

CMPSC 220 Final Reflection

Matthew Jadud

Revision 20090420

Abstract

At the end of each semester, I remind my students that I am likely to teach a course like they experienced again. For that reason, I ask them what I should *keep, change, add*, and/or *throw away* from the course in its next incarnation. This document is a reflection on the Spring 2009 offering of *CMPSC 220: Programming Languages* at Allegheny College.

1 Keep, Change, Add, and Throw Away

Feedback is a conversation. To this end, no 5-point scale is ever going to provide me with feedback that will help me improve my teaching. Engaging my students—people who are currently making a profession of taking courses—is one of the best ways for me to understand aspects of my instruction were most effective for them.

I employed this kind of reflective conversation with my students during the Fall 2007 semester at Olin College¹ and the Fall 2008 semester at Allegheny in CMPSC 111². What follows is a similar reflection on the feedback provided by the first- through fourth-year students enrolled in *Programming Languages*.

I introduced our discussion by assuring the students, first-and-foremost, that their feedback could not and would not offend. Secondly, I introduced the KCAT framework, suggesting that we might think about evolving my practice in terms of what aspects of their experience I should *keep, change, add*, and/or *throw away* from the course in its next incarnation. Although this limits the conversation in some ways, it provides a loose framework to help the students frame their feedback, and seems to have worked well in the eight years that I have been using it.

¹<http://www.sububi.org/2008/01/17/reflections-on-software-design-f2007/>

²<http://www.sububi.org/2008/12/14/reflections-on-intro-computing-f2008/>

1.1 Keep

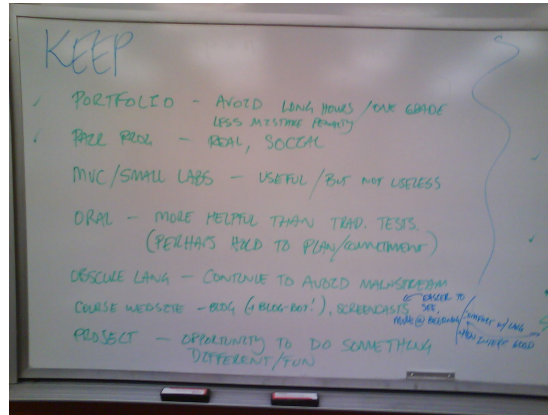


Figure 1. Aspects of CMPSC 220 that students recommend I **keep**.

There are several elements of CMPSC 220 that students were clear I should keep.

1. The course website (ultimately) became a clear and excellent resource.
2. The one-on-one oral quiz was appreciated for the challenge and learning opportunity it represents.
3. Small laboratories are more “digestable” and therefore very much appreciated.
4. The focus on non-mainstream languages (eg. the use of Scheme) provided welcome diversity.
5. Pair programming provided a social environment for students to engage with the course material.

In the next iteration of CMPSC 220, it is clear that more one-on-one sessions would be valuable. In the future, I will work to fix these more firmly in the course schedule. Further, the use of pair programming was handled flexibly—some students stopped programming in pairs (in part due to the complexity of scheduling, no doubt), and in some cases, pairs shifted around. I think maintaining this kind of flexibility is part of what will continue to make it successful.

The portfolio was appreciated because it provides a way for students to avoid code sprints to meet week-to-week deadlines, and the opportunity for revision mitigates the penalty associated with mistakes or confusion. The project was

appreciated because it provided an opportunity to do something different/fun; both of these, however, are more complex, and require additional revision on my part to make them truly successful elements of the course.

1.2 Change

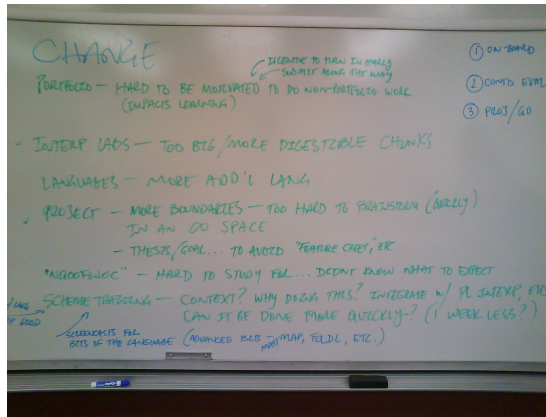


Figure 2. Aspects of CMPSC 220 that students recommend I change.

I am constantly reflecting on the effectiveness of my courses. As a result, I often realize how I would like to revise a course during the semester. It makes me happy when students reflect back things they think should change that are already on my list. The portfolio and project both fall into that category.

1. Inadequate context was provided at the start of the semester for our sprint through the basics of Scheme. Providing a better sense for why we are doing this, and how it ties into the interpretation of languages is necessary.
2. Additional screencasts to support the early stages of learning Scheme would be an excellent addition to the course.
3. The final project felt too open-ended.
4. The portfolio does not cover everything students do in class.
5. Intermediate deadlines will help students structure their time better.
6. The “interpreter laboratories” were too big, and need to be broken into smaller chunks.

In general, oral exams are difficult to prepare for—I do not know how to address this, and think it is part of the joy of an oral quiz or exam. This, I doubt, will change. As to final projects, cheeky though it may be, I will have 10 example projects to share with future students. The posters being produced will serve as starting points for future CMPSC 220 students to brainstorm from and discuss.

I am not yet sure how best to revise the portfolio, but am in general agreement with the problems/suggestions made by the class. Intermediate deadlines (the missing of which carries a penalty, perhaps?) will definitely find their way into the course DNA. Likewise, I will break up (and better motivate) the interpreter labs in future iterations of the course. To attach value to all student work, perhaps *everything* will become part of the portfolio, but I will ask for refinement on a selection of the laboratories/projects carried out over the course of the term.

1.3 Add

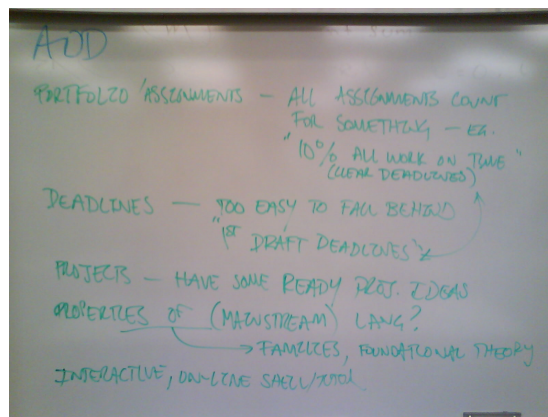


Figure 3. Aspects of CMPSC 220 that students recommend I add.

1. All assignments should count for something.
2. Some kind of "1st draft" deadline would be welcome.
3. Starter ideas for the final project would be welcome.
4. More discussion of mainstream languages would be appreciated.
5. Some sort of tutorial environment for Scheme (a la on-line Ruby tutorials) could be a useful/powerful addition to the course.

Much of what students suggested I add to CMPSC 220 has been discussed under the context of *change*. I do agree that I need to think about how to provide better breadth and/or context for the work of writing interpreters; perhaps through some carefully selected readings and discussion we might explore more of these ideas.

The last suggestion ties into my use of technology to support learning—I made use of video capture of lecture content, as well as full-video screen capture of narrated programming sessions (the latter is typically referred to as a “screen-cast”) this semester to good success. A more interactive tutorial environment might be interesting, but will take time to develop, if such an effort is to be undertaken at all. Perhaps a tool like this (and its evaluation) will become a senior exercise project at some point.

1.4 Throw Away

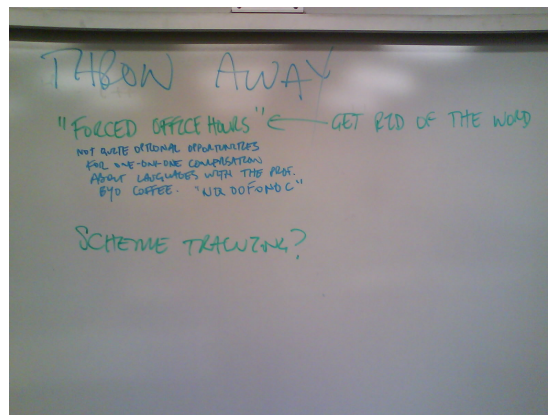


Figure 4. Aspects of CMPSC 220 that students recommend I **throw away**.

1. The term “oral exam” is not good. Find something else.

The best I could come up with (other than “forced office hours,” which wasn’t deemed significantly better than “oral exams”) was “Not Quite Optional Opportunities For One-oN-One Conversation about languages with the prof, byo coffee,” or “NQOOFONOC.” First, this acronym is poorly formed, and second, it is ridiculous. But, the point is well taken.

The second point (getting rid of the bootstrap into Scheme at the start of the semester) is, sadly, without enough context for me to elaborate further. That is, I failed to write enough to remember what point was being made. I can revise

this document to reflect what was being said, but I don't think the majority (or even a substantial minority) found the "training" portion of the course terribly egregious.

2 Conclusion

This was the first time I had the opportunity to teach an undergraduate course in programming languages. From the mood of students, the quality of their effort, and the apparent learning (from conversation with students throughout the term), I would call this instance of *CMPSC 220: Programming Languages* a success. Although the course is not without faults, they are all correctable or points where I can easily improve the course offering.

My single largest question is whether or not the first- through fourth-year students enrolled in *Programming Languages* this semester would recommend that I continue to teach this course to future generations of students of computing at Allegheny. My overall impression is that, given the kinds of modifications and considerations discussed here, the current students would say "yes."