Increasing Student Engagement by Having Them Run Your Test Data Through Their Programs

Donald R. Schwartz
School of Computer Science and Mathematics
Marist College
Poughkeepsie, NY, USA

Abstract – This semester I “stumbled” across a new approach to testing student programs. In the past, I’ve frequently wondered how carefully (if at all!) students evaluated the feedback they received when I returned graded programs. Having the students run the test data that would be used to grade their programs gave the students immediate feedback as to issues their programs might have. Since the assignment was very “fresh” in their minds, isolating and correcting the errors should be a rather straight-forward exercise. My hypothesis was that this would help them learn how to develop good test data and help to make them better programmers. This paper describes this approach and summarizes feedback from the students themselves.

Keywords: Assessing Student Programs / Projects, software engineering, programming issues, and laboratory practices / Teaching methods

1 Introduction and Description

This semester I “stumbled” across a new approach to testing student programs. My initial idea was to have the students run my test data for the first programming assignment, mainly so they could see what happens if they don’t submit the entire project folder (and, for example, just submit the .java file). The IDE we use requires all of the supporting files to run the project, so if they’re missing any components, their project won’t load correctly. I decided to include my test data, since it was already on the grading sheet for the program. So, I had the students run the test data for me. And yes, a few of the students discovered that they had not submitted the project correctly, so I allowed them to resubmit (mainly to save myself the time required to reconstruct their project folders in order to grade their assignments). It does make an impression on the students when they download their assignment from our course management system and it fails to load into the IDE! Some students realized that they had not carefully read the programming specifications, so their programs did not conform to the actual requirements for the assignment. (“How many points off if our input file format/program output is different than yours???”) A few more of the students discovered that their programs didn’t run as expected on the test data. Each of these groups of students asked if they could also resubmit. I told them they could, under the syllabus-defined late penalty of 10% per 24-hour period. This led to many “Is (this error) worth 10 points?”-type questions. Since it was the first assignment of the semester, I answered as truthfully (and quickly) as possible. But what I also discovered was that the students weren’t necessarily upset with the test data (and, by extension, with me!) – they were unhappy with how their programs failed when running the test data. The also began to realize the importance of actually reading the program specs carefully. The issue became “what did I do wrong in writing the program?” as opposed to “why did the professor choose test data that messed up my program?” I was intrigued at this realization.

There have been several studies regarding the variety of approaches to testing, assessing and providing feedback on student programming assignments. Some take an overall approach, incorporating all areas of the course [1][2][3]. Others concentrate on testing and development during the programming process [4][5][6]. Many focus on the various approaches to the process of code review [7][8][9][10][11]. A few describe approaches which allow students to resubmit their work after it’s been reviewed [12][13].

In the past, I’ve frequently wondered how carefully (if at all!) students evaluated the feedback they received when I returned graded programs. Did they read and reflect on my comments? Did they actually try to fix their coding errors? Or did they glance at the grade, put it away and then continue work on the current assignment. Having the students run the test data that would be used to grade their programs gave the students immediate feedback as to issues their programs might have. Since the assignment was very “fresh” in their minds, isolating and correcting the errors should be a rather straight-forward exercise. My hypothesis was that this would help them learn how to develop good test data and help to make them better programmers.

What began as a “one-off” exercise quickly became the norm – on the day when programs are due, the first class activity is “running the test data”. Always concerned about the amount of class-time this exercise requires, I constantly made sure students were actually working on the test data and not on trying to correct their programs (or doing anything else!). I also began arriving to class several minutes early so that the students already there could start running their test data before class actually began. (I realized this had become the norm the first time we had a program due on a day when we weren’t in
the lab – “How will we run the test data???” was a common question arising from the students.)

Students are reminded each time that in addition to assessing their programs, I would also be assessing the accuracy with which they recorded the results of running the test data. Inaccurate results (for example, claiming a set of data worked when it didn’t) would result in an automatic letter-grade reduction. I was pleased that, for the most part, students did indeed accurately record their results.

Of course, an overall goal of this activity was to reinforce how important it is for students to be able to generate their own test data. Would the students actually begin creating much more thorough test data as they actively encountered my test data?

2 The Survey Instrument

At the end of the semester, I put together a short survey to solicit feedback on the “experiment”. The following is the text of the survey:

| This semester I tried something new – having y'all run the test data yourselves for each program. I would like your feedback on this approach. |
| 1. What did you like about running the test data yourself? |
| 2. What did you not like about running the test data yourself? |
| 3. What did you learn as a result of running the test data yourself? |
| 4. Should I have next year’s students run the test data themselves? Why or why not? |
| 5. Did you come up with comparable test data to run on your own program BEFORE the due dates? Explain. |
| 6. Did you ever decide to turn in a program late because of the results of running the test data? If so, did it help your understanding of the programming assignment? |
| 7. What test data sets will you develop for Program 11???? |

Figure 1: The Survey Instrument

Students anonymously provided feedback, which is summarized in the next section.

3 Feedback from the Students

1. What did you like about running the test data yourself?

Most students commented that they liked seeing how their programs performed against my test data, but were sometimes frustrated when test data revealed cases they hadn’t considered. Several liked the fact that they could pretty much predict their grades based on how well their programs performed against the test data.

Here are some of their comments:

- Running the test data myself put me in a place where I can reflect on my thought process in solving a specific issue and plan my approach ahead of time for similar problems. It's a plus to see your program crash before your eyes than to have a sheet of paper with comments on it that I might not look at.
- I could somewhat gauge my grade from how the test data ran and could see that immediately.
- I felt that running the test data gave me a better idea on what my code should look like.
- It helped us figure out what we had done wrong with the code and understand why the errors exist.
- Allows me to immediately know what I did wrong and resubmit so I could potentially lose fewer points than I would if we had not run the test data ourselves.
- It's very helpful, especially if you would like to redo your program. You can see a lot of mistakes you may have missed in your final submission. When your program is completely correct, it makes you feel good about learning skills.
- It served as an automatic check on proper submission, for one. If you submitted the wrong thing, you had the opportunity to fix that mistake and have something to hand in immediately.
- I liked the way it made us see if we made a mistake and how it would lead us to find the reason as to why there is an error.
- We were able to test our program and see first-hand if it gave us the correct answers.
- Running the test data myself made me realize the small errors I made in the program before handing it in.
- I like that it gives me immediate feedback on how I did on the program. It’s also helped me develop test
data on my own – I learn well from examples, so seeing examples of test data is useful.

- It was nice to see if you missed a certain case when running your own test data.

- Since I ran test data of my own during development, I was able to very quickly enter the test data in class and see EXACTLY where I was going wrong if anything didn’t run as planned.

- If problems were uncovered through given test data, a student is able to correct those mistakes and turn in the program late, ultimately salvaging valuable points.

- I could see any errors I made so the next project I could fix my code to reflect the correct output.

2. What did you not like about running the test data yourself?

Here, students expressed the desire to get the test data early, so they could correct any errors before the programs were due and avoid any late penalties. (“Ah, this is why it’s critically important that you come up with your own test data!”) A few commented that the exercise made them nervous, because they didn’t want their programs to fail, and that they felt rushed, because I kept stressing that they had limited time to complete the exercise. A couple of students did express concern about the amount of class time required to run the test data.

- I would have preferred some more test data prior that hit all the possible errors so I could work out prior to turning in the project what I had messed up.

- There were chances for the program to not work on things you did not account for.

- I would prefer the test data come with the original prompt considering the 10-point late penalty that begins even after 1 minute of the 1:30pm deadline.

- Sometimes made me anxious about seeing if something went wrong.

- I thought that it caused anxiety showing if my program worked or not.

- I would have to resubmit late knowing I would get points taken off.

- It’s nerve-racking and slightly terrifying.

- Running the test data took a portion of class time I think could have been utilized better.

- Time consuming, lose 10-minutes of class or more.

- Running the test data was time-consuming and depending on how the program was made there were times when it was tedious.

- Stressful and felt rushed at times.

- I did not like how sometimes there would be one line of code that has a mistake in it and it would cause all of the test data results to be wrong. It wasn’t that I was mad at the data itself, but I would be frustrated with myself.

- There was pressure to run everything quickly, since it naturally had to cut into class time. This made reviewing where errors came from more difficult, though of course feedback on the programs when we got them back did that anyway.

- It’s a little discouraging because you’ve turned it in already and you’re finding mistakes sometimes. There’s a chance for resubmission but now 10 points will be taken off after it’s fixed.

- Possibility of user error when running test data if I’m not careful. My mistakes with that are on me since the professor won’t run test data.

- If you made a mistake in entering the test data, you had to restart.

3. What did you learn as a result of running the test data yourself?

Here, as I was hoping, students mentioned the fact that running my test data helped them to develop good test data themselves. Not all of the students agreed…

- Running test data myself displayed a lack in thought I had while writing code. This made me more cautious as I did more code.

- Nothing significant.

- I created more comprehensive test data myself because I saw the importance of testing my program.

- I learned that I have a habit of making simple mistakes.

- Develop a better understanding of the things I did wrong and how to correct them with immediate feedback.
It made me more conscious about my mistakes and it made me triple-check or quadruple-check my program before submission.

At times it would give further insight into scenarios where the code would fail.

I learned that I should run more test data myself to find mistakes before I hand it in.

I learned what the professor was looking for in the program and it was not a surprise to see what I did and didn’t lose points for.

Nothing more that if you handed the program back showing that I had an error…either way I usually know what caused the error.

How to better develop test cases of my own.

I wouldn’t say I learned more besides how well you test our programs.

I learned immediately what was wrong with my program and had an easier time fixing the problems.

The fact that I missed edge cases if I’m rushing when testing at home.

To look at my program in more detail and thoroughly run it with multiple tests before submitting.

Most of us find mistakes and understand the value of running test data as we build our programs.

I learned where my errors were consistently made and areas in which I needed improvement.

4. Should I have next year’s students run the test data themselves? Why or why not?

Most students encouraged me to continue this practice. A couple were concerned about the amount of class time devoted to these exercises.

Yes, it provides students with a format to process future test data of their own. Students better understand their mistakes.

Yes, because it will encourage them to thoroughly run their programs beforehand and identify exactly where they went wrong for next time.

You should give them the test data prior to the due date for them to test if it works.

Yes, will let them see right away what they did wrong.

Yes, because it allows students to better understand the material as well as have a chance at fixing mistakes if it doesn’t work.

Well, it would save you a bunch of time and time is valuable. Honestly, it’s up to you.

Yes. I think it is useful for students to learn how to test their own programs and seeing how a professor does it can help them learn strategies for testing their programs independently.

I think it is important for the students to run the data themselves so they know what to expect grade-wise.

Overall, I think the pros outweigh the cons, so yes.

I think they should check their work every other program, just so their confidence in their work stays at a steady level.

Yes, you do have to trust students to properly run data and record answers. But it’s less work for the professor to do, so allows students to get programs back quicker.

Yes, it challenges the students to see their faults and areas they can improve.

No, it doesn’t do anything extra except give them a rough idea of their grade.

Yes, it provides students with a format to process future test data of their own. Students better understand their mistakes.

Yes, because it will encourage them to thoroughly run their programs beforehand and identify exactly where they went wrong for next time.

Yes, I think it is important for the students to run the data themselves so they know what to expect grade-wise.

No, I think it burned too much time that could’ve been spent learning new things.

5. Did you come up with comparable test data to run on your own program BEFORE the due dates? Explain.

Yes, making my own data after seeing test data provided helped me understand what to expect.

I would run a few, short-winded tests to check methods and overall functionality, but never as extreme as the test data sheets.
• For the first project, no, but after seeing how important the test data was to the outcome of the program, yes.

• Yes. Depending on what the program was asking for, I would enter a series of values and compared what the results should have been to what they actually were.

• Yes, once you develop an understanding of what your professor is looking for, it is easy to replicate test data.

• Yes, as soon as I finished a program, I would make sets of data input based off of examples on the Assignment sheet and sometimes based off of examples we had in class.

• For the most part, yes. Any mistakes in the code hurt our final grade, even if we had the right idea of what do to, and so testing that code in various ways was important.

• I would occasionally if I had more time on my hands to come up with it and run it.

• Somewhat. We were able to test our own programs, but some things would get left out.

• Sometimes, to be honest, I’m lazy and just hand in the program hoping I did everything right. Most other times I have test data.

• Usually I would have comparable test data, however there were instances in which test data I hadn’t thought of didn’t work for my code.

• I attempt to. Usually my test data is more comprehensive and includes more ends/margin testing.

• Typically. I sometimes got lazy and went for “seems to work”.

• No, because I would not know if it is correct or not.

• Yes, I would use test data to check if the program ran successfully with multiple inputs, zero inputs, invalid data, etc.

• Yes, but only after I saw how detailed your test data was. My test data didn’t cover as much of the program as yours.

• Yes and no. While some test data was comparable, other times I either missed or simply skipped over concepts or ideas that change the outcome (i.e. invalid data).

6. Did you ever decide to turn in a program late because of the results of running the test data? If so, did it help your understanding of the programming assignment?

Of course, not all students needed to resubmit.

• Yes, and it did help my understanding of the program.

• Personally, I did not; however, I had plenty of friends who did use the knowledge to their advantage.

• No, but I could see for other people how this could help others see where they messed up.

• I did and it was helpful to my grade but for me it didn’t really help my understanding, just realize my stupidity.

• I’ve turned in a program late, because I knew it was incorrect and I would save points.

• No, because I could guess what most of the test data would be like and any errors would cost me less points than if I resubmitted, but for a student who made a lot of errors, it is a very good option.

• Only one project could have been acceptable to turn in late, but I had no time to continue/finish it, so I just handed it in.

• No, the test data provided rarely revealed catastrophic errors.

7. What test data sets will you develop for Program 11???

Here the responses varied, from rather detailed plans for the test data sets to a more generic “lots of data” approach.

4 Conclusions and Future Plans

There are many different approaches to student engagement and program testing, from fully automated to fully ‘by hand’. These offer various levels of feedback, from binary (success/failure) to very descriptive. My teaching approach is to offer detailed feedback about each error on every program. This is time-consuming, but in my opinion is very important. But what I’ve discovered is that not all students take the time to read (and act on!) all of the feedback.

My experiment this semester was encouraging in that it immediately engaged my students in the practice of testing and correcting programming errors. Students no longer waited for my feedback to determine where their programs might have errors – they experienced those errors when the programs were due. I would much rather have them correct mistakes before I actually grade the programs – let them do the work of finding
and debugging their code, as opposed to simply agreeing with the feedback they received when programs were returned. Student feedback indicates that they somewhat enjoyed the activities and found them quite useful, although it did make some of them “nervous” – not necessarily a bad thing, considering the importance of thorough testing.

I plan to continue this approach in future courses, with some modifications. I need to make a better plan for addressing the class time that is used by this activity, especially when it “drags” past the 10-minute mark. My initial guess is that we might do this activity during the last ten minutes of the class period, as opposed to the beginning. I am also considering modifying the late penalty, to further encourage students to address even the minor errors they discover during testing. I might also make the actual test data and grading sheets available before the due date, although my concern with this is that students might try to “patch” their code to conform to the testing plan. I am also considering incorporating peer testing, which I frequently use in upper-level classes, but have not tried in introductory-level courses.

5 References


