Black-box Unit Testing (and Demo): Team Assignment 4

Published | Edit

This is a team assignment.

Update your task board to cover everything you have completed, have in progress, or have not yet started during the first iteration. Make sure the tasks indicate who is (or was) responsible for doing them.

If you have not done so already, update your CRC cards and class diagrams to match what you have actually implemented to date, and keep them updated hereafter through the end of the semester. Note your architecture (i.e., set of class diagrams) only needs to cover the first iteration. If your architecture includes elements that you had planned to include in the first iteration but were unable to complete in time, so there is not yet any working code, then mark them as such (e.g., label them all "first iteration overflow"). If you have already included elements corresponding to new features planned for the second iteration, then you should also mark those (e.g., label them all "second iteration YAGNI").

Develop a set of black-box unit test cases for your system, covering all the classes corresponding to the first iteration as implemented (this is why you need to distinguish which parts of the architecture were really implemented in the first iteration). First, describe in prose, using github "wiki" or "issues" facilities, the set of equivalence partitions and boundary conditions that need to be covered for every operation in every class (except you do not need to consider constructors, finalizers, getters, setters, etc.).

Update your task board to include development of the corresponding executable test cases, if not already there. Then indeed expand your equivalence partitions and boundary conditions into executable testing code. Make sure to include appropriate setup/teardown, grouping of test cases according to any dependencies, mocks/stubs as needed, etc.

Set up this testing code to be automatically invoked using a testing tool suitable for your programming language and framework, and check that you can indeed run your entire test suite this way. The testing code, testing scripts, testing tool reports, sample data sets, etc. should all be stored in your github repository.

You do not need to include system-level test cases and grey/white-box test cases now, those will be included in an assignment for the second iteration, but do make sure your full application runs sufficiently to demo.

Schedule a demo with your team mentor before the deadline for this assignment. Plan for 20-30 minutes. The demo itself only needs to be about 10 minutes, but you should allow time for your mentor to try to break your system. Your entire team should attend if possible. (CVN teams should arrange online conferencing to present their demos.)

Report the results of running the entire test suite on the code that you plan to show in the demo. That is, run the test suite first, and fix as many bugs as you can, rinse repeat, before the demo. Tag the repository to clarify which revisions of which files actually contributed to the demo.

Do not submit this assignment until after your demo has taken place. Describe what you were able to demo and record any problems that arose, any recommendations from your mentor, etc. using github "issues" (or "wiki", etc.). Make sure to tell us which specific issues stem from the demo and how to find them. Remember to keep your task board up to date.

Each team should submit a single file, i.e., one member should submit. The name of your file should include your team name, and the contents of the file should also include your team name. Otherwise, all you need to include in this file is references to where to find your various materials mentioned above.

Points	10
Submitting	a file upload
File Types	doc, docx, pdf, txt, xls, and xlsx

Due	For	Available from	Until
Nov 10, 2016	Everyone	Aug 26, 2016 at 12am	Jan 31 at 11:59pm

+ **Rubric**