Western

UNIVERSITY · CANADA

Topic 13

Final Project Acceptance Testing

Computer Science 2212b
Introduction to Software Engineering
Winter 2014

Jeff Shantz
jeff@csd.uwo

Western Science

Department of Computer Science

Project Information and Acceptance Testing

- Integrating your code
- Final code submission
- Acceptance testing
- Other advice / reminders

Integrating Your Code

You *have* been practicing continuous integration, haven't you?

- In the next week and a bit, your team will
  - Finish writing code for its modules
  - Finish writing its tests
  - Ensure all modules compile together
  - Ensure all tests are passing
- We recommend that you set an *integration deadline* within your team well in advance of the final submission deadline
  - Ideally, at least one week in advance, if not sooner
  - That means by next Wednesday

Integrating Your Code

By the integration deadline, if a team member has not submitted his/her code:

- Other team members should feel free to write their own version of the code based on the class descriptions submitted for assignment 2.

By the integration deadline, if a team member has made major changes to your program without communicating them to the rest of the team:

- Other team members should feel free to revert that member's commit(s)
- See http://stackoverflow.com/questions/4372435/how-can-i-rollback-a-github-repository-to-a-specific-commit

Integrating Your Code

In either case, you should be *merciless*:

- Your grade is on the line
- Do not allow a team member to adversely affect your grade because
  - He/she has failed to deliver on time
  - He/she is attempting to make major changes to the software in an attempt to appear as having contributed to the project
    - We can see *when* you've committed in addition to *how much* you've committed.
    - Contributing nothing for the entire semester and then a huge amount at the last minute does not reflect well on you.

Avoid Submitting Bad Code

Any code you push to the repository

- Should compile
- Should not cause any existing tests to fail
- Should be free of obvious bugs

Everyone makes mistakes

- Even good programmers will have bugs
- But there is no excuse for breaking the build. If it doesn't compile or it breaks tests, you don't push it.

If you submit bad code to your repository

- Do not be surprised if it is reflected in your peer evaluations (and thus your grade)

Final Code Submission

All code must be pushed to your team's GitHub repository and tagged with the `asn4` tag by 23:59:59 on April 2.

- There are **no late submissions** allowed for the submission due date (please don't call my bluff).
    - Any code pushed after that time will not be tested.
    - We will check out the last commit on or before 23:59:59 on April 2.
- If it comes down to the wire and you've still got code that is not compiling, **comment it out**
    - We will not take time to edit this or that in acceptance testing in order to get your code to compile.
    - It will receive a grade of zero.

Acceptance Testing

Acceptance testing will take place on April 3 - 4 and 7 – 8

- Each testing session will take about an hour
- Your instructor will be present at your session
- A teaching assistant will also be present, but we cannot guarantee it will necessarily be the one assigned to your team during the term
- Sign up sheet will be posted this weekend
    - First-come, first-served
    - 3 hours of acceptance tests per day (i.e. 3 teams/day)

Acceptance Testing: What To Expect

We will

- Clone your GitHub repository onto a clone of an MC 10 system
- Checkout the last commit on or before 23:59:59 on April 2
- Run `mvn project` in the **ROOT** of your repository
  - This implies your `pom.xml` is in the root
  - If your code does not compile, we're done – *seriously*
- Run your JAR file with `java –jar` (and no other arguments)
  - e.g. `java –jar target/jarfile.jar`
  - If your program crashes, we're done – *seriously*
- Thoroughly test all stated and unstated requirements of the project
  - Many students are often taken aback as we're testing, or sometimes even angered by the tests we execute
  - We will be thorough – test your programs accordingly

Acceptance Testing: What To Expect

We will not

- Pass any additional parameters to your program
- Take time to set up any sort of configuration files
- Allow any changes to be made to the program to get it to compile or stop crashing
- Accept **any** negative attitude / arguing / aggression from students
    - One strike rule
    - If you fail a particular test, accept it. We're not going to argue with you about it. One test is not going to sink your grade.
- Accept an excessive number of interruptions from students
    - One strike rule – we are on a strict schedule
- Allow you to explain how to use a particular feature (unless we ask)
    - You are there to observe; your program should be self-explanatory

Acceptance Testing: What To Expect

You will walk out of your acceptance testing session knowing your *approximate* grade based on the acceptance tests you passed and failed.

- Usability issues encountered during testing will be factored into your grade afterward, which may lower your grade slightly
- Grades for bonus features will be factored into your grade afterward

Acceptance Testing: What I Expect

I am sincerely looking forward to seeing your finished products. With that said, I expect

- Your program to compile
- Your program to run and not crash
- Your program to handle valid/invalid inputs gracefully
- Your program to produce informative, professional error messages
- Your program to not pop up a confirmation dialog upon every action I take within it
- Your team to **show up** at acceptance testing
    - At least one member has to attend
    - **All members** should attend – take pride and ownership
        - I may take this into account when deciding on bonuses/ penalties associated with the peer evaluations

Your Unit Tests

- I will be marking them
- I will evaluate them based on your statement coverage
  - I will also be looking at your branch coverage to ensure you haven't just written tests to get 100% statement coverage
  - Your tests should thoroughly test your code
- GUI classes do not need to be tested
  - However, I will be looking at your GUI classes
  - If I see business logic in them (that is therefore untested), you will be penalized
- Hence, your unit test grade will be based on a combination of factors:
  - Your statement coverage (taking into account your branch coverage)
  - The quality of the tests you've written (have you just written a bunch of redundant tests? Have you achieved 100% coverage without actually thoroughly testing your code?)

Finally…

GOOD LUCK!

We truly want all teams to be successful!

Western

UNIVERSITY · CANADA