Assignments

Viewing assignment...

Settings for "Code Inspection and Testing Coverage (2014 Pair Assignment 2)"

Created by	Gail E. Kaiser
Date created	Oct 20, 2014 1:44 pm
Open	Oct 23, 2014 12:00 am
Due	Nov 24, 2014 5:00 pm
Accept Until	Nov 24, 2014 5:00 pm
Modified by instructor	Oct 23, 2014 10:14 am
Student Submissions	Single Uploaded File only
Number of resubmissions allowed	Unlimited
Accept Resubmission Until	Nov 24, 2014 5:00 pm
Grade	Points (max 20.0)
Alert:	Yes
Honor pledge:	Yes

Assignment Instructions

For this assignment, each pair should produce the document and submit separately. One of the two members of the pair should submit; you may submit as often as you want up to the deadline. (CVN students can submit this assignment individually.) Note the deadline is on Monday, not Tuesday.

There are two parts: code inspection and coverage testing. The code inspection meeting should still take place with the entire team (and your TA), but each pair is responsible for their own part of the code. The testing part should be done independently by each pair, also on their own code. Continue to share your team spaces in JIRA and STASH, including testing code, but do not share the assignment documents.

The first page of your document should indicate your team name and list the full names and uni's of every team member, but highlight the members of your pair to distinguish them from the others.

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 of your code inspection and coverage testing.

Schedule a code inspection meeting with your entire team and your TA. Each pair should choose one major module that they implemented to present during the meeting. That is, the team should inspect a total of two major modules, one from each pair. A "module" might correspond to a large class or to an interrelated set of smaller classes. Choose modules that seem most likely to contain "smelly" code, e.g., hard to read, badly structured. During the meeting, walk through each module. One member of the pair should present, while the other records the process followed and any bugs found. Note that a "bug" here might be poor choice of identifiers, lack of comments, or failure to follow coding conventions, in addition to a programming logic error.

After the meeting, each pair should add a code inspection log (as actually recorded during the meeting) and corresponding bug reports (constructed from the log after the meeting) to JIRA. You do not need to fix these bugs as part of this assignment, but if you do, then include the resolution in the bug report and add any new test cases to JIRA (with the testing code in STASH).

Starting on the third page labeled "Code Inspection", and continuing for one or more pages, indicate which module you chose for code inspection, describe the process followed for walking through your module, and discuss any problems found and how you plan to resolve them (or have already resolved them). Include your code inspection log and bug reports copied from JIRA.

For coverage testing, again choose a major module that you suspect may have latent bugs (in this case, actual

For coverage testing, again choose a major module that you suspect may have latent bugs (in this case, actual programming flaws). This can be the same module as for code inspection or a different module. The goal is to exercise every statement and every branch in that module; note branches include exiting a loop, raised exceptions, and any other non-straight-line flow of control. Use a coverage tool to determine whether or not your existing test cases (e.g., from previous team assignments) conveniently already fully cover your module, and if so document this. If not, add new test cases (both issues in JIRA and code in STASH) to attempt to reach full coverage. This may not be feasible, e.g., for some rare exceptions affected by the operating system's scheduler or device drivers, so make sure to document which statements and branches you were able to cover and which you weren't. Add bug reports for any new bugs found to JIRA; you do not need to fix these bugs as part of this assignment, but if you do, then include the resolution in the bug report.

Starting on a new page labeled "Testing Coverage", and continuing for one or more pages, indicate which module you chose for coverage, describe the process followed for checking whether or not coverage has been achieved, and discuss any new test cases that were constructed to exercise additional statements or branches. Also described any challenges or roadblocks that arose that prevented 100% coverage. Include any new bug reports copied from JIRA, and how you plan to resolve those bugs (or have already resolved them).

Optionally, for extra credit, start again on a new page labeled "Extra Credit". Devise as many metamorphic properties as you can think of for any portion of your system. This could be the full system from the customer's perspective, specific subsystems, or individual methods, and discuss how you could use each of those properties to devise new test cases. You do not actually have to code and run these test cases, but if you do, then discuss the results and add those test cases (and any corresponding bug reports) to JIRA/STASH. Metamorphic Testing will be presented in class prior to the deadline for this assignment.

Additional resources for assignment

Assignment6Testing.pdf (350 KB; Oct 20, 2014 1:57 pm)

Team6MAssignment6StaticAndDynamicTestingFinal.pdf (635 KB; Oct 20, 2014 1:58 pm)