

# A Case Study in Extreme Quality Assurance (XQA)

By James A. Canter and Liz Derr

<b>1</b>	<b>INTRODUCTION</b> .....	<b>1</b>
<b>2</b>	<b>THE PROJECTS</b> .....	<b>2</b>
<b>3</b>	<b>THE TRADITIONAL APPROACH</b> .....	<b>3</b>
3.1	TEST DEFINITION AND AUTHORSHIP .....	3
3.2	PROCESS FLOW.....	4
<b>4</b>	<b>FAILURE POINTS IN THE TRADITIONAL APPROACH</b> .....	<b>5</b>
4.1	LACK OF CLEAR, UNAMBIGUOUS, AND COMPLETE REQUIREMENTS .....	5
4.2	LACK OF FUNCTIONAL TEST COVERAGE.....	6
4.3	HIGHLY DEPENDANT UPON INDIVIDUALS .....	6
4.4	PROCESS FLOW MAKES QA PRACTICE VULNERABLE TO SLIPPAGE .....	6
4.4.1	<i>Lack of Process Awareness Across the Organization</i> .....	7
4.4.2	<i>Lack of Communication Among Functional Groups</i> .....	7
4.4.3	<i>Change Requests</i> .....	8
4.4.4	<i>Late QA Involvement in Development Projects</i> .....	8
4.5	TEST EXECUTION NOT SCALABLE .....	8
4.5.1	<i>Manual Testing Too Expensive</i> .....	8
4.5.2	<i>Off-The-Shelf Automation Inadequate</i> .....	8
4.6	TESTING THE LEGACY APPLICATION .....	9
<b>5</b>	<b>GOVERNMENT &amp; INDUSTRY APPROACH</b> .....	<b>10</b>
<b>6</b>	<b>IMPLEMENTATION OF XQA AT EGREETINGS NETWORK, INC.</b> .....	<b>11</b>
6.1	REQUIREMENTS IMPROVED .....	11
6.2	TEST COVERAGE IMPROVED.....	13
6.3	SLIPPAGE MINIMIZED .....	14
6.4	TEST EXECUTION MADE SCALABLE .....	15
6.5	PROCESS ACCELERATED .....	17
6.6	TEST ANALYSTS MORE EFFICIENT .....	19
6.7	SOFTWARE DEVELOPERS MORE EFFICIENT.....	19
6.8	REGRESSION TESTING COSTS REDUCED .....	20
6.9	MANAGEMENT OBSERVABILITY IN SUPPORT CMM GOALS .....	21
6.9.1	<i>Early Risk Identification</i> .....	21
6.9.2	<i>Systematic Assessment of Requirements Quality</i> .....	21
6.9.3	<i>Systematic Assessment of Process Quality</i> .....	22
6.9.4	<i>Project Progress</i> .....	22

## 1 Introduction

Extreme Programming (XP) approaches have gained significant attention recently. This has been spurred by business' motivation is to decrease time-to-market in order to maximize revenue opportunity.

In this quest, without quality practices that keep pace, new releases deliver defects to customers just as quickly!

While Egreetings had not embarked on the journey toward XP methodology, quality practices that emphasized testing first were employed with dramatic results. These practices and techniques could easily fit the XP model.

# A Case Study in Extreme Quality Assurance (XQA)

By James A. Canter and Liz Derr

XP methodology frames requirements as “stories”, then emphasizes customer authorship of Acceptance Tests (functional tests) that are made ready for automated testing as soon as possible, preferably before code is ready. It advocates the use of one or more “testers” to assist the customer in defining and automating these tests.

Except that Egreetings used formal Use Cases, this model was followed. In an XP environment, these practices could be easily adapted to stories, using “pair programming” principles with the customer to author CEG prototypes that generate high-coverage user-acceptance or “functional” tests.

This paper explains how these advanced quality practices succeeded at Egreetings.com, a top 50 website. It’s also apparent that these practices can be made applicable to non XP practices with impressive results.

## 2 The Projects

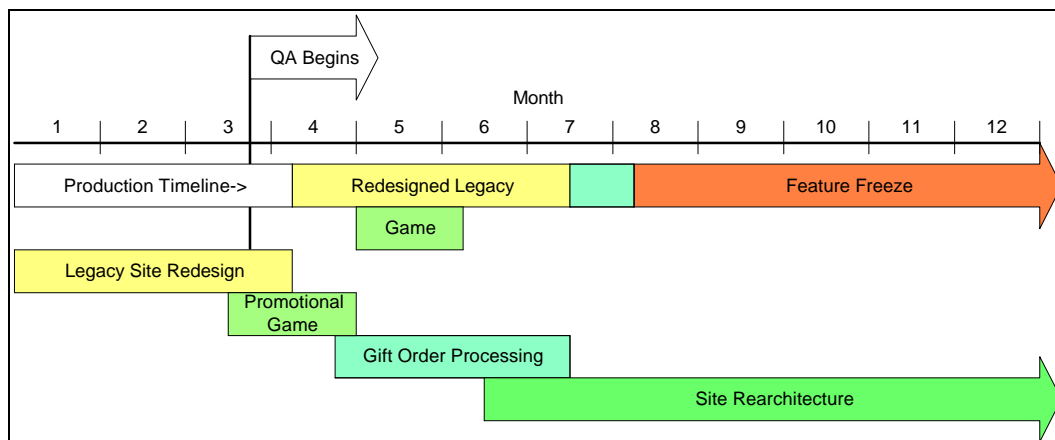
At Egreetings.com, a Quality Assurance Department was formed to establish software development, requirements authorship, application deployment, and quality verification practices. All while business imperatives demanded launch of a full site redesign, credit card processing for gift sales, and then development and launch of a fully re-architected application.

Rearchitecture’s goal was to provide a site that could sustain daily content releases and weekly feature releases. But before the rearchitected site was to be launched, the company needed to enhance their revenue stream by offering gift-purchasing capabilities.

So the legacy application was to be enhanced with additional gift functionality while the site was being re-architected for a later launch. All of the new functionality was to be folded into the new site orthogonally.

Here is a timeline of the major projects and events:

*figure a, part i*



## A Case Study in Extreme Quality Assurance (XQA)

By James A. Canter and Liz Derr

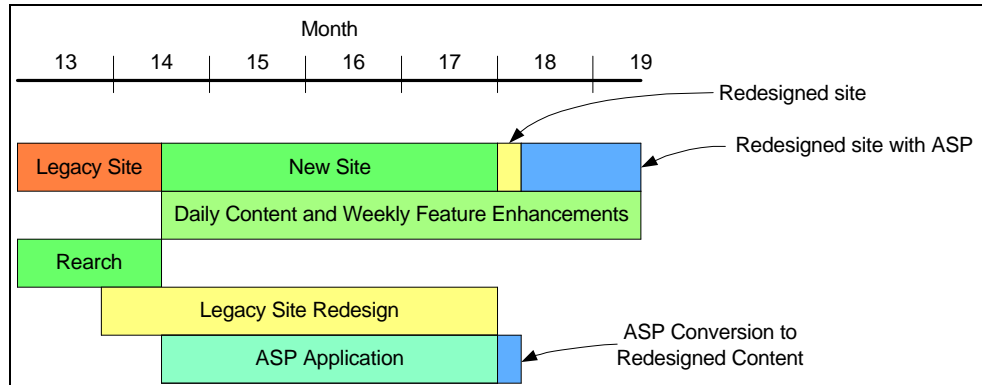


figure a, part ii

The legacy site Redesign Project launched with over 80 undiscovered defects. The average number of defects discovered post-launch during this period was about 25. One permanent and three temporary testers performed ad-hoc testing of releases until midway through the 4<sup>th</sup> month.

### 3 The Traditional Approach

#### 3.1 Test Definition and Authorship

For Egreetings Network, Inc., not having engaged in the evolutionary process improvement inherent in the Capability Maturity Model (CMM)<sup>1</sup>, the approach to quality development could be characterized as:

- Our engineers did some testing
- We used domain experts to run through our software
- The whole company checked out the new releases
- We used a test check sheet to make sure we're okay

Tests were designed intuitively, on the fly, relying on the testers' domain expertise. Company wide testing depended upon the initiative of individual testers during a specifically scheduled test window. And there were no requirements from which to derive test cases.

Perhaps the most widespread practice is to develop test cases through "Intuitive Decomposition", where the test analysts asks the question, "How can I break this?" Good test analysts take pride in their ability to sense where defects will occur.

A more disciplined form of test design involves "hierarchical functional decomposition." Here a functional breakdown starts with a hierarchical list of business functions. A

<sup>1</sup> CMM is a product of the Software Engineering Institute established by Carnegie Mellon University. It provides a maturity model that describes an organization's ability to achieve software launch and quality goals predictably.

## A Case Study in Extreme Quality Assurance (XQA)

By James A. Canter and Liz Derr

business function is the smallest, discrete functional component of a system. In UML practices they are normally referred to with a verb-object description, such as “Place Order”, “Login User”, or “Register User.”

Once these low level business functions have been identified, tests cases are designed intuitively and through use of established, written standards adopted by the enterprise.

Checking against the organizational standards helps assure test analysts that they haven’t missed anything.

But this takes time, and quality organizations in immature enterprises are always crunched for testing time, let alone test case design!

### 3.2 Process Flow

Classically, development and testing process can be characterized like a waterfall<sup>2</sup>, or semi-waterfall. Tests are designed and executed after the software is delivered for test. If tests are ever automated using off-the-shelf automation tools, this usually occurs after launch.

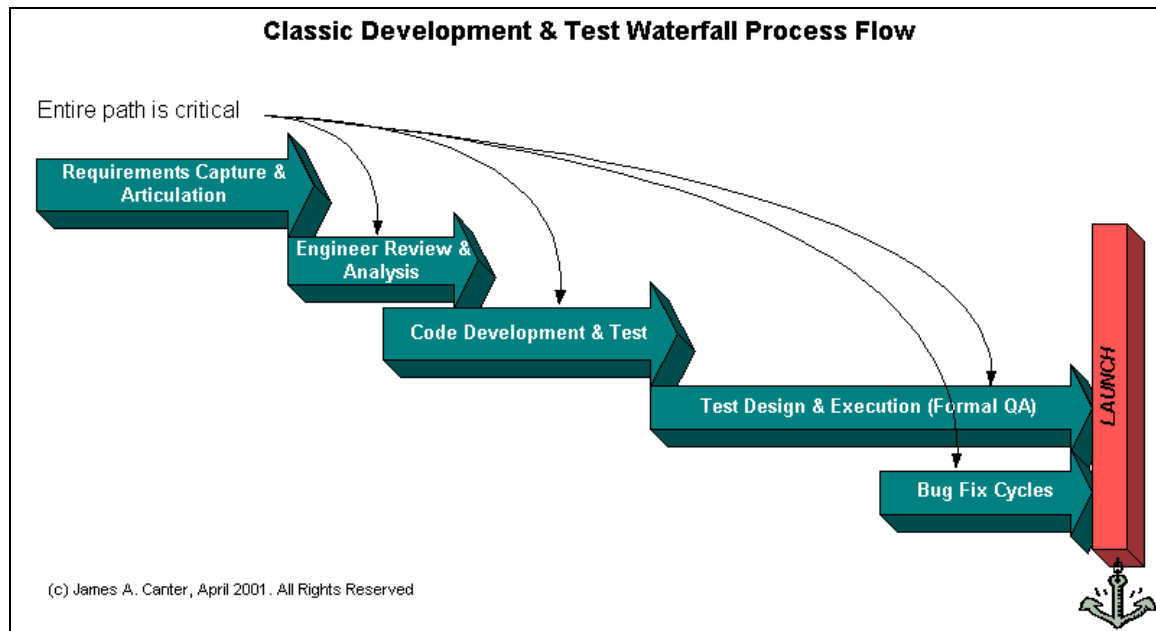


figure b

<sup>2</sup> Waterfall describes a process where an operation can begin only after the previous operation is complete, thus each step in the waterfall is dependent upon completion of its predecessor.

# A Case Study in Extreme Quality Assurance (XQA)

By James A. Canter and Liz Derr

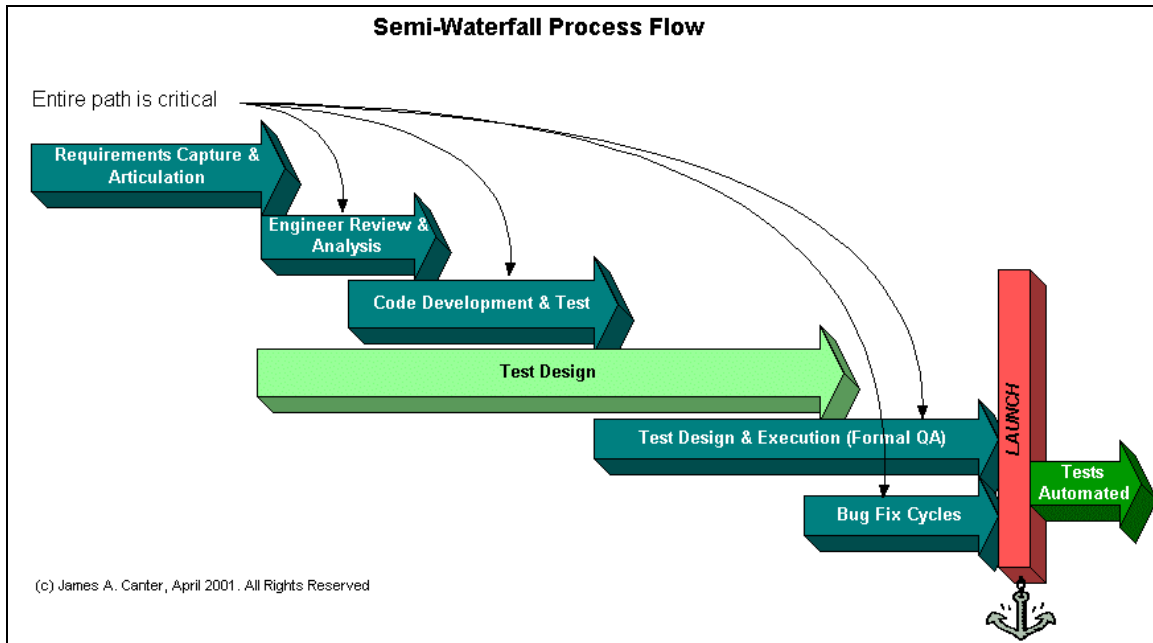


figure c

More mature organizations provide for early quality assurance after the requirements are complete. This is represented in figure c above. Test analysts are afforded the opportunity to analyze the requirements in the course of designing test cases. This allows the enterprise to engage in more disciplined testing after the application is delivered.

The delineation between test design and execution is not so crisp. Often, test analysts must “play” with the new application to fill out their test suite.

As you can see, the only parallel effort depicted is Test Design. In most organizations this effort benefits mostly from test analysts experience, domain knowledge, and their intuitive prowess in breaking applications.

## 4 Failure Points in the Traditional Approach

All aspects of these failure points were present at Egreetings while using traditional approaches to test the legacy site.

### 4.1 Lack of Clear, Unambiguous, and Complete Requirements

Effectiveness of the *most* disciplined form of analysis and intuitive decomposition depends upon well-articulated and unambiguous requirements. Without a mature requirements authorship practice, analysts’ testing experience must be the basis of reliable defect discovery through complete and comprehensive test scripts.

## **A Case Study in Extreme Quality Assurance (XQA)**

*By James A. Canter and Liz Derr*

The problem is that organizations with mature requirements articulation capabilities will have evolved in large part with contributions from a mature Quality Assurance organization. Enterprises starting at CMM level 1<sup>3</sup> don't have the knowledge, process or techniques in place to provide such requirements to engineers and test analysts.

So with the best test analysts, major problems can go undetected before launch resulting in project failure, if not significant cost and time-to-market overruns.

### **4.2 Lack of Functional Test Coverage**

Let's assume an organization is near achievement of CMM level 2<sup>4</sup>, and has a solid requirements capture and articulation practice in place. Early decomposition of requirements into test cases helps discovery of serious design flaws and missing functionality early enough to minimize significant time and cost impacts. Statistics have shown that as much as 56% of defects discovered in an application can be traced to requirements<sup>5</sup>. But intuitive decomposition still relies on the analyst's testing experience.

Often those engaged in intuitive decomposition can find themselves wrapped in a complex logical pathway that seems to have endless combinations and permutations. For example, let's say that an application component has 37 inputs. An exhaustive test suite would cover 130 billion possible combinations. Which tests should be implemented to remain within reasonable time and cost constraints?

### **4.3 Highly Dependant upon Individuals**

This aspect ties heavily with the previous paragraph. Test design based on intuitive decomposition requires experienced test analysts who know how to identify and hone in on vulnerable parts of the application.

In fact, at Egreetings, only one or two engineers had all of the domain knowledge regarding functional components of the application.

### **4.4 Process Flow Makes QA Practice Vulnerable to Slippage**

Timeline crunches are notorious for sacrificing the overall quality of testing for the sake of accelerating time-to-market. Time-to-market is almost always driven by the need to open new or enhance existing revenue opportunities.

---

<sup>3</sup> CMM Level 1 organizations are those who have not yet embarked on process improvement. Their software effort can be characterized as ad-hoc, and sometimes chaotic. In such organizations, few processes are defined, and successful implementations can be attributed to individual effort as well as heroics.

<sup>4</sup> CMM Level 2 organizations have basic project management processes in place to track cost, schedule, and functionality. Process discipline has been established to repeat earlier successes in projects associated with similar applications.

<sup>5</sup> Requirements-Based Tutorial, Bender & Associates, January 1999

## A Case Study in Extreme Quality Assurance (XQA)

By James A. Canter and Liz Derr

Sometimes business proponents don't fully appreciate that delivering poor quality to market can be more damaging to a revenue opportunity than delaying the launch. Can you really say you made it to market on a specific date when the features must be pulled for repair, then re-released at a later time?

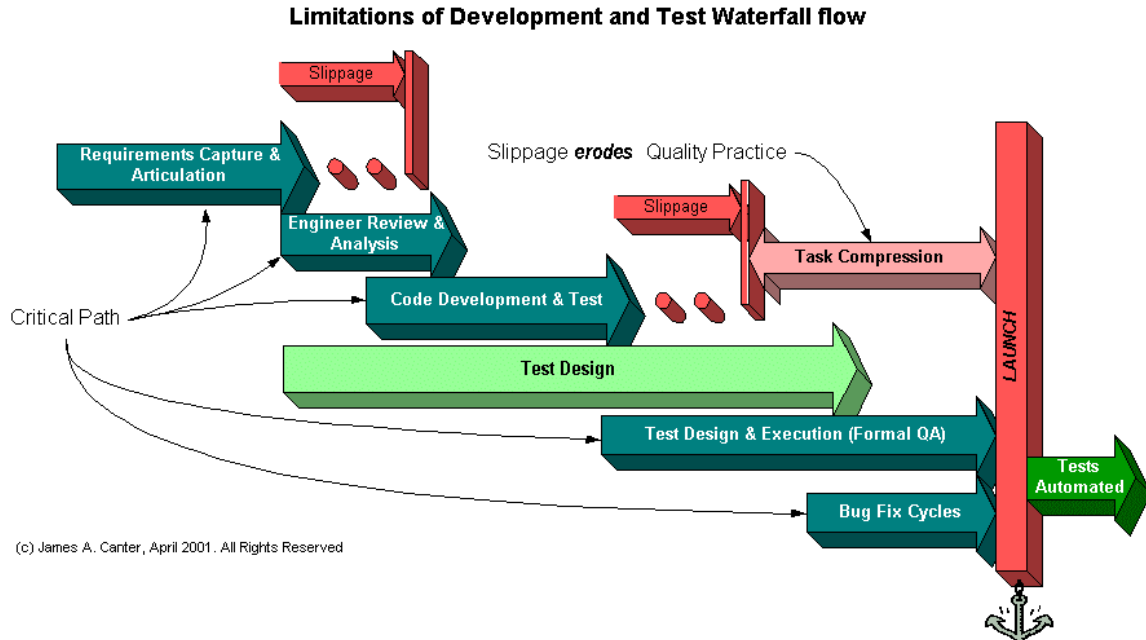


figure d

As seen in *figure d*, slippage can occur regarding design and coding. When slippage occurs, the quality practice is compressed resulting in tests that don't apply to current application functionality. This may be immaterial, since slippage can also prevent completion of the entire test cycle.

Slippage occurs as a result of these major factors.

### 4.4.1 Lack of Process Awareness Across the Organization

Various stakeholder groups tend to act on their own without even knowing what must be communicated to other stakeholders to keep everyone on track. Sometimes one stakeholder group doesn't even know who the other stakeholders are.

### 4.4.2 Lack of Communication Among Functional Groups

As engineers work with architects and business proponents on new features, design issues and incompatibility with legacy components force revisions. A developer who lacks high-level domain knowledge about the application sometimes makes revisions. These changes may or may not be communicated to business proponents or QA. Thus, if the organization is lucky they're caught during quality testing.

## A Case Study in Extreme Quality Assurance (XQA)

By James A. Canter and Liz Derr

Developers and test analysts interpret the requirements differently. This causes either false defects to be identified and provokes possibly extensive test case revisions, or identifies a major incorrect logic path taken by the developer (very expensive).

### 4.4.3 Change Requests

Business proponents discover changes or additional features that could have a large impact on revenue once the application is launched. These and other design revisions are often reflected in change requests.

If change request communications are excellent, test analysts *may* be able to modify their test suite quickly enough to test the modified application, but unless the operation is mature – probably not. Only in mature enterprises all potential major failures are discovered *before* launch.

### 4.4.4 Late QA Involvement in Development Projects

Lack of testing can cause slippage even when the launch date is met. This occurs when customers find enough serious errors as to force rollback to the previous version.

## 4.5 Test Execution Not Scalable

### 4.5.1 Manual Testing Too Expensive

It was clear that using the Cause-Effect Graphing discipline at Egreetings generated useful test scripts that significantly improved both timelines (by uncovering requirements defects before coding began) and quality.

Additionally, it was clear that repeated manual execution of the effective but growing test library would become cost prohibitive. Without an automated test library, these effective tests could not be brought to bear on subsequent releases. And daily content releases with weekly and semi-weekly feature releases were primary goals of Rearchitecture.

### 4.5.2 Off-The-Shelf Automation Inadequate

Having purchased off-the-shelf testing and test management tools, it quickly became apparent that the 300-odd test cases implemented for Gift Order Processing could not be maintained. This was due to the fact that QA had no experienced test automation technologists and had to rely on record-playback disciplines. Since the test tool accessed functionality through manipulation and observation of the content layer (black-box testing), frequent content changes would defeat its use. The only way to maintain the automated test library was to

- 1) Wait until the application was delivered,
- 2) Rerun all recorded tests,
- 3) Re-record all failed tests,

# A Case Study in Extreme Quality Assurance (XQA)

By James A. Canter and Liz Derr

- 4) Record all new tests.

## 4.6 Testing the Legacy Application

First, QA covered the existing application using traditional methods, relying on experienced test analysts who had learned the application to regression test releases that were delivered in two or three week intervals. Test coverage was improved the expensive way, by adding bodies. Individuals also kept their own test check sheets and added to them based on post-launch defect discovery.

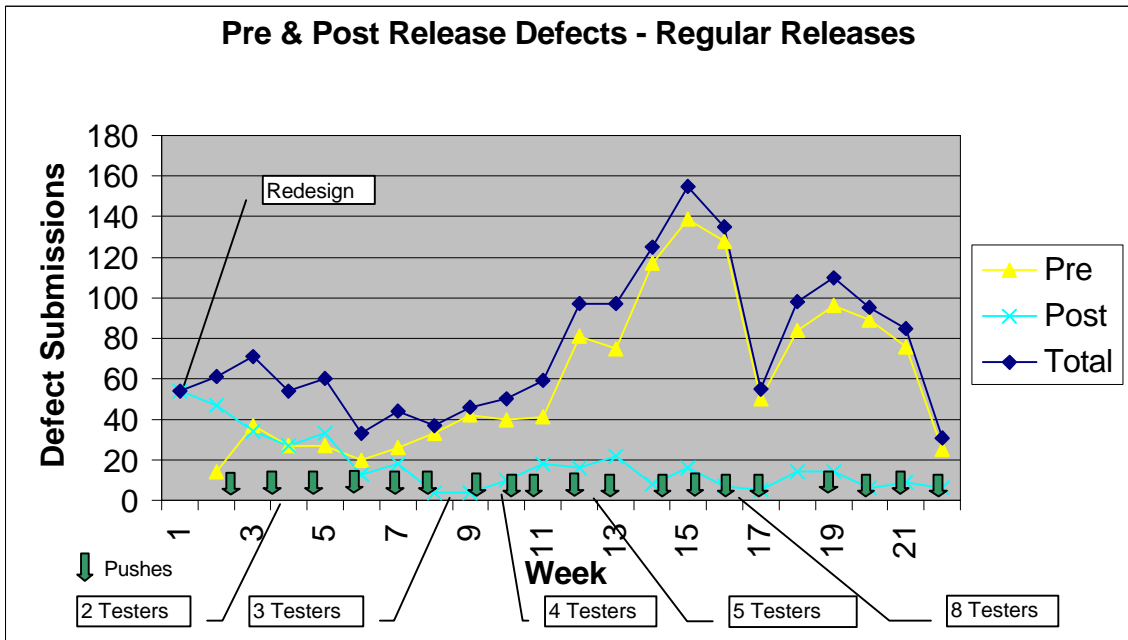


figure e

This covered the legacy site releases from Legacy Redesign launch through the feature freeze that commenced at the beginning of month 8.

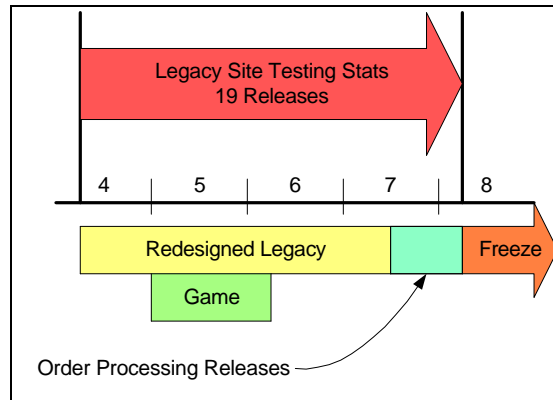


figure f

## A Case Study in Extreme Quality Assurance (XQA)

By James A. Canter and Liz Derr

Figure g shows pre and post defect discovery to the site as a whole. Installation of the Gift Order Processing functionality occurred near the end of this period (about week 12), while several other features were released in parallel. All tests were manually executed.

These statistics, which did not differentiate among application and deployment errors, averaged 7 for the last three weeks. Accounting for the deployment problems at that time, it would be easy to justify an adjustment of the number to 5 or 6.

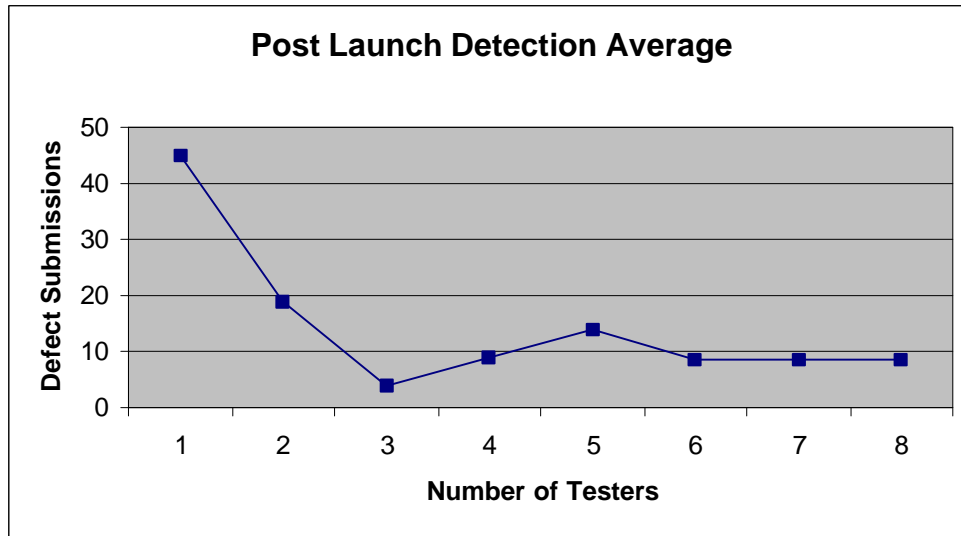


figure g

Another trend can be seen in the approximate averages produced after assignment of additional testers. Assignment of testers 5, 6, 7, and 8 did *not* materially increase regression test coverage for the remaining 12 releases. In fact, if the number of anticipated projects had not increased in the queue, there would have been no need for these additional test analysts.

This trend affirms the limitations of intuitive test design. Experienced test analysts who documented their own tests failed to penetrate a barrier to fewer than 5 average undetected defects per release.

## 5 Government & Industry Approach

Where enterprises develop applications with no large financial risk, or risk to life and limb, the business imperative is to launch as early as possible, then fix problems discovered in production as soon as possible. For systems that are developed to operate machinery (missiles, the Space Shuttle, aircraft, and marine systems) intolerance for error has led to the development of the Software Engineering Institute and its Capability Maturity Model at Carnegie Mellon University.

Generally, business organizations form effective Quality assurance programs only after they have suffered enough pain (cost overruns), or lost enough customers (results that don't justify the investment).

## **A Case Study in Extreme Quality Assurance (XQA)**

*By James A. Canter and Liz Derr*

There has been reluctance by business to adopt CMM disciplines because of their perceived cost and/or establishment of “red tape” within the organization.

In CMM, actual processes are not the focus, but *management observability* of adopted processes and capabilities. CMM’s 5 levels of capability define evolutionary milestones to the “ultimate” practice.

When engaging in this purposeful evolution, organizations are free to use the project management, requirements management, development and test practices and tools that best suit their needs.

The fact is that organizations that move up the capability levels identify, and then meet different, more sophisticated needs that support management’s ability to observe performance at deeper and deeper levels of detail.

In the interest of running more efficiently, thus delivering very high quality products to market in shorter time frames, organizations must have the will to make the investment in their infrastructure. Extensive organizational training is required even beyond classrooms and canned learning courses. Mentorship focused by defined training requirements for specific disciplines must also be realized.

One could say that the main result of achieving successive CMM levels is organizational transformation. It involves the birth and growth of a quality-focused culture with the will to evolve.

So how does VP of Engineering or a CTO of a young enterprise embark on this path, especially when the Business Proponents don’t understand its importance? How does one recruit business proponents who don’t want a bunch of red tape delaying critical delivery and revenue opportunities?

## **6 Implementation of XQA at Egreetings Network, Inc.**

### ***6.1 Requirements Improved***

At Egreetings, simple processes were put in place that promoted the Business’ interest by graphically demonstrating risks early in the development cycle. Here’s an answer to the question about how VP’s and/or CTO’s embark on a process improvement path, and how Egreetings embarked on a path toward CMM level 2.

Requirements were a new concept at Egreetings Network, Inc. With a Quality Assurance department in place, it became imperative for business proponents to provide requirements. The awful quality of the site had been sufficient to motivate proponents to do this.

## A Case Study in Extreme Quality Assurance (XQA)

By James A. Canter and Liz Derr

Because the organization was immature in capturing and articulating requirements, initially a simple narrative describing content accompanied with a flow diagram were acceptable.

Classic intuitive decomposition was *not* used for new projects. Instead test analysts used a relatively old, but undiscovered technique for modeling the logical paths of the requirements.

Developed at IBM over 30 years ago for the purpose of testing logic gate functionality on processing chips, the technique was adapted successfully for black-box software testing<sup>6</sup>. Central to this practice is development of a Cause-Effect Graph (CEG) which effectively prototypes the state flow for proposed features. Such graphing successfully captures logical relationships, including constraints between inputs and outputs.

A major benefit over traditional intuitive decomposition is that the rigor introduced through authorship of the CEG has a definite end: when the graph is complete, it successfully describes the functionality articulated in requirements. This is a very apparent and verifiable milestone. So unlike the intuitive approach, additional tests are rarely, if ever discovered after launch.

That means test coverage is complete. And completion of test design with reliable coverage becomes a known milestone. Under established process and consistent practice, the definition of reliable and comprehensive tests grows to be more predictable. Thus project management implications became *very* significant.

Rigor achieved by this discipline drives discovery of inconsistencies, ambiguities, and omissions in requirements documents. Thus, a reliable and effective flow of discussion regarding requirements problems and their resolution is initiated.

Additionally, completion of CEG analysis is easily achieved before code completion by experienced practitioners when conducted in parallel to engineering analysis in support of actual code development.

With this feedback, business authors were able to learn what was needed to produce better requirements *before* developers used those requirements to code.

In the “Game” promotional project, three major design flaws were discovered before coding began. The business proponents learned how to effectively articulate requirements using flow diagrams and screen shots. They found that future promotional projects were cheaper and easier to manage.

Next, Use Case authorship was taught. Use cases were first used on the Gift Order Processing project. It was this project that defined the enterprises’ transition from schematics and flow diagrams into carefully articulated use cases.

---

<sup>6</sup> Requirements-Based Tutorial, Bender & Associates, January 1999

## A Case Study in Extreme Quality Assurance (XQA)

By James A. Canter and Liz Derr

Gift Order Processing use cases were subjected to the same analytical inspection rigor as had been applied to the “Game” project. Lessons learned by use case authors carried over into use cases written for the Rearchitecture Project.

The largest factor in promoting and sustaining this evolutionary path was senior management’s support. Senior management, including the CTO, VP of Engineering, VP of Business Development, and SVP of Marketing supported establishment of new processes and worked together to ensure progress could not be derailed by one functional organization.

### **6.2 Test Coverage Improved**

The same process used to improve requirements improved test coverage. Capturing the intended functionality in a logical model (cause-effect graph) points to holes in the requirements.

Using proven mathematical techniques, the model was also used to generate test cases that provided 100% functional coverage against the model with the fewest number of test cases.

Remember the example in paragraph 4.2 that talked about an application component with 37 inputs? This reflected 130 possible combinations. Applying defined and accepted mathematical algorithms against functionality expressed in the CEG results in 22 tests: a very manageable test suite.

So the technique facilitated improved requirements while dramatically improving test coverage. Projects subjected to this practice *consistently* launched with an average of less than one undetected defect per launch. This was true for all major projects launched after Legacy Redesign.

Here’s the defect history for the Gift Order Processing project.

## A Case Study in Extreme Quality Assurance (XQA)

By James A. Canter and Liz Derr

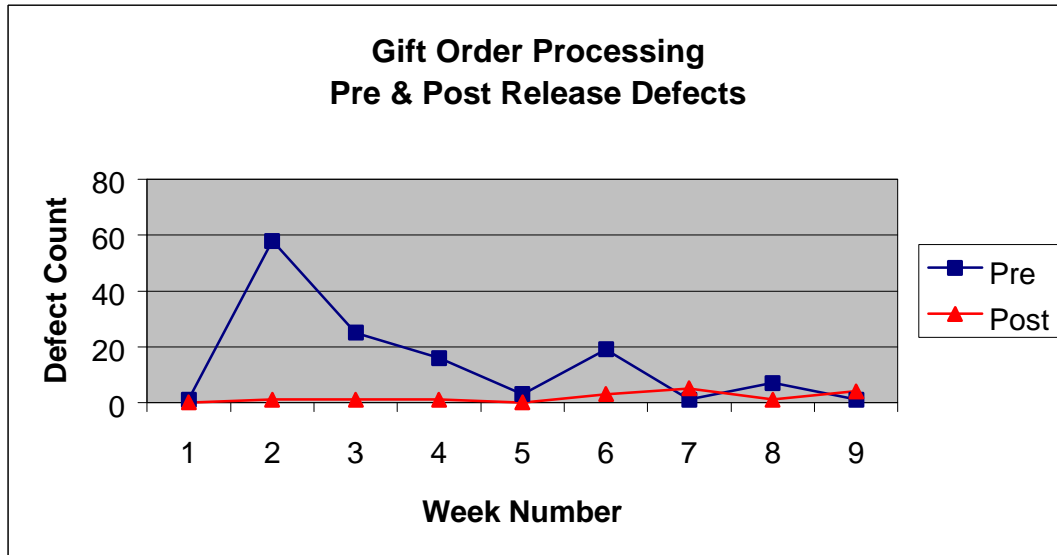


figure h

### 6.3 Slippage Minimized

As the team applied CEG practices and requirements improved, dialog commenced among QA test analysts, requirements authors, business proponents, and engineers. The dialog began on an ad-hoc level, later maturing to more formal forums.

The need to coordinate business proponents with QA, requirements authors, architects, and engineers culminated in the formation of a project office. The project office initiated standard process flows, roles and responsibilities and became the high-level shepherd assuring cross-functional communication and coordination.

Project and Product Managers were designated in each functional group and provided the project office with progress statistics and status. These specialized delegates to the project office also raised risks. This became an important means of ensuring quality issues received the attention they deserved.

The Cause-Effect Graphing process also provided a means for test analysts to identify and articulate high-risk requirements process issues in project status forums.

These practices matured considerably during the 8-month rearchitecture project serving to materially enhance

- Process awareness across the organization
- Communication and coordination among functional groups
- Effective distribution and approval of Change Requests

## A Case Study in Extreme Quality Assurance (XQA)

By James A. Canter and Liz Derr

Since test analysts had proven the effectiveness of early involvement in the Game, and Gift Order Processing projects, QA's early participation continued.

Because of their consistent and diligent involvement at the beginning of projects, test analysts were key to maintaining open communication and collaboration channels cross functionally.

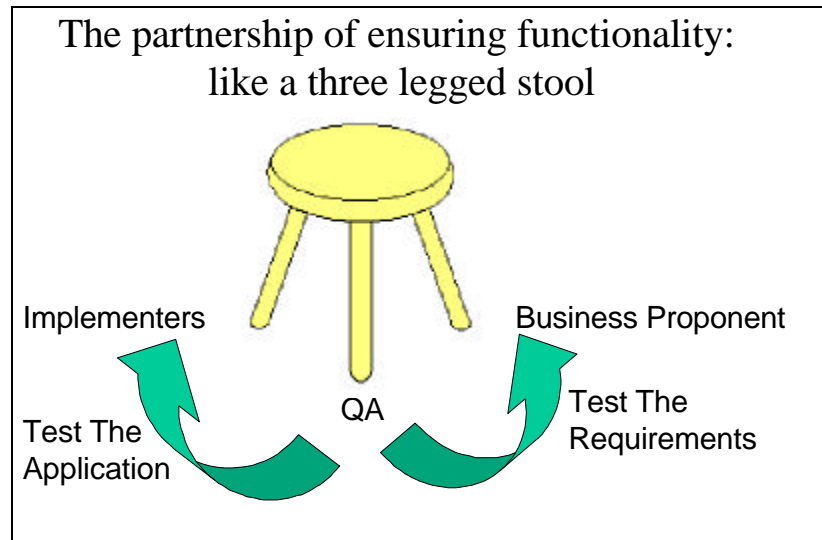


figure i

For the first time, the role of test analyst became valued in the enterprise. In the course of analyzing requirements for the ASP project, test analysts discovered nine major design defects before coding began. Had these defects gone undetected until test time, business proponents and engineers estimated that the project would have slipped by two months. The VP of Business Development was sure that Egreetings then would have lost a major customer.

So, working with the Project Office, test analysts engaging in these practices played a significant role in preventing slippage.

### 6.4 Test Execution Made Scalable

QA embarked upon a 10-month infrastructure development project. To be delivered concurrent to the Rearchitecture project, its goal was two-fold: Deliver a utility application that allowed for rapid implementation and modification of automated tests by test analysts unfamiliar with test automation scripting languages and in the absence of the application to be tested. That way automated testing wouldn't have to wait until after launch. Each new release would have automated tests from the regression suite *combined* with new tests targeting the current release.

## A Case Study in Extreme Quality Assurance (XQA)

By James A. Canter and Liz Derr

The basic architecture of the tool is to rely on database technology to associate test actions with screen objects. Test automation script segments would be stored as properties of actions on objects.

Test analysts would use a GUI interface to build a model of the “should be” view of the UI (User Interface) objects in the next release, derived from screen shot design documents.

Basic object classes were user-defined and named after MS Window and browser objects, for instance:

Edit Box	Check Box	Radio Button
Image	Push Button	Table
List Box	Scroll Bar	Screen
Link	Static Text	

Test analysts would select a UI object and place it into a collection of test steps. As the object was placed, the test analyst would pick the appropriate action for that step.

For example, logging in to the site required the entry of the user’s email address. The test analyst would select the “Email Address Edit Box” object and add it to a test step collection. Then the analyst would chose the “Edit\_Set” action, entering the user’s email address for the specific test. Both the instance of the object and the action were saved in the test step collection with the data to be entered into the edit box at test execution time.

The tool would select the script segments from the database and arrange them into an executable sequence based on the order of the test steps in the test collection. The assembled script was saved to disk in a way that was accessible by the 3<sup>rd</sup> party test automation tool.

Because database technology is effective in managing change, test analysts would use content requirements documents to change UI objects in the object model. These changes automatically modified the tests that depended on them. Where object classes changed or objects were eliminated, test analysts could generate an impact analysis report that allowed them to identify affected tests, then delete or modify them. This was accomplished *without* having to run the tests against an actual application.

The test utility module was launched one month before the rearchitected web site, hosting over 600 regression tests.

Ultimately, the team captured over 1600 regression tests and effectively modified them with an average of 2-4 person-hours in support of daily releases. This figure represents about 1-2 hours of a test automation technologist who is accomplished in the 3<sup>rd</sup> party automation tool’s scripting language in support of 1-2 hour effort by a test analyst.

## A Case Study in Extreme Quality Assurance (XQA)

By James A. Canter and Liz Derr

Additionally, test analysts had constructed over 300 automated tests against the ASP Application Project under the legacy site design (e.g. using legacy UI objects). It was not known until the last minute that the Site Redesign Project would be launched only two weeks before the ASP Application.

Redesign Launch would break a very large percentage of the some 1300 automated regression tests being run daily against the rearchitected site at the time. So the site was frozen for two weeks. The first week was used to modify server-side code to support the new client screens; the second week was for testing and bug fix cycles.

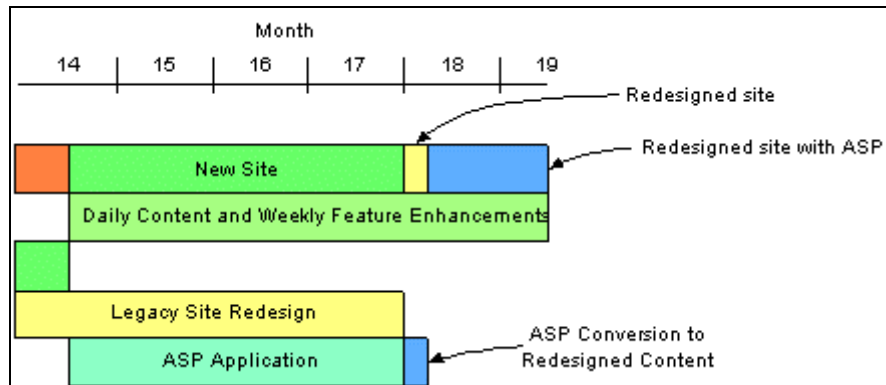


figure j

Two test analysts along with the test automation technologist converted all 1300 automated tests in 4 days. The tests were run daily during the second week, and the launch was successful and defect free.

Next the three test analysts assigned to the ASP Application Project, converted their tests to the redesigned UI in about 2 days.

### 6.5 Process Accelerated

Because test analysts became able to script automated tests in the absence of the target version, the process flow was modified at Egreetings as shown in figure k.

## A Case Study in Extreme Quality Assurance (XQA)

By James A. Canter and Liz Derr

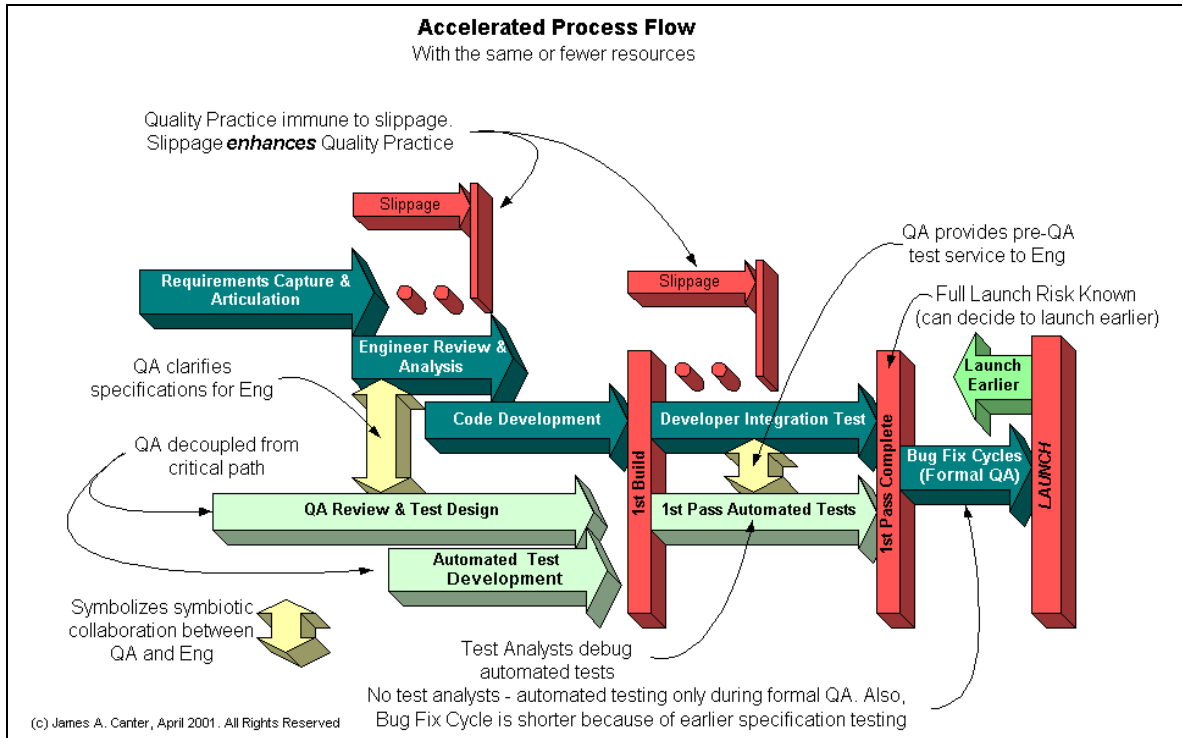


figure k

Finally, the team had achieved the capability of keeping QA out of the critical path. There was no need to bring in contract testers.

Automated tests were consistently delivered 1-3 weeks *before* code was ready for integration testing. Test analysts “manually” executed the automation scripts generated by the script maintenance utility, identifying and correcting defect in their automated tests as well as defects in the new application version.

Engineers gave priority to defects that stopped test execution. Once all automated tests had run, Egreetings knew the launch risk.

Formal QA was defined as the period in which identified defects were resolved and verified. Since it was fully automated, engineers provided as many as two code drops per day. These were subjected to Egreetings full automated test suite, including both new and regression tests.

Business Proponents were able to effectively triage known defects, assigning the highest priority to those representing the largest risk. Based on the availability and confidence in defect data spawned by comprehensive testing, proponents launched every project subjected to RAV early. The ASP Application Project (3 ½ months) launched with *one* known defect. The defect represented functionality that had to be developed in a follow-on project. For three weeks after launch, no additional application defects were detected in production.

## A Case Study in Extreme Quality Assurance (XQA)

By James A. Canter and Liz Derr

Every project subjected to CEG disciplines consistently launched with less than one application defect for each release.

### **6.6 Test Analysts More Efficient**

The first benefit was that test analysts were freed from daily and weekly regression testing, thus able to focus their expertise where it counted most: analyzing and testing requirements while generating high coverage test cases, and implementing the test cases in the utility *before* code was complete.

In this way, fewer test analysts were required because these resources were moved on to the next project *before* formal QA testing began!

### **6.7 Software Developers More Efficient**

The rigorous approach to creating the test cases (via CEG graphs) had several benefits within the engineering organization. It sped up the coding process, reduced the need for QA and Development to debate whether certain behavior was a bug or was as designed, and helped foster management team confidence that verification against requirements would be reliable.

QA test analysts were finding, reporting, and resolving issues that had typically fallen to the engineers to resolve, such as error handling and boundary conditions. Often the business proponents didn't include this information in their requirements. In the traditional process flow that was originally followed at Egreetings, no one noticed these issues until coding is underway.

Requirements, in whatever form, were unclear about what is supposed to happen in each possible scenario. Such discovery had the effect of slowing down the coding process because engineers were forced to resolve these issues with business proponents. Worse yet, if discovery was obscured by an engineer's assumption about the requirement's author's intent, the cost impact when discovered by QA could be significant. QA's test analysts were trained to discern where requirements could be interpreted in more than one way. This was a key attribute to collaboration among test analysts and engineers.

Since test analysts were uncovering these issues well before coding started meant that coding could proceed apace. Also, the engineers became confident that test analysts understood the requirements and that both groups were working on the same page.

With tests ready when the code was ready, engineers received almost immediate feedback on the implementation. Major problems were uncovered very quickly.

Knowledge that the test cases were mathematically proven to be complete with respect to validating the requirements gave both engineering and project managers confidence to trust that once tests passed, the product reflected the requirements.

## A Case Study in Extreme Quality Assurance (XQA)

By James A. Canter and Liz Derr

Prior to introduction of Rapid Application Verification processes, the developers didn't expect much at all from QA, since it was almost impossible to manually test vaguely specified features in the few days afforded QA before launch. In the traditional process model, QA simply didn't catch bugs before they went live. In fact, several emergency releases per week were required to fix undetected defects. The number of undetected defects required the full time commitment of critical developer resources.

When QA began insisting on complete specifications from the Business Proponents, the lives of the developers improved significantly. Developers had much clearer requirements to code from, and their code was thoroughly tested before it went live, so they were freed from day-to-day firefighting to focus on their development tasks.

### 6.8 Regression Testing Costs Reduced

Manual functional regression testing was eliminated. Testing flash animations for new products remained a manual task. Ongoing direct costs to regression testing included a portion of the salaries for the test analyst and test automation technologist who maintained the automated test suite from release-to-release, and the cost of the manual tester who validated flash animations.

The following chart, figure 1, illustrates the significant cost savings achieved.

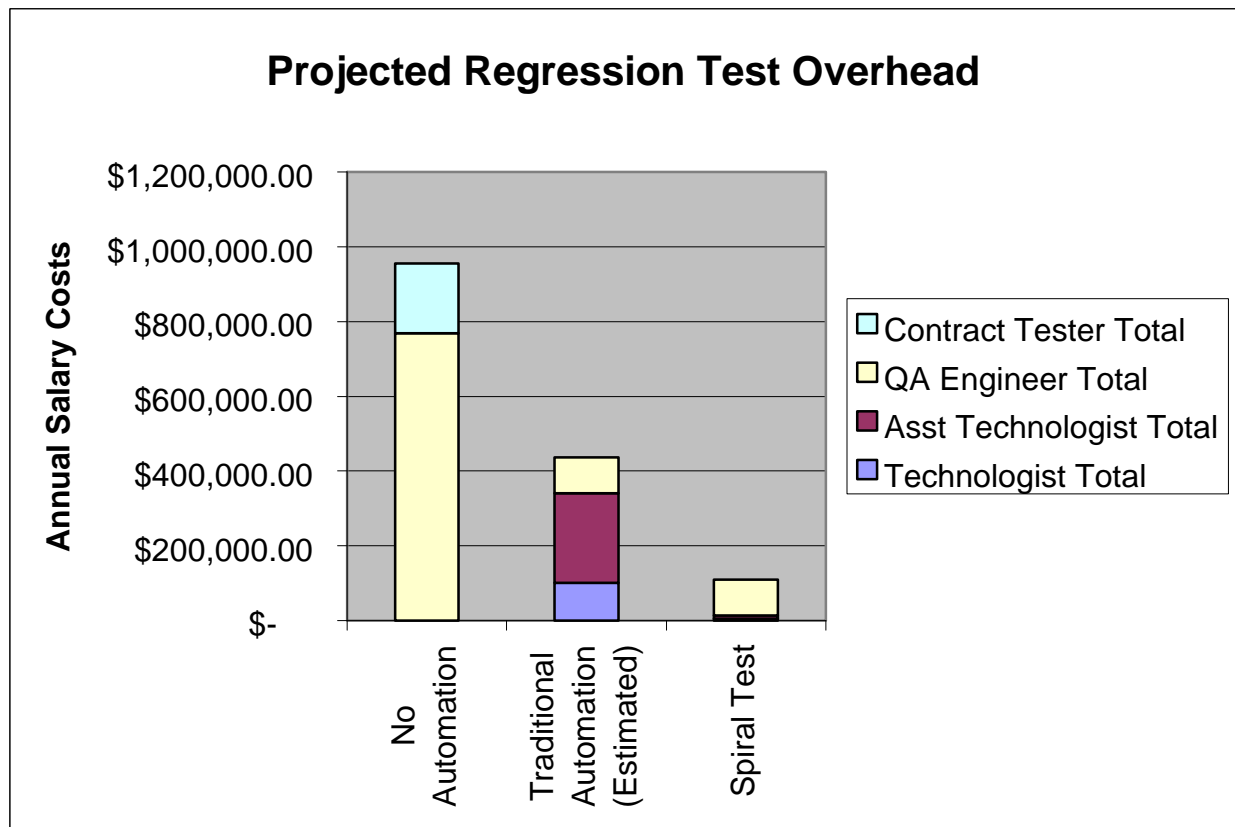


figure 1

## **A Case Study in Extreme Quality Assurance (XQA)**

*By James A. Canter and Liz Derr*

As previously stated, our effort to automate tests against the legacy site using traditional automation methods failed. Thus the cost to Egreetings for Traditional Test Automation is estimated above, and stretches to presume that such an effort could be successful.

Pursuit of manual testing would mean removal of resources from development projects and compromise test execution coverage. Also, manual testing only *seemed* more nimble than traditional automation. This was an illusion because repetitive execution of so many tests was fraught with human error.

And traditional automation solutions could not respond to the rate of change needed to maintain traffic on the website. So, without the script maintenance tool, it was clear that the low defect rate could not be sustained and Egreetings would lose access, thus the utility of their significant investment made in the test suite.

### **6.9 Management Observability in Support CMM Goals**

#### **6.9.1 Early Risk Identification**

Consistently, and as a result of the rigor of CEG practice, significant launch risks were identified before coding began. The program office managed tracking and resolution of these risks.

#### **6.9.2 Systematic Assessment of Requirements Quality**

New as well as experienced requirements authors were afforded the benefit of consistent and disciplined feedback as requirements were analyzed to produce comprehensive tests.

As test analysts discovered missing, inconsistent, or ambiguous requirement phrases, the question they generated was documented in an ambiguity log. With these, the document, location within the requirements document, question raised, and the date discovered were captured. Resolutions, subject matter expert's name, and resolution date were also captured.

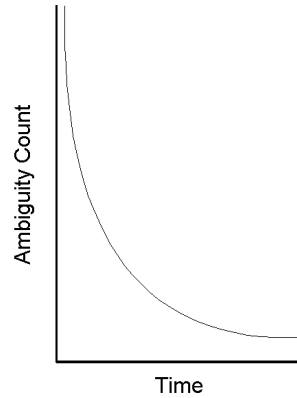
Ongoing status of the total number of ambiguities, and remaining open ambiguities were tracked as a measurement of project status.

At forums hosted by the project office, high-risk ambiguities were identified and discussed. The discussion often resolved the ambiguity or gave rise to a project risk.

With consistent application of this process, test author learning curves became apparent.

## A Case Study in Extreme Quality Assurance (XQA)

By James A. Canter and Liz Derr



*figure m*

### 6.9.3 Systematic Assessment of Process Quality

Test analysts were trained to identify patterns in ambiguities. Without exception a pattern consistently pointed to inconsistent application of documented process or lack in the process itself.

Where an ambiguity pattern suggested such lack, the analyst raised process issues at Program Office project forums.

As with requirements ambiguities these questions were resolved, policy and procedure restated and emphasized, or process redefined to address a discovered deficiency.

### 6.9.4 Project Progress

Graphing functionality by hierarchical segment gave visibility to test creation progress. QA status reports were able to convey progress of test authorship, tests automated, executed successfully once, and tests blocked because of software error(s).

The ability to define comprehensive tests, and then automate them before code was ready established a new, critical project milestone: “1<sup>st</sup> Pass Testing Complete”. This milestone identified to project, business, and engineering managers that the launch risk had been fully defined. It also marked the transition into formal QA testing and the final bug-fix cycle. This was also the point where defect triage and management began in earnest.