26 March 2014

Integration and Regression Testing: Lessons Learned & Best Practices

Team Members:

Jeremy Lerch

Afshin Chaharmahalian

Xiang Cheng

Integration and Regression Overview

The Integration and Regression testing procedures were put in place to ensure that the code being generated through enhancements or defect fixes worked properly and did not cause any issues that were not identified by unit testing. The Scrum process has a focus on completing user stories in one sprint. At the completion of the user story, it is necessary to integrate the code generated into the baseline. Rather than allowing everyone to merge their own code changes into the baseline, there is a method for getting changes approved that involves independent code review of the source code differences and independent black box testing of the enhancement. The user stories are handled on a first in, first integrated basis that is handled through the use of the configuration management forum. This procedure is put in place to ensure that the code is integrated properly and no changes are accidentally overwritten when enhancement branches are merged to the baseline.

The regression testing focuses on black box testing, testing of the software functionality without looking at the internal implementation of the software. This is accomplished partially through the use of the Selenium automated software testing framework and partially through user written test cases for new functionality. The Selenium IDE allows for the recording of test case scripts. These scripts can then be run through the browser either individually or altogether and Selenium will indicate whether the script passed or failed. Selenium test scripts can also be written or edited manually.

Testing Best Practices and Lessons Learned

Static Testing: Static testing is the process of performing code reviews or inspections to catch any defects that may be introduced in new code. There are also commercial tools such as Lint or Klockwork that can be used to perform static testing on the source code. Static testing is useful for identifying defects such as uninitialized variables or incorrect loop handling as well as ensuring that the new source code adheres to the coding standards in use on the program. It cannot be used as a replacement for functional testing but can be used to help locate identify defects earlier in the development cycle. On ClubUML, we have integrated a code review process into the integration checklist to help prevent defects from being introduced into the baseline.

Don’t write test cases guaranteed to pass: Developers can sometimes fall into the habit of writing test cases for their software that will be guaranteed to pass due to their familiarity with the software implementation. While it is important to test how the software works under normal circumstances, it is also important to test how the software works with unusual or unexpected inputs. Testing the software for both valid and invalid inputs is important to ensure that it is able to properly handle invalid inputs. It is also important to write test cases that will test boundary conditions which is a common source of software defects. It is also important for the developer to test all input methods. During the testing of one of the enhancement branches, differing behavior was seen when entering identical text into a text box on a form. The different behavior was exhibited because one developer was entering the text into the text box manually and the other was copying and pasting the same text into the text box. It is not feasible to write test cases that will exercise every single possible input into a system but it is important to write test cases that test the areas where defects are more likely to be introduced.

Continuous Integration: One method currently in use in industry to handle regression testing is the use of continuous integration. Continuous integration is a practice where all code changes are continuously integrated into the baseline. This is done to prevent issues that can be seen when code change integration is delayed by bringing integration problems to light earlier in the process. In continuous integration, the code changes are merged often and the software is automatically rebuilt whenever code changes are merged. At this point, an automated set of unit tests would be kicked off to verify whether the code changes work correctly and did not break any existing functionality. Reports are generated indicating whether unit testing was successful or if it was not, what test failed. This information can be used by the responsible developer to correct any errors that were caused by the integration of their code changes. The developer is responsible for maintaining these automated unit tests as they change code. This would be the ideal method to use in managing integration and regression testing for ClubUML but would require infrastructure that is unavailable for use. To use continuous integration you would need at least one dedicated build server that handles merging in software changes and performing a build of the new software and test server that would be used to run the automated unit tests and generate the results reports.