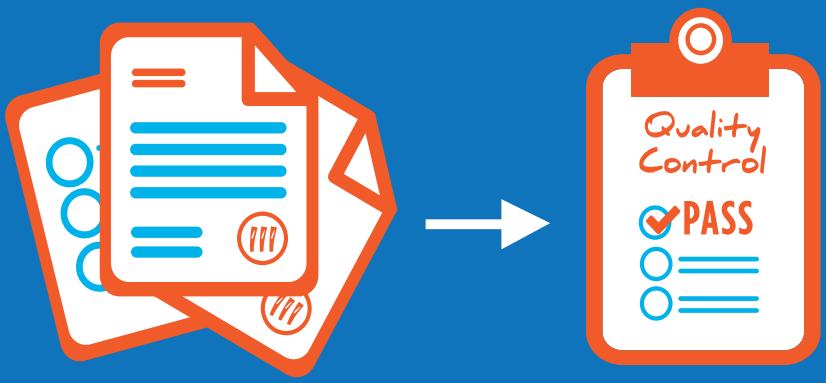
VERSION
1.0

Segue Technologies, Inc.

QUALITY CONTROL

Critical to the Quality of Software Products



www.seguetech.com

Version 1.0

Published 2014 by Segue Technologies, Inc.

Cover Design, Layout, and Illustrations created by
Segue Technologies.

All links featured in this book can be found at

www.seguetech.com/blog

About Segue Technologies

Segue has developed innovative, dependable, and user-friendly applications since our founding in 1997. We provide a wide range of Information Technology services, focusing on Software Engineering, Information Management, Quality Assurance, and Systems Integration. We are a growing small-business, supporting Federal, Commercial, and Non-Profit clients.

www.seguetech.com

TABLE OF CONTENTS

4 Foreword

6 Infographic: The Rising Costs of Defects

QUALITY CONTROL AND QUALITY ASSURANCE

7 Defining Quality Control and Quality Assurance

11 What is Quality Control as a Service?

17 Different Methods of Software Testing

AUTOMATION AND MANUAL TESTING

21 For Great Quality, Bring Your Software Testers in Early

25 Automation Testing: Problems, Myths, and Misconceptions to Consider

29 How Important is Test Automation in a Software Project?

BUGS AND DEFECTS

34 Quality Control: Kills Bugs Dead

37 Straightforward Bug Tracking for Quality Control

41 Rising Cost of Defects

PROCESS IMPROVEMENT

45 Why is CMMI Appraisal important for Software Development Companies?

50 What Does it Mean to be Appraised as CMMI-DEV Level 3?

FOREWORD

Is Quality Control (QC) an important part of the software development lifecycle? Can't you just develop something correctly and save money? SQS compiles an annual list of the world's top ten software failures, and this year's launch of Healthcare.gov was number one. This is a large and very public example of the importance of QC in software development and how neglecting it can be very costly. This highly publicized website fiasco brought QC to the forefront of the news and made it a prime water cooler topic. Lorinda Brandon, Quality Evangelist wrote:

"Suddenly, Americans are sitting at their kitchen tables – in suburbs, in cities, on farms – and talking about quality issues with a website. The average American was given nightly tutorials on load testing and performance bottlenecks when the site first launched, then crumbled moments later. We talked about whether the requirements were well-defined and the project schedule reasonably laid out. We talked about who owns the decision to launch and whether they were keeping appropriate track of milestones and iterations. ...When the media went from talking about the issues in the website to the process used to build the website was when things really got interesting. This is when software testers stepped out of the cube farm behind the coffee station and into the public limelight."

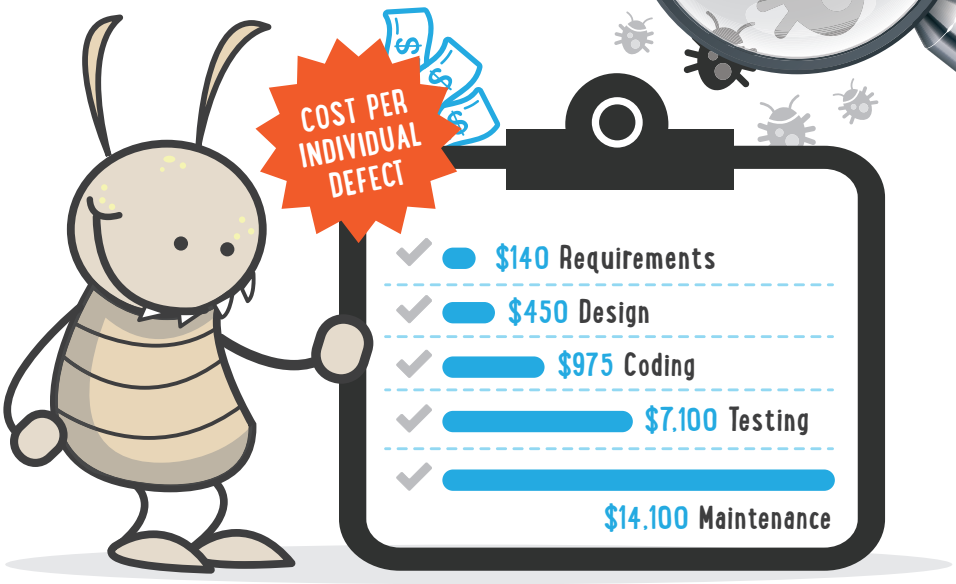
QC is one of the most important parts of the software development lifecycle. It is a key contributor to the ultimate quality of a product. QC activities are not just about finding issues, but also about using testing to verify and validate that the product meets the stated requirements, design, and technical specifications. Quality improvements help a business to reduce post release costs of support and service, while boosting customer confidence and generating a good reputation that could translate into greater revenue opportunities. A sufficient investment in QC would have solved numerous problems with the healthcare.gov website, from the function of the site itself, to the tremendous political and media turmoil that resulted.

This eBook, written by our technical experts, is based on experiences that have helped us achieve quality goals and highly effective software products for our customers. We have grouped the content into three sections:

- Quality Control and Quality Assurance: Defines QC, QA and some testing methods
- Automation and Manual: Best Practices and tips for testing
- Bugs and Defects: The importance of tracking and finding defects

Is QC important? It's absolutely critical! And, I'm excited that Software quality has been brought into the spotlight.

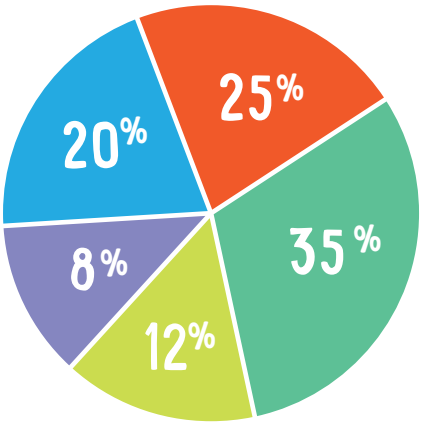
THE RISING COSTS OF DEFECTS



DIVISION OF DEFECTS
INTRODUCED INTO SOFTWARE BY PHASE

Software Development Phases

1. REQUIREMENTS
2. DESIGN
3. CODING
4. USER MANUALS
5. BAD FIXES



%
Percent of Defects Introduced

SOURCE: Computer Finance Magazine
http://qa.siliconindia.com/news/Bugs-and-Their-Impact-on-Productivity--Profitability-nid-127700.html
http://www.jrothman.com/2000/10/what-does-it-cost-you-to-fix-a-defect-and-why-should-you-care/

Chapter 1

Quality Control and Quality Assurance

Defining Quality Control and Quality Assurance

Quality Assurance and Quality Control are often seen as loathsome corporate entities, assigning blame, slowing down the production process with excessive control measures, and creating tons of "useless" documentation. This perception is unfortunately the result of years of bad corporate QA and QC implementation, where upper management groups impose policy (without process user input) to define adequate "Controls", untrained or unqualified personnel leading QA and QC activities, and poor communication about vision and goals.

It is far too common for companies to approach QA and QC the wrong way, thereby spoiling their staff on the concept and limiting the benefits that can be gained not only for customers, but for internal quality of work environment.

To effectively implement QA and QC, you must first properly define "Quality" and then the role that each (QA and QC) play in achieving quality. There are two general views of Quality:

- The Customer view of Quality: Is the external view that means that the product or service they receive satisfies their needs
- The Producer view of Quality: Is the internal view that is based on whether or not a product satisfies the stated requirements

QUALITY CONTROL

QC focuses on the product produced. This focus is two-fold, ensuring that both customer and producer share the same vision of quality, and that work is objectively reviewed to eliminate defects.

Unfortunately the two views of quality (customer and producer) are not always in sync; therefore QC's first goal is to reduce the "Quality Gap", the gap between customer expectations (not necessarily stated) and the development team's understanding of the explicit requirements. By minimizing the quality gap, QC ensures an end product that matches the customer's needs and expectations.

The second goal of the Quality Control entity is to find defects before they reach the customer. This effort maximizes customer satisfaction when the product or service is delivered. In a perfect world, every requirement analyst and developer would objectively check their work and communicate efficiently, both with the customer and with each other. They would therefore produce a perfect product for the customer the first time. Real world experience shows that self-checks are insufficient tools for discovering and rectifying errors and that breakdowns in communication occur in even the most in-tune organizations. As a result, Quality Control serves as an objective 3rd party entity to execute the proper checks at each critical point of the development process, thereby providing sufficient feedback to the producer (Analyst, Developer, etc.) to correct defects and avoid a cascading effect of negative consequences. In a word, Quality Control is the second look, the "spell-checker," that helps to ensure that work is on the right track from the start through the finish.

QUALITY ASSURANCE

QA focuses on processes and their continuous improvement.
Its goal is to reduce variance in processes in order to predict the quality of an output (final or interim product), gather best practices for the company, reduce cost, and reduce time to market. QA is strongly linked to innovation and creativity. Quality Assurance neither imposes nor defines processes for other people, but it provides advice and support to the process owner, which leads to the ability to measure success and make decisions based on facts.
A well-known approach to Quality Assurance is the PDCA (Plan Do Check Act) Cycle:

1. **Plan:** Define the mission, vision, and goals to be achieved by an activity or a process. Identify the procedures, methods, and tools needed to achieve the goals. Define the measures to be used to check the results of the process.
2. **Do:** Execute the plan, train users to methods, deploy products, and use tools to perform scheduled tasks.
3. **Check:** Evaluate whether or not the goal has been achieved by using measures, metrics, and facts.
4. **Act:** When gaps are defined, identify the origin of the problem and define an approach to correct or close the gap (Return to the Plan phase).

In conclusion, neither QA nor QC focus on people, as they do not care "whose fault it is" (although they do care when there is something valuable to be remembered!). The goal of a good QA and QC implementation, in any organization, is to make things better. It requires good communication between the QA/QC groups and process owners. It also requires QA/QC lead(s) to communicate and explain why, when, how, and what is being done. Key attributes for success of QA/QC implementation are:

- **Participation:** Both process owners and users need to provide their expert input on how things "should" work, and define it. QA should be a support function for process related questions.
- **Transparency:** Open communication and the ability to look at all aspects of the process are critical to fully understand and identify both what works and what doesn't
- **Clear Goals:** The entire team should not only know how QA/QC is implemented, but also the intended results

What is Quality Control as a Service?

Segue has defined a service-based delivery model for Quality Control (QC). This service has been CMMI Level 2 appraised and it is the methodology by which we conduct software quality control testing processes. QC involves the measurement of actual quality and determination of defects in a given program. Our QC service is carried out in four main stages: service request, service agreement, service delivery, and service delivery completion, which is followed by a test report.

Figure 1

QUALITY CONTROL SERVICE REQUEST

The service request is submitted to the QC manager by the customer through a completed service request form. The QC manager reviews the request and makes the estimate for the service based on the scope and level of activities requested by the customer. Determination of the estimates is a negotiated deal between the QC manager and the customer and leads to the computation of the budget for QC for the project.

Different levels of service are shown in below:

- **Smoke/Quick:** Tests basic Navigation High-level functionality for obvious visual bugs.
- **Sanity/Partial Regression:** Confirms that a recent program or code change has not adversely affected existing features, and executes Test Cases based on criticality.
- **Full Regression:** Confirms that a recent program or code change has not adversely affected existing features. Test Cases are executed for all areas regardless of criticality.
- **Requirements Verification:** Examines proposed requirements, features, and changes to ensure they are testable.
- **Performance Monitoring:** Observes response times and lapses, and reports anything that is unusually long.
- **Installation:** Tests your application installation instructions.
- **Adhoc/Exploratory:** Performs an informal assessment of the application.
- **Functional:** Assures that each element of the application meets the functional requirements cases.

- **System:** After receiving the fully integrated application (All features are released to testing), performs End-to-End scenario testing of the completed application.
- **Stress/Load:** This type of service only focuses on load and stress of a stable system to evaluate its robustness compared to a defined standard or target.

SEGUE INTERNAL SERVICE LEVEL AGREEMENT (SISLA)

Segue's internal service level agreement (SISLA) represents the agreement made between the Project Manager (PM) and the QC manager to facilitate effective coordination and adherence to the stated requirements. The agreement is initiated by the start of a QC effort for a Segue project and ends when either the project manager or QC manager decides to terminate it. The decision to terminate the agreement may result from a change of responsibilities or failure to adhere to project requirements by either of the agreeing parties. Disputes arising from non-compliance with the terms and conditions of a SISLA are resolved by executive management.

A SISLA specifies the requirements of the customer and the requirements of the QC manager that facilitate a complete execution of the service request. It is the PM's primary duty to provide the QC team with all the system requirements, including the business rules and tool requirements. In addition, the PM has a duty to inform the QC team about technical requirements of the system, which include the database design, database platform, browser type(s), SW language, operating system, and hardware. The QC

lead, on the other hand is responsible for updating the PM and providing a plan for execution of the SISLA and service requirements. The QC lead also conducts a series of tests to measure the quality of and identify any defects in the system.

SOURCE DELIVERY

Service delivery is a series of activities and tests conducted by the QC team to ensure the full execution of the SISLA and service requirements. This process begins soon after the approval of the service request. The QC manager verifies the level of skills and the available human resources to carry out the specified tasks. Based on the available resources, the QC manager designs the Test Strategy Document, which facilitates execution of all required tasks. This is followed by task analysis, which involves identification of the level of testing needed, task plan, suitable time allocation. The QC manager assigns the tasks to the members of the QC team according to their knowledge and skills. Once the tasks are assigned to QC team members, the QC lead designs the test by performing three major tasks, namely requirement verification, functional risk verification, and test case generation. The team sets up a production-like environment, which helps in management of test data.

The test team uses the test risk analysis results and test strategy to perform the actual tests. The test execution cycle consists of three phases, which include entry criteria, core validation, and exit criteria. The test team conducts a "lessons learned" exercise that is scheduled two days after completion of every test cycle. The lessons learned session helps in collection of critical information for improving testing for subsequent cycles and/or projects, such as:

- Factors that resulted in extemporary performance of some tests
- Issues arising from the transition between development and testing phases
- Detection of recurring defects and risks missed from the previous test cycles
- The makeup of the level of training needed

Solutions for resolving defects are identified and a decision for improvements in the subsequent cycle made.

SOURCE DELIVERY COMPLETION

After completing the service delivery phase, the team submits a QC report to the PM to furnish them with information regarding the state of the system and lessons learned during the test. Delivery of the Test Project Report by the QC manager/lead and its reception by the PM, finalizes the service delivery process unless a new service request is made to initiate a new agreement.

Different Methods of Software Testing

People interact with different variations of software on a daily basis. It may be an application you access from your smartphone (or tablet), a web page you access using a browser, or a stand-alone desktop application. However, what you might not know is that each of these examples has undergone several rounds of software testing. Software testing is the exercise of verifying that the code being deployed will meet the users' functional requirements. Software development can be a challenging endeavor, and involves taking requests from the customer and turning those requirements into actionable code. To ensure that this translation has been successful, software companies enlist several different software testing methods. These methods include smoke testing, alpha/beta testing, and regression testing.

SMOKE TESTING

Smoke testing is the practice of performing basic functionality and workflow actions against the deployed code. This verifies that additional testing can be conducted, and that the code build is working as intended at a basic level.

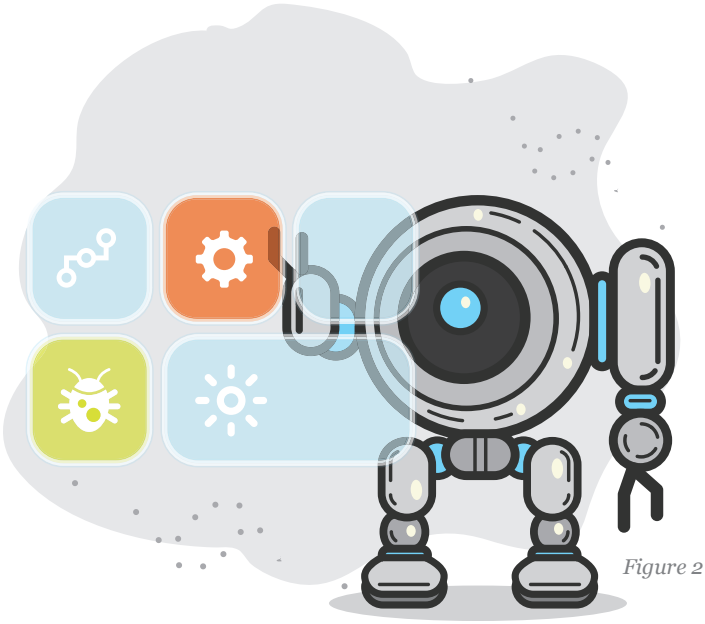


Figure 2

Smoke testing is typically performed by an internal team, several times over the course of a software development project. For example, when a new version of a software application is released, testers will smoke test the new version to ensure basic functionality (e.g. a login page) works as intended.

ALPHA/BETA TESTING

Alpha or Beta testing is performed by a small external group of users against a deployed software application. These testers perform actions to verify the functionality, look and feel, and workflow of the application before its wide release.

It's common now for software slated for a wide commercial audience (e.g. Microsoft Operating Systems) to have a set of alpha testers perform initial testing. Feedback from that group of alpha testers is used to correct bugs and improve functionality. A subsequent version of the software is then released to a wider, but still limited, audience. This round of testing is considered "beta testing", which should allow for any remaining bugs and errors to be identified before final release.

REGRESSION TESTING

Over the course of a software development project, several releases (or builds) of the application are deployed. These releases can cover new functionality to the application, or correct existing bugs. Regression testing is the practice of ensuring each build not only successfully meets the application's functional requirements, but that it did not break functionality which was deployed in previous releases.

Here is an example: Build "A" is released and tested, then build "B" is released but during testing functionality from build "A" didn't work. Regression testing verifies functionality of build "B" and build "A". For more information on regression testing check out *Regression Testing: Why You Should Incorporate It Into Your Quality Assurance Process.*

This is just a few of the software testing actions that applications can undergo. Each type of testing provides a greater opportunity for the application to not only meet the customer's needs, but also ensure that the efforts of the project development team are met with success. Segue provides several of these testing methods for our external customers, or for our software developed in-house.

Chapter 2

Automation and Manual Testing

For Great Quality, Bring Your SoftwareTesters in Early

Many clients come to us at Segue and are debating when to start testing as they're developing new software. In just about all cases, software testing should start as soon as the design and requirements are being baselined. Involvement in the beginning of a project makes testers more efficient by enabling them to learn more about the product they will be testing. Having a more thorough understanding of the product and business rules allows the tester to design test plans and cases more efficiently. This early start provides several advantages which improve the overall efficacy of software testing.

During the design and requirements phase, testers can work with developers to determine what aspects of a design are testable and what areas have higher risk. This knowledge will help prevent testing errors and make testers better equipped to design test cases and identify defects.

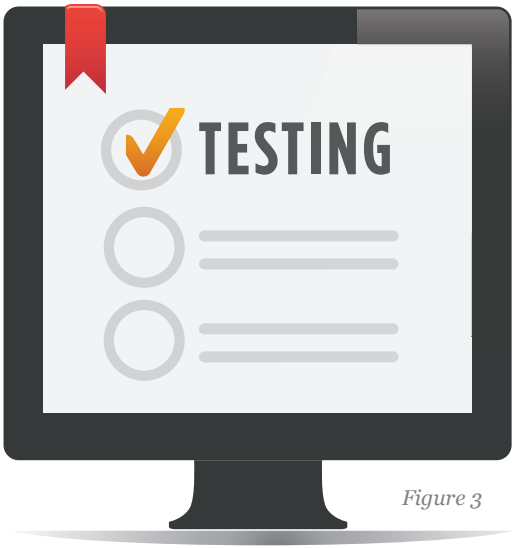


Figure 3

IDENTIFYING DEFECTS IN SOFTWARE DEVELOPMENT

Identifying defects earlier in the process is less expensive. For example, if a tester is verifying a requirement and sees that an error message is missing, they can identify this defect before a developer has spent time coding under the impression that there is no error, thus saving money on work that may need to be redone. The importance of early involvement is illustrated by a variety of studies which show that catching a fault early in the requirements and design stages is more cost effective than catching it after a system

has been released. The ratios vary between a 5:1 and 100:1 difference in cost, but all studies say it costs a lot more in resources to fix a system after release. These costs for fixing the application do not take into account the risks to an organization's reputation and the workaround costs associated with faults and missing functionality, so the true cost is much higher.

Delayed involvement limits how many issues are identified, due to both lack of time to test and the testers' lack of understanding the requirements. Waiting to bring in testers at the end of a project is a characteristic of the Waterfall Model. The Waterfall Model is the idea that each sequence should be completed before proceeding to the next. With the Waterfall Model, Quality Control is not involved until the end and therefore issues are found very late in the development life cycle. This is one of the reasons the Verification and Validation method is one of the preferred methodologies. Verification and Validation (V&V) is the process of ensuring that the product or system in development conforms to customer expectations as captured by the system requirements.

Segue's verification techniques include:

- Requirement Specification Verification
- Functional Design Verification
- Verification of the Data Migration Plan
- Reviews, Audits, Walkthroughs and "Buddy Checks"

V&V Quality Control activities are carried out in parallel to requirement, design and development activities to ensure a higher level of quality. The V-Model illustrates that testing activities

(Verification and Validation) can be integrated into each phase of the product life cycle. The Validation part of testing is integrated into the earlier phases of the lifecycle, which includes reviewing end user requirements and design documents. The specific methodologies used for testing may vary from project to project, but there is a role for testing in every phase of your project and throughout the lifetime of the application.

The goal of Quality Control (QC) is to ensure the delivery of reliable products that satisfy the stated requirements and meet the intended business need(s) of the client. To meet this goal, Segue ensures that Quality Control is involved early in the Software Development Life Cycle.

Automation Testing: Problems, Myths, and Misconceptions to Consider

Automated testing is a great complement to manual testing in that it provides reusability and repeatability to the test process, saving you money and time. Automation can help you quickly perform regression tests of the environment, which quickly determine the stability and usability of the underlying code. However, there are many misconceptions and pitfalls associated with Automated Testing. These misconceptions include the idea that Automation Testing:

- replaces manual testing,
- is developed early in the Product Life Cycle,
- can test everything, and
- always indicates bugs in the code.

Here are the reasons why these beliefs are simply myths.

AUTOMATED TESTING CAN COMPLETELY REPLACE MANUAL TESTING

Automation can never fully replace manual testing. Instead, automation's most basic function is to actually determine if the code is operational. However, automation can never replace human intu-

ition and unpredictable behavior. Furthermore, automated tests require manual intervention to create, setup, run, and interpret the test results.

As our new Automation Test Engineer here at Segue, I have hit the ground running, providing automation to several projects. Rather than replacing the Manual Testing, though, I work in conjunction with the Quality Control team to automate the stable, high-priority tests where appropriate.

AUTOMATED TESTING CAN BE USED EARLY IN THE TESTING PROCESS

To actually use Automated Testing early in the process is usually not possible as the environment typically changes rapidly, causing automated tests to easily break. Instead of creating automated tests early in the process, involve the automation team to guide development team to create code that is more conducive to automation, such as including unique DOM IDs.

AUTOMATED TESTS CAN TEST EVERYTHING

Usually, it's best to focus automation on repetitive, predictable, and simple tasks for high-level, mile-wide, inch-deep regression testing. Automation is also great for multi-user performance testing. However, as I mentioned before, automated tests can never replace the flexibility and adaptability of human testing that incorporates intuition.

FAILING AUTOMATED TESTS INDICATE ALL BUGS IN THE CODE

Although that may be the case, automated test tools can fail due to changes in the development environment from moved or renamed fields, underlying database changes, new data constraints, or test scripts that run too fast. For example, an automated test may try to click on a button before the page containing the button finishes loading.

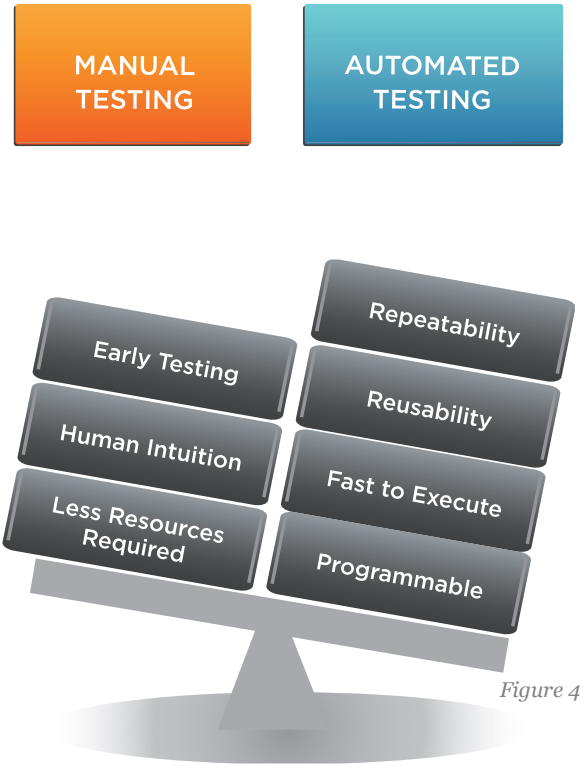


Figure 4

In summary, automated testing is a fantastic complement to manual testing but will never replace human testers. Automation successfully tests predictable and repeatable sequences of steps. These automated tests are easily repeatable and quickly executed. However, an end user's behavior is never predictable! So make sure to use automation for the expected and manual testing for the unexpected. Now that you understand a few pitfalls of Automated Testing, you can better incorporate Automation in your Testing Process with realistic goals and expectations.

How Important is Test Automation in a Software Project

Software projects, especially large and complex ones, require a significant investment in testing to ensure that they are successful. This specific investment is in time, staff resources, and additional costs, which can quickly add up and make the cost of testing increase the overall scope of the development project. Luckily, automation testing can often be implemented to speed up the process and reduce these costly testing investments. Automation testing is testing without using manual labor or testing that creates automated scripts for previous repetitive, but necessary, testing in a formalized testing process already in place. It also adds additional testing that would be difficult to perform by hand.

An example of this would be a scenario in which your team is building a new product from scratch. For each development iteration, you have to verify new requirements, implement new features and ensure that whatever you have built and tested successfully previously continues to work properly. This is where automation comes in; it not only acts as a safety net against old features or regression tests, but most importantly, it totally frees up precious developer and tester time, allowing them to focus on the tasks at hand more efficiently.

WHY IS AUTOMATION NECESSARY?

Since not all projects require automation, you need to consider the efficiency of manual tests and see how much coverage the project entails. For instance, if the project is a small one, automation testing should not be a requirement since you may not have the money or resources needed to complete the project and hire an automation specialist. However, for larger projects, bringing in an automation expert is a necessary step in testing, even if it may be a little costly. Using simple automated scripts instead of manual tests with certain projects that use websites could be both cost and time effective. This would ultimately reduce the time spent on testing as a whole.

Figure 5

WHAT AUTOMATION TESTING CAN ADD TO YOUR PROJECT

An automation specialist is a more highly skilled person than a regular tester, since he/she would already be an experienced manual tester with additional automation skills. Bringing in a person with these kinds of skills would enhance the testing process even more by reducing manual testing hours and decreasing the number of test cases and test data. Furthermore, the tools and frameworks you use to achieve automation can become a dependable resource for larger projects if there is a budget to cover that aspect of testing. Manual testing should still occur for a while even though automation helps speed up the testing process as a whole. However, automation in general should not be the real goal of your testing efforts for a big project, because if you focus on the whole picture, the main goal should be to support new development efforts by providing quick feedback to the whole team.

THINGS TO CONSIDER WITH AUTOMATION

1. **Cost and Resources:** What is your budget and how many resources can you afford?
2. **Stakeholders:** Who is involved in the project? (i.e. Project Management, Quality Control, Development, Requirements)
3. **Manageability:** What is the scope of the project? Can Automation be included and be still be cost effective?
4. **Execution Time:** Manual vs. Automation. How much time can you save with automated tests vs. manual?

In addition, it is also important to keep your test cases/scripts and test data independent of the selected test automation tool due to the changes in format, layout, design, technical changes and updates to the code. For example, creating tests with hard-coded test data, system configuration, and properties makes them difficult to maintain, because in the long run, configuring the test data makes it difficult to change your test updates midway through a project if you were to run into any unanticipated issues and problems. Keep in mind that the most important part of your automation is the tests.

THE VALUE OF AUTOMATION TESTING

A lot of project teams spend most of their time and effort creating a nice framework with a lot of features but forget about the tests. Don't let the main code become more important than the test code since what you use to test should be the main priority. The real value of any automated testing effort is derived from the test results it produces, instead of the quantity of automated test scripts. Additionally, you should not automate just for the sake of automation. Consider the costs involved, the resources and stakeholders, and the main concerns like maintainability and execution time before adding new tests. Automated tests reduce the testing time as a whole, but you need to comprehend that they become part of the production code base and therefore must be maintained just like the rest of the code, for the entire life of the application. Adding tests that are overly complex or difficult to maintain can slow down the feedback cycle to the team and should be avoided, which would lessen the importance of automation.

Here at Segue, we have seen the benefits of doing automation testing firsthand. For instance, a recent Segue customer was very appreciative of the fact that automated tests were performed at a high level on their project, which enhanced their testing abilities. Implementing automation testing in the beginning of the project cut their scope in half, despite the fact that their testing team was having trouble coming up with a tool to decrease manual testing time initially. They were able to release their project on schedule and also grasp the idea of automation to integrate into their other testing phases. This in fact reduced costs, man hours, and resources.

Customer satisfaction is the key and automated tests can be of great value, especially when time is of the essence. Testing performance in a faster, timed manner greatly decreases the chances of having to do a rework and provides you with results much more quickly.

Chapter 3

Bugs and Defects

Quality Control: Kills Bugs Dead

What type of person is a Quality Control (QC) Tester? There are many kinds of professions out there. For instance, some people are movie critics, video game testers, teachers, scientists, mystery shoppers, food tasters, and stockbrokers. All of these occupations have something in common: looking for the best quality in a product.

As a Quality Control Tester, one comparable occupation that comes to mind is an Exterminator. Both positions require someone who has just one goal in mind: to locate all types of bugs. In order to find all the bugs, the Exterminator has to test the premises, the surroundings, and numerous environments to see whether or not

the bugs are spreading from one place to another. Of course, in doing so, all of this comes at a cost. Most Quality Control costs are associated with preventing, finding, and correcting defective work (bugs). These costs can range in price from low to high, depending on the customer's specific needs. Many of these costs can be significantly reduced or even avoided based on the quality of the product and time constraints.

Figure 6

There are many ways of locating and eliminating the bugs in the QC world. Some customers like to bring in a highly decorated expert to identify and eliminate the defects. They have testing applications and software tools that will help locate the bugs from a system. In comparison to an Exterminator, a customer will call for

an expert to get rid of all necessary bugs with their high-tech tools, to do all of the dirty work to ensure a safe and bug free zone.

The Exterminator and the QC tester both require a specialized toolset to do their work. For the Exterminator, some of these may include such materials as bug sprays, chemical mixtures, traps, and possibly baits in order to hunt down the bugs. The same goes for a QC Tester in needing various testing techniques, testing tools, and resources in order to identify and report the bugs (also known as defects).

The QC Exterminator's job is to ensure the quality of the interface and its environment. He then validates the final product in order to meet the customer's needs and requirements (to include no bugs!). This will help test the surroundings and guarantee the best quality.

In conclusion, in addition to a passion for hunting down and eliminating bugs, a person who is a good QC Tester requires both functional and technical knowledge. Provided with an ample amount of time and the proper set of testing tools, a person with precision, innate skills, methodical thinking, and a desire to solve puzzles in a scientific manner would make a great Quality Control Tester.

Straightforward Bug Tracking for Quality Control

According to Joel Spolsky (a.k.a. Joel on Software), "keeping a database of bugs is one of the hallmarks of a good software team." However, it's amazing how few software shops fully utilize this strategy yet tout the importance of software quality control. Bug tracking is the process of finding defects in a product and making new versions of that product that fix the defects. Bug tracking is important as complex software systems typically have numerous bugs so managing, evaluating, and prioritizing bugs is a difficult task.

WHAT CONSTITUTES A BUG?

Before you can track software bugs, let's define what exactly constitutes a software bug. Examples of obvious software bugs are when an application crashes or displays the dreaded "404 Not Found" web page. Clearly, these are bugs, but what about other problems such as pages that are not visually appealing, are missing data, or are displaying the wrong content? Anything that requires someone on the team to make a change to the software product should be considered a bug, whether that means a change to the code, a cascading style sheet, or the content management system.

WHAT IS SOFTWARE QUALITY CONTROL?

The Quality Control (QC) team tests software to ensure quality through verification and validation. Verification and validation assures that a software system meets a user's needs. Verification answers the question: "Are we building the product correctly?" The software should conform to its specification. On the other hand, validation answers the question: "Are we building the right product?" and when to verify. If a test fails and can be duplicated, a bug is reported.

WHY TRACK BUGS THAT DON'T REQUIRE A CHANGE TO THE CODE?

Communication between teams is greatly enhanced by the simple task of tracking software bugs. How else does your QC team know why their tests are broken? Once a bug is fixed, then QC knows to re-run their test to confirm that the bug is actually fixed. Furthermore, documenting software bugs provides an invaluable historical record.

OK, so now what? How do you avoid getting bogged down by the task of tracking bugs? Simplification! First, use a database to log your bugs, preferably backed with a simple bug flow process. There are oodles of bug tracking databases you can buy. Here at Segue, we use Seapine's Test Track which tracks defects, feature requests, change requests, tasks, and more. Once you have a bug tracking process in place, ensure that all your teams consistently follow that process.

THREE PARTS TO EVERY GOOD BUG REPORT

It's pretty easy to remember the rule for a good bug report. Every bug report needs exactly three things:

1. Steps to reproduce
2. The expected results
3. The actual results

For more information on bug reports, read "How to Effectively Report Software Defects" by Quality Control Manager LaTonya Pearson.

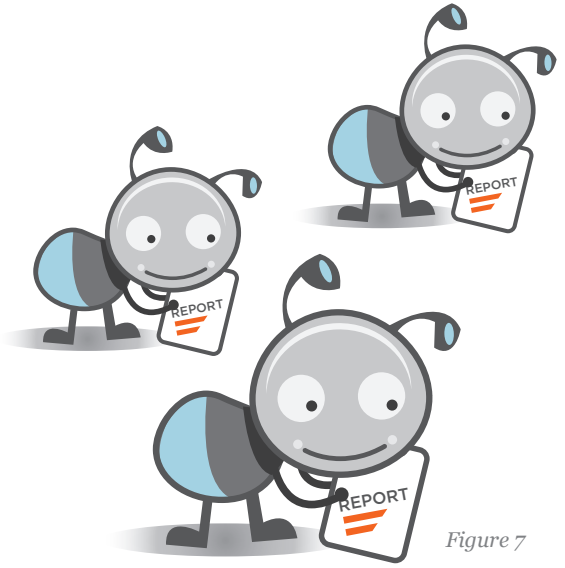


Figure 7

BUG RESOLUTIONS

Once a bug is fixed, a tester verifies the fix by following the steps to reproduce the bug. If the actual results now match the expected results, then the test passes and the bug report can be closed. Otherwise, the bug report is reopened.

Joel also says, "Remember that the only person who can close a bug is the person who opened it in the first place. Anyone can resolve it, but only the person who saw the bug can really be sure that what they saw is fixed." At Segue, we've found that while the time invested in bug tracking is minimal, software quality control and communication among teams is greatly improved.

The Rising Costs of Defects

The early detection of defects, in a process, is important for the successful execution of a project. However, the detection and prevention of defects is a significant challenge in the software industry. A large portion of the cost of software development consists of error removal and reworking on projects. The reworking process costs more than the initial process so early detection of defects during the design and requirements phase is necessary to avoid this extra expense. A large number of defects usually occur in the initial stages of a project and early defect detection will lower the overall cost of the project.

THE ADVANTAGES OF FINDING DEFECTS EARLY

The cost of finding and fixing defects rises considerably across the life cycle. This is because the requirements and design specifications will require rework before changes can be made to the code. Also, a single defect in the requirements may well propagate into several places in the design and code and, because of that, all the testing work done up until that point will need to be repeated in order to reach the confidence level in the software that we require. It is often the case that defects detected at a late stage, depending on how serious, are not corrected because the cost is too expensive.

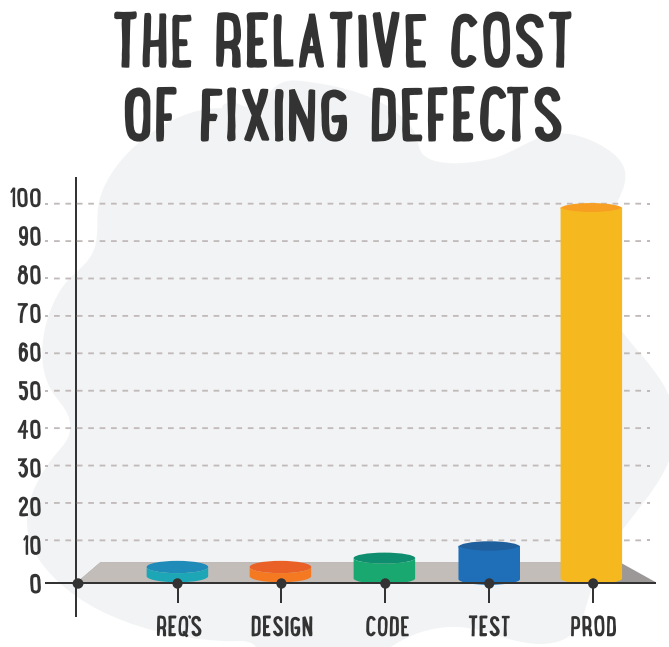


Figure 8

THE IMPORTANCE OF REVIEWS

"It is better to try to keep a bad thing from happening than it is to fix the bad thing once it has happened." This proverb definitely applies to defects in the software development life cycle. IEEE Software reports that rigorous reviews commonly remove up to 90% of errors from a software product before the first test case is run. Rigorous reviews are more effective, and more cost effective, than any other error-removal strategy, including testing. But they cannot and should not replace testing (IEEE, 2001). The Systems Sciences Institute at IBM has reported that the cost to fix an error found after product release was four to five times as much as one uncovered during design, and up to 100 times more than one iden-

tified in the maintenance phase (Figure 9). A review or inspection during the design phase can identify a significant number of errors. According to Crosstalk, the Journal of Defense Software Engineering, most failures in software products are due to errors in the requirements and design phases – as high as 64 percent of total defect costs (Figure 10).

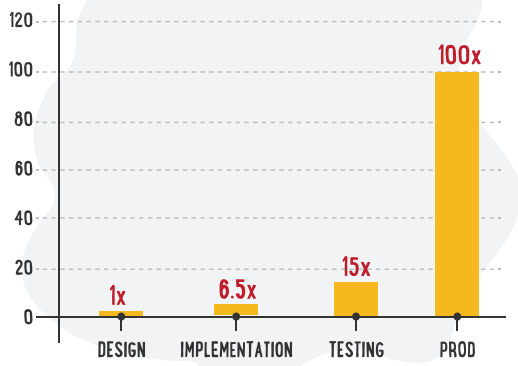


Figure 9

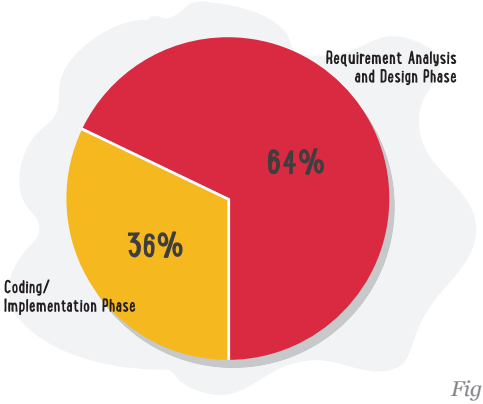


Figure 10

SEGUE INCLUDES QC THROUGHOUT THE PRODUCT LIFECYCLE

It can be concluded that defect identification is an important process in software development. At Segue, we conduct reviews to evaluate all phases of a project or software development. We focus our efforts on reviews conducted during the earlier stages of software development for defect detection that are more cost efficient than those at the later stages. Early defect detection allows our project managers to achieve the shortest schedules of projects and provide high quality products to the customers.

At Segue Technologies we believe that quality control activities should be implemented throughout the product life cycle, not just at the end. For more information on when to begin quality control, please check out my blog, For Great Quality, Bring Your Software Testers in Early.

Chapter 4

Process Improvement

Why is CMMI Appraisal Important for Software Development Companies?

As a software development professional that has been in the IT industry for over 30 years, I think it is fair to say I have a qualified perspective on the effect that process and standardization have on producing software products and services. Back in the day, however, I didn't always understand the need for these formalities. I would ask questions like, "Is it really necessary to 'document' my code?" or "Do we really need to do 'that' on 'all' of our deliverables?" or say, "I can get so much more done if I didn't need to worry about 'dotting the I's and crossing the T's.'" I make sure my code is documented and tested; why do I have

to consider other approaches?" It all just seemed to be a waste of time and money to me.

Well, that was me as a coder, many moons ago, and I'm sure that these types of concerns still resonate with others today. Now I have a different perspective as an owner and partner of Segue Technologies, and the current Director of Software Engineering. There are a whole host of challenges that arise with answering those core questions while still maintaining productivity and quality across the enterprise of services and development platforms. As a business owner, I am fully cognizant of the acronym ROI (Return on Investment) and how most decisions, if not all, are based on what the ROI expectations tell us. This both applies to the operations and success of Segue, and to the value and quality of our services for our customers.

Can a balance be achieved between administration and management costs and just "cranking out the code?" Should a formal analysis be conducted and investments made with the hope of the coveted ROI? Or was I right in my junior years? As an organization, you can start to find this balance by delving into process organizations whose whole mission in life is centered on identifying everything under the sun coupled to "best practices." You know who they are: ISO, ITIL, Six Sigma, IEEE, and a host of others. They all have white-papers and checklists that are truly inspirational and offer practical guidance in possibly achieving that necessary "balance."

WHAT IS CMMI?

CMMI stands for "Capability Maturity Model Integration" and is a model that provides appraisal and training for process improvement in organizations. CMMI was created by the Software Engineering Institute (SEI) as a result of research into organizations that would consistently deliver quality software on time and within budget. SEI wanted to know what it was that distinguished these high performing organizations from the majority who produced inconsistent or failed results. The CMMI developed as a documentation of the attributes shared by the successful organizations. Additionally, it organizes these attributes into Practice Areas, each with goals and specific practices that form a roadmap to how to become one of these high performing organizations. CMMI can improve capability in people, processes in a single project, department, or entire organization.

THE BENEFITS OF CMMI ACCREDITATION

Through experience and (dare I say) wisdom, I have seen firsthand the benefits that CMMI accreditation has brought to our organization, to include:

1. Expectation of Sustainment for the long haul
2. A balance between productivity and process by aligning our internal processes, with the Process Areas called out in the CMMI guidance.

Repeatability is the key for process success, and as they have now become the norm and not painful, everyone is in sync with the

expectations. Without an appraisal, it is doubtful that our organization would have implemented a sustainable and measurable process that everyone is aware of and that drives operations. We have seen ROI through optimization of tasks, estimation techniques, ownership/accountability, management, and how all corporate components play a role in quality through a consistent work-flow, just to list a few!

WHAT TO EXPECT DURING A CMMI APPRAISAL

The beauty of CMMI is its different levels of maturity. These help guide an organization to continually improve and implement practices in chunks rather than having to deal with everything at once. We can now progress and refine improvement as we mature by building on the CMMI foundation we have established.

Figure 11

The more mature your organization becomes, following the CMMI progression/capability levels, the more you become competitive through optimization and quality. Most important to my business and to the value I can provide my customers, is continually improving ROI.

What Does it Mean to be Appraised as CMMI-DEV Level 3?

Software Development can cover a broad range of projects with a wide range of complexity and costs. From a quick project that can be completed in a week with a single knowledgeable developer, like a simple website, to multi-year enterprise development projects with large teams broken into requirements, development, testing, and other key areas. A lot of companies provide Software Development as one of their service areas, but even looking at their experience and self-described capabilities, how can a customer tell if their development contractor has a proven and repeatable approach? This blog will describe what it means for a Software Development company to be appraised as CMMI-DEV v1.3 (Staged): Maturity Level 3 and the benefits of conducting business with a company that has been appraised at CMMI level 3.

WHAT IS CMMI-DEV?

The CMMI version 1.3 contains three "constellations" or models: services (CMMI-SVC), acquisition (CMMI-ACQ), and development (CMMI-DEV). The CMMI-SVC model provides guidance for organizations that provide services within their organization and to external customers. The CMMI-ACQ model provides guidance to organizations to enable their leadership to make informed and decisive acquisitions. The CMMI-DEV model is used for process

improvement in organizations that develop products. CMMI for Development contains practices that cover project management, process management, systems engineering, hardware engineering, software engineering, and other supporting processes used in development and maintenance (CMMI Institute).

CMMI-DEV provides guidance for process improvement across a project, department, or organization that will lead to lower costs, improved quality, and on time delivery of products and services. CMMI-DEV guidance covers the life cycles of products from conception through delivery and maintenance. CMMI-DEV best practices are flexible enough to apply to a variety of industries, yet stable and consistent enough to provide a benchmark against which your organization can measure and compare itself. Therefore, when a company is appraised at CMMI for Development, it means that the company has been trained, assessed, and appraised in the areas of product and service development.

Within CMMI-DEV there are five maturity levels, of which an organization can be appraised at level 2 through 5. Below is a graphic representation of what each of the levels mean:

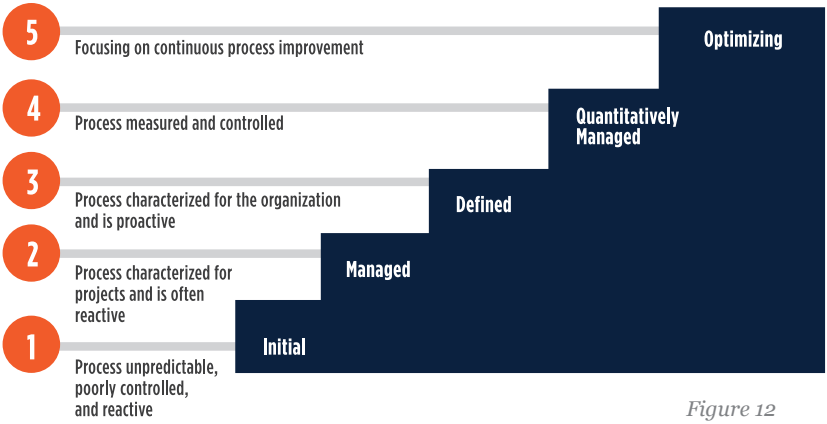


Figure 12

In order to be appraised at a level, an organization must meet all the Process Areas within that level and all the Process Areas in the levels below:

Maturity Level 2 (ML 2) in CMMI-DEV includes the following Process Areas:

- Configuration Management (CM)
- Measurement and Analysis (MA)
- Project Monitoring and Control (PMC)
- Project Planning (PP)
- Process and Product Quality Assurance (PPQA)
- Requirements Management (REQM)
- Supplier Agreement Management (SAM)

Maturity Level 3 (ML 3) in CMMI-DEV includes the following Process Areas:

- Decision Analysis and Resolution (DAR)
- Integrated Project Management (IPM)
- Organizational Process Definition (OPD)
- Organizational Process Focus (OPF)
- Organizational Training (OT)
- Product Integration (RD)
- Requirements Development (RD)
- Risk Management (RSKM)
- Technical Solution (TS)
- Validation (VAL)
- Verification (VER)

When Segue Technologies was appraised CMMI-DEV v1.3 (Staged): Maturity Level 3, it means that we have a detailed process that guides the product lifecycle from its conception throughout to its delivery and maintenance (Institute, 2014), which is shown by our adherence to the Process Areas in both ML 2 and ML 3. It means that, as a company, Segue uses the CMMI best practices for its product and service development.

WHY YOU SHOULD DO BUSINESS WITH A COMPANY THAT IS APPRAISED FOR CMMI?

Doing business with a company that is appraised for CMMI for development has many advantages. First, since CMMI for development leads to better quality products, doing business with such a company means that the products provided will be of high quality.

Another benefit of doing business with such a company is that they can provide more accurate schedules and realistic timelines, leading to more realistic deadlines for product releases. The CMMI-DEV will therefore provide the best practices throughout the product lifecycle to ensure timely delivery and quality products.

Figure 12

ABOUT THE AUTHORS

LaTonya Pearson is the Quality Control Manager for Segue Technologies. She is responsible for assuring that software development complies with the Quality standards defined by Segue Technologies and follows the CMMI recommendations. The QC Manager is also responsible for managing QC resources, assessing product risks, and test planning. They will regularly provide status reports regarding product development and process performance to the Board of Directors. Monitors and improves the Quality Control Process. Evaluates risk, based on the testing effort and need. Defines and manages test planning. Defines testing priorities, approach, methodology, and strategies. Manages testing resources and assigns testing roles and responsibilities to the QC team. Works directly with project managers to solve any organizational issues. Provides training and education material to the QC team. LaTonya Pearson has a Bachelor's of Science and a Master's degree in Business Administration. She also has a Certification for Capability Maturity Model Integration (CMMI) for Development, as well as a Certification from the American Software Testing Qualifications Board.

Ann Millikin is the Quality Control Test Automation Engineer for Segue Technologies. She is responsible for creating automated test scripts to increase the effectiveness, efficiency and coverage of your software testing and to perform other repetitive tasks such as data loading. When she's not writing automated scripts using a combination of Selenium, Java, and TestNG, Ann Millikin develops requirements detailing the specifics for the software developers. Ann Millikin has a Bachelor's of Science in Management Science and Decision Support Systems from Va Tech. She also has completed software testing and java courses.

Mike Behrmann has over 30 years of experience in software development and design, database architectures, and emerging Internet and enterprise-computing technologies. His IT roles have included software engineer, database architect, and project/program manager. He began his career as a computer programmer in the mainframe environment during the early 1980s and transitioned through the DOS and MS Windows evolution to the web development choices of today. Mike has guided the establishment of an Agile development approach, and formalized the Requirements and Testing teams of Segue's Software Engineering division. His efforts in process definition and continual improvement have resulted in Segue's CMMI-Dev Level 3 appraisal.

ABOUT QUALITY CONTROL:

Thank you for reading our eBook. We hope you found it informative and interesting. If you are interested in working with Segue, please contact us so we can learn about your needs and plan a development path that works for you.

Segue Technologies provides:

Web

- Custom Application Development
- User Interface Design
- Content Management Systems

Data

- Business Intelligence
- Data Quality and Profiling
- Enterprise Data Management

Mobile

- iOS Development
- Android Development
- Mobile Web