element of the software test environment.

3.x.5 Installation, testing, and control. This paragraph shall identify the developer's plans acquiring or developing each element of the software test environment, installing and testing each item of the software test environment prior to its use, and controlling and maintaining each item of the software test environment

3.x.6 Participating organizations. This paragraph shall identify the organizations that will participate in the testing at the test sites(s) and their roles and responsibilities.

3.x.7 Personnel. This paragraph shall identify the number, type, and skill level of personnel needed during the test period at the test site(s), the dates and times they will be needed.

3.x.8 Orientation plan. This paragraph shall describe any orientation and training to be given before and during the testing.

3.x.9 Tests to be performed. This paragraph shall identify, by referencing section 4, the tests to be performed at the test site(s).

4. Test identification. This section shall be divided into the following paragraphs to identify and describe each test to which this STP applies.

4.1 General information. This paragraph shall be divided into subparagraphs to present general information applicable to the overall testing to be performed.

4.1.1 Test levels. This paragraph shall describe the levels at which testing will be performed.

4.1.2 Test classes. This paragraph shall describe the types or classes of tests that will be performed.

4.1.3 General test conditions. This paragraph shall describe conditions that apply to all of the tests or to a group of tests. Also included shall be the approach to be followed for retesting/regression testing.

4.1.4 Test progression. In cases of progressive or cumulative tests, this paragraph shall explain the planned sequence or progression of tests.

4.1.5 Data recording, reduction, and analysis. This paragraph shall identify and describe the data recording, reduction, and analysis procedures to be used during and after the tests identified in this STP.

4.2 Planned tests. This paragraph shall be divided into the following subparagraphs to describe the total scope of the planned testing.

4.2.x (Item(s) to be tested). This paragraph shall identify a CSCI, subsystem, system, or other entity by name and project-unique identifier, and shall be divided into the following subparagraphs to describe the testing planned for the item(s). (Note: the "tests" in this plan are collections of test cases. There is no intent to describe each test case in this document.)

4.2.x.y (Project-unique identifier of a test). This paragraph shall identify a test by project-unique identifier and shall provide the information specified below for the test.

a. Test objective
b. Test level
c. Test type or class
d. Qualification method(s) as specified in the requirements specification
e. Identifier of the CSCI requirements and, if applicable, software system requirements addressed by this test.
f. Special requirements
g. Type of data to be recorded
h. Type of data recording/reduction/analysis to be employed
i. Assumptions and constraints
j. Safety, security, and privacy considerations associated with the test

5. Test schedules. This section shall contain or reference the schedules for conducting the tests identified in this plan. It shall include:

a. A listing or chart depicting the sites at which the testing will be scheduled and the time frames during which the testing will be conducted
b. A schedule for each test site depicting the activities and events listed below, as applicable, in chronological order with supporting narrative as necessary:
    1. Onsite test period and periods assigned to major portions of the testing
    2. Pretest on-site period needed for setting up the software
    3. Collection of database/data file values, input values, and other operational data needed for the testing
    4. Conducting the tests, including planned retesting
    5. Preparation, review, and approval of the Software Test Report (STR)

6. Requirements traceability. This paragraph shall contain:

a. Traceability from each test identified in this plan to the CSCI requirements
b. Traceability from each CSCI requirement and, if applicable, each software system requirement covered by this test plan to the test(s) that address it. The traceability shall cover the CSCI requirements in all applicable Software Requirements Specifications (SRSs) and associated Interface Requirements Specifications (IRSs), and, for software systems, the system requirements in all applicable System/Subsystem Specifications (SSSs) and associated system-level IRSs.

7. Notes. This section shall contain any general information that aids in understanding this document. This section shall include an alphabetical listing of all acronyms, abbreviations, and their meanings as used in this document and a list of any terms and definitions needed to understand this document.

Appendixes. Appendixes may be used to provide information published separately for convenience in document maintenance. As applicable, each appendix shall be referenced in the main body of the document where the data would normally have been provided and shall be lettered alphabetically.

Appendix C: Training Document Slides

Requirements

Requirement Status Explanation

Before you learn your responsibilities with requirements, here's an overview of the 'life-cycle' of requirements.

- *(Note: The process is new, so other users may erroneously push a requirement to an incorrect status, so always view the actual requirement to make sure)*

If a requirement has a particular *status*; here's what it means:

- **Requires Review:** *Requirement is still waiting to be approved; it may change so don't write any test case(s) on it yet*
- **Approved** *Requirement will be implemented as written. You should select the Req. Verification Methods and start creating and linking test case(s) to it*
- **In Progress:** *Requirement has the developer tasks linked to it. You should definitely be working on writing and linking the test case(s)*
- **Open:** *Requirement has all the necessary test case(s) linked and completed **(b/c WE - T&E - wrote and linked them)***
- **Resolved:** *Requirement's test case(s) are **ALL PASSING***
- **Re-opened:** *Any one of a Requirement's test case(s) are currently FAILING*
- **Closed:** *An **official, fielded release** was made in which all the requirement's test case(s) PASS*

Part-Time Responsibilities:

1. Monitor your *tg:Home* dashboard for requirements you are being asked to write/review/edit

Filter Results: Requirements I Need to Write/Review/Edit

Project	Summary
Gonzo Training - T&E	Make GONZO fill in spreadsheets

Displaying issues **1** to **1** of **1** matching issues.

   - If you are asked to write/review/edit a requirement see the below section
2. Monitor your *tg:Home* dashboard for requirements that need test cases written and/or linked

Filter Results: My Requirements that Need Test Cases Written and/or Linked

Project	Summary
ARES - _ICODES Data Cleanser	ECP-356 (1/2) Connect Standalone and Embedded DC to CSL
ARES - _ICODES Data Cleanser	ECP-257 (3/3) Default Symbol Name to second-highest spot in Symbol Mapping library Report Settings
ARES - _ICODES Data Cleanser	ECP-257 (2/3) Add tooltips to Symbol Name attribute

Displaying issues **1** to **3** of **3** matching issues.

3. Study the requirements you are responsible for testing (also in your dashboard) and ask for clarification/adjustments if needed.
    -

Two Dimensional Filter Statistics: Requirements I am Responsible For Testing		
	Status	
Components	In Progress	T:
Data Cleanser Libraries	2	2
Data Cleanser GUI	3	3
Data Cleanser Embedded	3	3
Total Unique Issues:	3	3
Showing 3 of 3 statistics. Filter: Requirements I am Responsible For Testing		

4. Edit the requirement and select the **"Req-Verification Method(s)"** that will be used. The choices are:
    1. **Test:**
        - Most Common: Give the software input(s) and check for certain output(s)
            - *Example Requirement: Word 2020 will auto-generate numbered lists*
                - *A bunch of test cases testing numbering in Word 2020*
    2. **Analysis:**
        - Interpret and analyze results (i.e. requires you to run multiple tests and compare the results with some formula)
            - *Example Requirement: Word 2020 will open a document an average of at least 25% faster than Word 2015 on the same hardware*
                - *A performance test case checking the time it takes to import a document*
    3. **Inspection:**
        - You don't give the software any inputs, you just examine it to make sure it meets the requirements
            - *Example Requirement: Word 2020 will have over 100 fonts available to choose from*
                - *A test case that opens word and makes sure over 100 fonts are available to the user*
            - *Example Requirement: Word 2020 help system content*
                - *A test case that would check through content of the help system for accuracy.*
    4. **Demo:**
        - Do a practical application of the software to show it can do the requirement (often involve a basic "smoke" test is needed to confirm a general ability).
            - *Example Requirement: Word 2020 will allow users to directly and easily publish books to Amazon*
                - *A test case where a user runs through a typical scenario publishing a book to Amazon*
            - *Example Requirement: Word 2020 will work in Windows Vista*
                - *A "smoke" test case where a user runs through basic functionality showing the software works in that old operating environment*

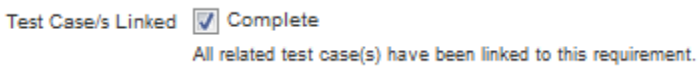
- *Edit the "Req-Verification Method" before writing and linking the test cases, but you can go back and edit it after if you need to.*

Create the "shell" *(summary and description)* of the new test cases that are needed OR update the description/steps of an existing test case. If you don't have time to finish writing the test case (or don't have enough info yet) be sure to mark the existing Test Case as **"Incomplete"**.

Link those test case(s) to the requirement(s) they will check

Finish writing/updating the test cases so they test the assigned requirements.

Edit the requirement and **check the box:** ***"Test/s Link Completed"*** when you've linked and finished **every** necessary test case to the requirement.

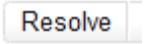


- **THAT IS VERY IMPORTANT TO DO AND MEANS YOU THINK YOUR TEST CASES ARE DONE AND WILL COMPLETELY VERIFY THE REQUIREMENT**

Click **"Tests Linked"** button at the top

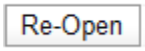
- **Never click 'Tasks Linked' or "push" the requirement to the spot where you can click it. If you want to click it and it isn't available to you, please ask a T&E staff member for help.**

ONLY IF **ALL** the linked test cases pass, click the **"Resolve"** button at the top



  - *If you are unsure if all the test cases really pass, please ask a T&E staff member for help.*

If **ANY** of the linked test cases fail, click the **"Re-Open"** button at the top (or, if it has never been "Resolved" just leave it alone)



  - *If you are unsure if any of the test cases are really failing and merit re-opening the requirement, please ask a T&E staff member for help.*

NEVER "Close" a requirement *(this is done by someone else and happens only after an official version is released to the field with all the test cases passing)*

Always monitor projects dashboard for requirements still needing test cases or requirements you are being asked to write/review/edit.

Filter Results: Requirements I Need to Write/Review/Edit

Project	Summary
Gonzo Training - T&E	Make GONZO fill in spreadsheets

Displaying issues **1** to **1** of **1** matching issues.

Filter Results: My Requirements that Need Test Cases Written and/or Linked

Project	Summary
ARES - _ICODES Data Cleanser	ECP-356 (1/2) Connect Standalone and Embedded DC to CSL
ARES - _ICODES Data Cleanser	ECP-257 (3/3) Default Symbol Name to second-highest spot in Symbol Mapping library Report Settings
ARES - _ICODES Data Cleanser	ECP-257 (2/3) Add tooltips to Symbol Name attribute

Displaying issues **1** to **3** of **3** matching issues.

Two Dimensional Filter Statistics: Requirements I am Responsible For Testing

	Status	
Components	**In Progress**	**T:**
Data Cleanser Libraries	2	2
Data Cleanser GUI	3	3
Data Cleanser Embedded	3	3
Total Unique Issues:	3	3

Showing **3** of **3** statistics.
Filter: Requirements I am Responsible For Testing

- Please save the following file in a safe place on your **BugBox**
  - **\\tapslo034\Public\T&E_Training\Checklists\T&E_Requirements_Checklist.pdf**
- Please print it out as well so you have it for reference
- Once you've read through the checklist as a reminder, please continue on to the next slide.

Full-Time Responsibilities:

1. Assign testers who needs to write/edit/review requirements by temporarily making them the "Assignee" of the requirement
   - *Once testers have finished writing the requirement, make sure it gets reviewed by a System Architect and/or Tech Lead and assigned to a developer*
2. Monitor your project'(s) dashboard'(s) for new requirements that need Test Cases (aka are "Approved" or "In Progress") and...

Two Dimensional Filter Statistics: GDMS-NG All Requirements

Components	Status: Open	In Progress	Resolved	Closed	Requires Review	T:
Transponders/Panic Alarms	13	0	68	1	0	82
User Settings	9	0	28	39	0	76
Map	11	3	37	2	0	53
Geofences	18	0	30	3	0	51
Routes	9	0	41	1	0	51
Overlays	6	0	28	1	0	35
Replay	12	0	18	0	0	30
Incidents	8	0	14	7	0	29
Placemarks	4	0	17	0	0	21
RFID	13	0	5	0	0	18
Framework	6	0	2	0	0	8
User Login/Mgmt	5	0	0	0	0	5
No component	0	0	0	0	1	1
Total Unique Issues:	**114**	**3**	**288**	**54**	**1**	**460**

Showing **13** of **13** statistics.
Filter: GDMS-NG All Requirements

Two Dimensional Filter Statistics: GDMS NG Requirements Needing Test Cases

Components	Status: In Progress	T:
Map	2	2
Total Unique Issues:	**2**	**2**

Showing **1** of **1** statistics.
Filter: GDMS NG Requirements Needing Test Cases

Two Dimensional Filter Statistics: GDMS NG Req. Test Cases Needing Review

Components	Status: In Progress	T:
Map	1	1
Total Unique Issues:	**1**	**1**

Showing **1** of **1** statistics.
Filter: GDMS NG Req. Test Cases Needing Review

3. Assign testers the requirements they are responsible for creating test cases for (and testing) by making them a 'Watcher' on a requirement
4. Make sure testers are making progress on their requirements
5. Make sure **the box: *"Test Case/s Linked"* is checked when the test cases are done**

6. Monitor project's dashboard for "req. test cases needing review" *(to make sure development is reviewing the test cases linked to the requirements we say are done)*.
7. Once the test cases have been approved and are in an **"Open"** state... *(the SA/Tech Lead should push the requirement through the workflow by clicking 'Tests Linked' .. if they don't, you can always take matters into your own hands and do it.. but it is always best the test cases get reviewed first)*
    1. *"Resolve"* the requirement when all linked test cases *"Pass"*.
    2. '*"Re-Open"* the requirement if any of the linked test cases *"Fail"*.
    3. Repeat the *"Resolve"* and '*"Re-Open"* process until....

When development is in the last development cycle before release; alert the appropriate PMs of any requirements that are still NOT *"Resolved"*

"Close" requirement when all linked test cases *"Pass"* **IN THE OFFICIAL RELEASE OF THE SOFTWARE**.

Requirements Workflow:

(See "Requirements Responsibilities during JIRA Processes" In Appendix A)

Writing Requirements:

After working in T&E for a while, you may be asked to write the initial drafts of requirements. If this is the case, it will usually be because a user has requested a feature (and the government/customer has agreed to pay for it). If you are asked to write a requirement, it is because we hope that be having your input at the *beginning* it will prevent future bugs. So keep the following tips in mind:

- Remember, you are writing a requirement that describes what the software needs to do *so that* the customer's need is fulfilled (NOT the customer need).
- Be as specific and clear as possible so there is no confusion
- You may need to write multiple requirements to fulfill the entire request of the user. Divide it into multiple requirements if it is the logical thing to do.

Example:

- **Feature Request: Orek vacuum cleaner should be able to pick up a bowling ball**
    - *OK* Requirement that could be written:
        - REQ1: Orek will have suction strength to hold a 10 lb bowling ball 5 feet in the air
    - **GOOD** Requirements that could be written:
        - REQ1:Vacuum will generate at least 100-PSI when on high power
        - REQ2:Vacuum attachment will be curved to make an air tight seal with a sphere 1 foot in diameter
        - REQ3:Vacuum arm will be able to handle a weight of 10 lbs without breaking

Requirements Details:

- More information on requirements can be found **here** in the document called **ENG-RDRM-WI**

JIRA Dashboards

tg:Home Dashboard

You should have already set up the *tg:Home* dashboard at the **beginning** of the training.
Your tg:Home dashboard shows you...

- Your resolved & migrated bugs that developers are waiting for you to confirm or re-open
- Your "orphan" bugs that still need to be linked to test case
- The status of your test cases and test suites *(including which ones you need to **run**, **write**, and **fix**)*
- The tasks (and new tasks) you've been assigned
- Any requirements that you've been assigned to write or edit
- Any requirements you are responsible for writing test cases for and updating
- Any CS Problem Reports (*i.e. user problems from the field*) that you've been assigned to investigate/test.
- Your favorite "filters" (more on filters in the next slide)
- Leader-boards for TG and your all time bug statistics (just for fun)

To help you sort out what is what, remember the *general* color code for **ALL** the gadgets/widgets in the dashboards is:

- **Blue** = bugs, suggestions, and CS Problem Reports
- **Green** = test cases and test suites
- **Orange** = tasks (your's & developer's)
- **Red** = requirements
- **Purple** = JIRA specific filters

Now that you understand what bugs, tasks, and test cases are: please take a look again at your *tg:Home* dashboard to the **right** (see Figure 6)

Quiz 1:

1. Open the dashboard to the **left** and try to identify **ALL** the things the tester needs to do
2. Open the dashboard to the **right** and read the answer key below:
   A. 17 bugs needs to be checked by the tester
   B. 181 test cases need to be run by the tester
   C. 65 test cases need to be 'fixed' by the tester
   D. 32 test cases need to be written by the tester
   E. 2 new tasks still needed to be started by the tester
   F. 56 orphan bugs needs to be linked to test cases
   G. 1 problem from the field needs to be researched by the tester
   H. 1 requirement needs test cases linked to it and is the tester's responsibility

Application Specific Dashboards

You will need to monitor the dashboard(s) of the application(s) you test.

Let's add one as an example:

1. Once logged into **jira** click the tiny "Manage Dashboard" link in the top right
2. Click the "popular" tab
3. Click the star next to "tg:SLP"
4. Click 'Home' at the top left
5. Click the *tg:SLP* link in the top left to see the dashboard for SLP

Every application we test in TG has its own dashboard. You can follow the same process to add the dashboard for the application(s) you test in the future

The application dashboard..

- Reminds you of the bugs (listed by your name) you have to confirm
- Lets you see the bugs other testers need to confirm (**YOU are responsible to check the bugs of other testers if the bug relates to the applications and/or areas YOU test**)
- Shows what test cases still need to be run, fixed, or written
- Shows what requirements need test cases written
- **Lets you see recently completed developer tasks in your applications**
- Gives you a feel for the overall state of the application

Quiz 2:

1. Open the dashboard to the **left** and try to identify **ALL** the things the tester(s) who test the application need to do
2. Open the dashboard to the **right** and read the answer key below:
   A. 25 bugs needs to be checked
   B. 54 orphan bugs need to be linked to test cases
   C. 5 test cases need to be written
   D. 275 test cases need to be run
   E. 2 requirements need test cases written and/or linked to them **that make sure the application fulfills those requirements**