COMSW4156_001_2013_3: ADVANCED SOFTWARE ENGINEERING (Fall 2013) View Site As: - Select Role -

Home
Announcements
Textbooks
Calendar
Syllabus
Assignments
Site Settings
Roster
Mailtool
Gradebook
Library Reserves
Files & Resources
Piazza
Evaluation
Help

Assignments

Viewing assignment...

Settings for "2013 Team Assignment 6: Static and Dynamic Testing"

Created by	Gail E. Kaiser
Date created	Nov 8, 2013 1:06 pm
Open	Nov 14, 2013 12:00 am
Due	Nov 26, 2013 5:00 pm
Accept Until	Nov 26, 2013 5:00 pm
Modified by instructor	Nov 13, 2013 2:49 pm
Student Submissions	Single Uploaded File only
Number of resubmissions allowed	Unlimited
Accept Resubmission Until	Nov 26, 2013 5:00 pm
Grade	Points (max 20.0)
Alert:	Yes
Honor pledge:	No

Assignment Instructions

Your team testing document should consist of a single file uploaded to courseworks plus the specified material entered into JIRA at http://ase.cs.columbia.edu/jira. The actual implementation code, including all testing code and scripts, should be maintained using git in STASH at http://ase.cs.columbia.edu/stash. Only one member of your team should submit the courseworks file (but all team members can edit jira and stash). You may submit the courseworks part as often as you want up to the deadline (and continuing changing in jira and stash thereafter).

The first page of your document should indicate your team name and list the full names and uni's of every team member.

The second page should give a short synopsis (overview) of the project you actually completed. This can be copied verbatim from a previous document if nothing has changed.

The third and remaining pages should present the process and results (bug reports) of your static and dynamic testing.

Modify your user stories, use cases, CRC cards and UML diagrams to reflect the requirements and design you really implemented. This should have already been done for the previous team assignment, but make further updates if anything has changed since then. Mark anything that has been scoped out of your implementation (but do not delete it). You do not need to include any of these materials in your testing document.

Select two major modules of your implementation for static testing (code review) and select two major modules of your implementation for dynamic testing (test suite execution). These can be the same modules but need not be. A "module" might correspond to a major class or an interrelated set of classes. For static testing, choose those modules that seem most likely to contain "bad" code, e.g., hard to read, poorly commented, badly structured. For dynamic testing, choose those modules that seem most likely to contain errors that would be visible to the end-user (who cannot see your code).

Schedule a code review meeting with your TA; the meeting should be done in the style of a "walkthrough" rather than a full Fagan-style inspection. One pair should be responsible for presenting each module (this does not have to be the same pair who implemented that module). Add a code

inspection meeting log (manually recorded during the meeting by someone other than the presenters) and specific bug reports (constructed from the log after the meeting) to JIRA. Note that a "bug" here might be poor choice of identifiers or lack of comments. Define post-meeting activities to fix the bugs as technical tasks in JIRA.

For each of the two modules selected for dynamic testing, define technical tasks in JIRA to construct and execute a test suite, which consists of a set of unit test cases. Define test cases corresponding to each equivalence partition, both valid and invalid, and present in prose or pseudo-code the defining characteristics of that partition and each of the test cases for that partition - including those at the boundaries. Define a separate technical task for each module to add additional test cases to your suite to try to cover all branches (if your original test suite does not already do so) and document the coverage achieved. Add bug reports to JIRA for each bug found, whether from partitioning/boundaries or from separate coverage tests. Export all these materials to your testing document.

Finally, considering your full system from the customer's perspective, determine the metamorphic properties and associate these with the corresponding user stories. For each metamorphic property, explain how you derived the secondary test cases, then conduct those tests and record the results. If you cannot find many metamorphic properties at the full system level, you may also consider the metamorphic properties of individual functions. Add bug reports to JIRA for each bug found.

You should plan to fix as many bugs as you can before the final report, a separate assignment, but you do not need to have fixed any yet for this testing assignment. For now just record the bug report's