intetics
The Remote In-Sourcing® Company

A Single Quality Assurance Specialist Can Transform the Entire Development Process

Summary

Adding a Quality Assurance specialist allows you to better manage your project, but it also presents its own set of problems. What common issues do organizations (and QAs) face when testing their project results? The following case study describes the management process of large projects when a QA specialist is introduced in the middle of the project development life cycle. Emphasis is placed on main problems encountered in such situations and the solutions that were applied in an existing Intetics project.

Introduction

The main objective of the project was to find a BI solution that would maintain a large trading network database. The development team created and maintained the following set of programs and services:

- 30 Web applications, each containing at least two language locales
- 5 self-recording protocols for exchanging data with customers
- Web Service comprising of more than 50 methods

Under these conditions there was only one QA specialist on the team who tested the results of the project.

The main problems at the start of testing

The following were the main problems encountered at the start of testing. In general, there was a complete lack of organized working documentation, as well as lack of functionality testing:

1. Arrangement of working documentation:
    a. Lack of Project Documentation
    b. Lack of Centralized Document Storage
    c. Complete lack of test documentation

2. Gaps in Functionality Testing :
    a. Large volumes of regression testing
    b. Complete lack of testing automation
    c. Unchecked non-functional requirements

Solution to problems

1. Arrangement of working documentation

First, the QA and the project manager decided to reorganize the document flow of the project.

Three most time-consuming issues were identified to fix the document flow:

1. Vague User Stories coming from the customer
2. Documentation bypassing the storage system (Confluence) and settling in personal correspondence and letters or being stored as disparate documents
3. Reluctance to document functionality because of uncertainty of future changes in features

Vague User Stories coming from the customer

The main reason for indistinct requirements was the customer's inability to provide specifications.

Solutions:

1. Provide template of specification to customer.
2. Train the customer on basic principles of creating specifications.

Deciding on a common set of specifications was advantageous because the team was able to implement the functionality covered by the specifications faster. There was no need to waste time writing e-mails and verifying details.

Documentation bypassing the storage system (Confluence) and settling in personal correspondence, letters or stored as disparate documents

The reason for the violation of documentation cohesiveness was the need to send documents to partners in parts.

Solutions:

1. Develop a common documentation template based on needs of the customer and partners
2. The document has to be first created in Confluence and when necessary exported to a text file to be sent to partners
3. Create a house style template for application to exported document
4. Provide partners with limited access to the documentation storage system

Resolving this problem made it possible to assemble all project documents in one place. The customer's development team got the opportunity to track changes in documents and carry on versioning of documents.

Reluctance to document functionality because of uncertainty of future changes in features

Our goal was to possess documentation on all system modules, but the customer disagreed with this proposition, insisting that detailed documentation is not needed in this case.

Solutions:

1. The analysis of each 'one-off' functionality for the possibility of further changes
2. Full functionality coverage with detailed test cases when the probability of change was low
3. Independent document development when the team suggested alterations

2. Functional Testing

Working with test cases

The main problems encountered while working with test cases were:
1. Lack of time to write testing documentation
2. Expensive support

Solutions:
1. Arrangement of working documentation allowed to move past detailed test cases to simple checklists
2. Detailed test cases were written only if there is no specification, saving time on portions where specifications existed

Advantages of this approach:
1. Reducing the time to complete testing documentation
2. Simpler maintenance
3. Enabled to start testing earlier because all documentation was completed faster

Manual Testing

The main problems encountered while conducting manual testing:
1. At the project start all tests were exclusively manual, which considerably slowed down the testing process
2. Absence of exploratory and nonfunctional testing
3. Majority of effort put into regression testing during the iteration

Intetics QA implemented the following solutions to deal with the above problems:
1. Exclusion of significant volumes of testing at the end of iteration
2. Connecting exploratory testing to ‘script’
3. Automation of regression testing by team

Testing automation

When testing was being implemented during the development cycle, all tests were exclusively manual. This considerably slowed down the testing process. The QA specialist was faced with the task of implementing automation testing to reduce the time of regression testing. He chose Selenium and SoapUI, which made the testing process, as well as the testing of Web Services, quicker and more efficient.

Working with Selenium began with the simplified part, Selenium IDE, which allowed quick recording of a test case in a browser and consequently ran this script with a single mouse click from the browser.

Advantages of such automated tests:

1. Quick creation
2. Provides quick feedback
3. Involves developers in the testing process
4. Enables quick and independent verification of quality before continuing the project
5. Such tests are templates that can be further implemented easily using Selenium WebDriver and included in the Continuous Integration cycle

After Selenium IDE automated testing was implemented, the tests were easily transferred to the WebDriver format and included in CI. For this purpose developers and the tester were provided with necessary time on the customer's side.

SoapUI was chosen as the means for Web services testing for the following reasons:

1. Low barriers to entry
2. Ability to quickly create the test suite to run the full test in one click
3. Tests can be included in CI
4. Ability to develop tests using Mocks together with the implementation
5. Ability to create test Load and Security

Results of implementation of project test automation:

1. Autotest Coverage was 75% of application functionality
2. Full transition from Selenium IDE to RC, and then to WebDriver for the year
3. Creating test cases by customer's scripts for all web services
4. Development team was fully involved in the quality assurance process

Resulting Improvements

1. The application is covered with tests cases at all levels
2. Project documentation is always up to date
3. Regression testing time reduced to 4 hours
4. Testing is no longer "dumped" on the end of iteration
5. The whole team is involved in the testing process
6. Number of production customers defects reduced from 5 per week to 1 in 2-3 months
7. Free up time for checking non-functional requirements