UTEP CS CQI Process

Languages Area Report, Fall 2017

April 6, 2018

On March 29, 2018, the committee members, Nigel Ward, Vladik Kreinovich, and Yoonsik Cheon, met to discuss the two courses in this area: CS 3350 (Automata) and CS 3360 (Programming Languages). Discussion was based on the CS3350 Report, dated March 25, and the CS 3360 Report of March 21.

Comments on the CS 3350 report

The report shows that all learning outcomes were met. (Since statistics were computed over all students taking each test, including some who later dropped, the values in the report probably slightly understate the true performance.)

Comments on the learning outcomes

We discussed how the knowledge taught in this course is useful in other courses. There are many outcomes, not all of which are probably equally important. From a practical view, the outcomes related to Regular Expressions, State Machines, and Context-Free Grammars are probably the most critical. From this perspective, outcomes 2a, 2b, 2d, 2e, 2g, and 2i may deserve more attention than some of the others.

Other comments on issues seen in 3350

Recently enrollments are large: 83 in Fall 2017. With larger enrollments, attendance seems to drop and also other forms of engagement, for example, many students are completely unconnected, don't reply to emails, and don't respond to handwritten invitations to talk with the instructor. This is a concern, although not one directly for this committee.

Comments on the CS 3360 report

The report shows that all learning outcomes were met. Performance on the programming assignments, although an essential part of this class, was hard to relate directly to learning outcomes, so these were discussed in a non-quantitative way. Most of the recommendations made in the previous outcomes analysis (Fall 2015) were adopted, and these seem to be effective.

Comments on the learning outcomes

The decision on the last cycle to drop the level of the "attribute grammars" outcome is felt to have been appropriate; it reduces student frustration, with no practical ill effects.

Other comments on issues seen in 3360

Large enrollments (50+) are a challenge here too, reducing, for example, the time available for in-class demos, and leading (among other considerations) to the decision to make all assignments be pair programming. The drop rate is around 10%, which is a cause for concern. It seems that many of the students who drop, or who were dropped, lack adequate programming skills. The programming assignments were reviewed, and felt to be at an appropriate level, given that the aims of this course

include going beyond getting just a taste of different languages, to really understand the various programming paradigms and becoming able to use different languages to build real programs.

Reflecting some of these concerns, Nigel and Yoonsik earlier this month sent a note to the faculty giving suggestions regarding who to advise into the course, with the goal of reducing these problems.

Initial Discussion (March 29th Meeting)

It was felt that many students, in both classes, demonstrate an inability to write non-trivial code.

Initial Set of Possible Recommendations

Make CS 3331 a prerequisite to CS 3360.

Ask the Fundamentals Area instructors to tighten their standards, to only give a C or better to students who demonstrate the programming skills adequate to succeed in these and other upper-division CS classes.

Ask the CS 3 instructors to include more integrative assignments, so that students are not intimidated by large or loosely-defined programming assignments.

Investigate whether the students in trouble are disproportionately from EPCC.

Follow Ups

To enable more informed recommendations:

1. Nigel will survey the current CS 3360 students to see whether they think that they have the skills needed to succeed in this course, and if not how they think the curriculum might be adjusted. The results are in Appendix 3.
2. Nigel will look up the academic record of students in CS 3360 this semester who dropped or who are currently failing, to examine which related courses they have taken (3350, 3331, 3432), where they took CS 1, 2, and 3, their grades in those classes. The results are in Appendix 2; and a further follow-up in Appendix 4.

Follow-up Discussion (April 6th Meeting)

Discussing the findings below (Appendices 2-6), we were all surprised that Arch 1 is the single strongest predictor of success in 3360. This could be for many reasons, including 1) that students there learn C, 2) that they there learn key concepts including memory allocation, the call stack, and subroutine calling mechanisms, 3) that the criteria for success there are stricter and thus better at "weeding out" students

unlikely to succeed in the upper division, 4) that this course provides the most relevant experiences in solving complex integrated problems, providing a level of problem-solving maturity that is valuable.

However it is not clear whether the high drop rate in 3360 is a problem only in this class. Anecdotally this is true of many other classes, even at the upper division.

We recommend

1. The drop rates in other courses should be examined, to determine whether this problem is unique to this class. If it is more general, more general solutions should be considered.
2. Advisors should strongly encourage students to take Architecture early, and particular, before or concurrently with 3360, especially for weaker students, as determined by considering the other risk factors identified in Appendices 5 and 6.
3. The department should consider renumbering 3360 to be at the 4000 level, to send a signal to students that it should be taken after most 3000 level courses, due to the content dependencies (even though they are soft dependencies) and maturity-level expected.
4. The prerequisite for 3360 should be modified to "Architecture or a B-or-better in CS 2302".

Appendices

Appendix 1: Email to the faculty, March 9, 2018

> Dear Colleagues,
>
> As you advise students into classes, please try to encourage them to take 3360, Programming Languages, after 3432, after 3350, and after 3331. If they know CFGs and Assembler, then large parts of this course will be for them review and enhancement (one of the great strengths of this class is in helping them connect up the knowledge they've learned previously), otherwise they'll be learning those things for the first time, and many struggle. We (Yoonsik and I) don't think these need to be hard pre-requisites, since a strong student can learn them adequately from the textbook and lectures, but for most students having taken those classes first will be a great help. Regarding 3331, the connection there less direct, but in PL we need students to be good at debugging and at getting things to work without much hand-holding (although of course we can do some), and for this purpose, having experience with fairly-large software frameworks, and environments is very helpful. This is perhaps more fundamental; while it can be rough for students not to know, say CFG, the ones who drop or fail are probably more commonly those who can't write code. Half of this class is about learning new programming languages, and we see great struggling among those who lack (general) programming skills; every semester there are a few who it seems simply can't write code, quite apart from the challenges related to the new programming languages they learn.
>
> Of course, we know that better advising won't solve all problems: looking at the transcripts of at-risk students, we note some who are in PL after taking all three of

these classes still have great trouble; spot-checking, many of these are straight-C students. And of course there are always a few special cases relating to family health issues etc.

Anyway thanks in advance for your role in guiding into this class only students likely to succeed in it!

Yoonsik and Nigel

Appendix 2: Notes on students who dropped or who are at risk of failing 3360 in Spring 2018

Of the 62 students who started, 8 dropped by the drop deadline. In addition, after the 2nd midterm, 3 are at risk of failing. To seek to understand the factors involved, I below record some observations, based on information in Goldmine, and more specific information known to me or the TAs.

The 8 students who dropped

overall GPA 2.5
Took CS1-2-3 at EPCC, earned As in all
C in Automata, B in Objects
W in PL last Fall
was doing fine

overall GPA 2.24
took CS1-2-3 at EPCC, earned A, C, C
started EPCC in Fall 2010
C in Objects, C in HCI
W in PL last Fall
came in hoping that with a different professor he'd do better

overall GPA 1.86
CS 1-2-3 at UTEP: B, C, F/C
Objects F, D,
Automata F, F
came in worried that she would have trouble

Overall GPA 3.34
CS 1-2-3 at UTEP: A, A, C
C in Objects, B in Automata, W in Arch

overall GPA 2.63
took CS 1 at EPCC
W, W, C in CS 2
W, F, B in CS 3
C in Automata, C in Objects

overall GPA 2.77
CS 1-2-3 at EPCC: A, B, B

B in Objects, C in Automata

Overall GPA 3.33
CS 1-2-3 at EPCC: A, A, A
B in Objects, B in Automata
no other CS courses
appears to have dropped all courses for this semester

overall GPA 3.42
CS 1-2-3 at UTEP: C, B, C
A in Objects
no other CS courses yet
told a TA that he was dropping since his other classes were too demanding

The 3 others with an F so far in the course

2.82 GPA
CS 1-2-3 at UTEP: B, B, C.
B in Automata, C in Objects
no other CS courses

3.06 GPA
CS 1-2-3 at UTEP: BC, B, C
Automata D, Arch not yet, Objects not yet

2.77 GPA
got sick and was hospitalized
CS 1-2-3 at UTEP: F/D/A, W/A, D/B
B in JPO; no other CS classes

Appendix 3: In-class survey comments by students who self-reported as "not thriving" in this class (Spring 2018)

I explained to the class the CQI process, and told them the faculty were concerned with the failure rates in this course, and considering possible changes to the curriculum. On the board I wrote three questions:

- Are you thriving in this course (earning the grade you want)? Y/N
- Should any of the following be prerequisites to this course: Arch 1, Automata, Adv. OOP?
- Should students be tested on programming skills before entering the course?
- Any suggestions?

The third question was motived by the observation that "some students just cannot write code."

Verbatim responses from those who unambiguous self-reported as "not-thriving" were as follows (excluding non-relevant comments about the teaching in 3360 or other courses).

- They should be required or not allowed to be taken concurrently with Arch 1 and automata,
- I believe that Arch 1 be required.
- I'm not thriving in PL but it's for other reasons other than skill (had a full time job). In fact, one of the reasons I'm surviving is because of automata, A.O.O and Arch 1. So in my opinion they should be pre-requisites to PL.
- Arch 1, should be a *required*, not recommended prerequisite class. Advanced OO shouldn't be a prerequisi5te. Automata should be taken alongside PL. Programming skills should be tested if by now Java still proves difficult we should not try to learn other languages.
- I've had Automata and Advanced Object and I did really well in those classes, however I don't know why I'm doing bad in PL. I think it's just that I've forgotten the material in those classes.
- Even though I have taken and passed Arch 1, Adv O-O and Automat, the way I learn is different from the way the class was being taught to me.
- No, but because I have been missing some readings. I'm actually taking Arch 1 and Automata at the same time with this class, and it has worked for me because I see the same concepts, but in different context, so I get the concept better. Yes, we should test the skills, and required that while doing CS1, CS2, and CS3, they teach more than one PL.
- Have taken Adv O-O, and Automat not Arch.
- No. I did better on the second test. No, the class itself does not rely on programming skills 100%
- No, because little stuff I keep forgetting on tests. Should Arch 1, Adv O-O, Automata required? I would say no because stuff from this class can help you on other classes. Should test prog. Skills? Not really because the course is I feel more theory than actual coding.
- No, taking PL together with Arch 1, O-O, and Automata. I find myself having to learn or figure out things that are taught on the previously –mention class, bu that are only briefly mentioned in PL.
- No - I have taken those courses and they don't help. No – class has little to do w/ programming skills.
- Should Arch 1, Adv O-O, Automata be required? I think they can be the difference why I'm not failing, but they but is not helping me a lot. Should test prog. Skills? No I think we don't need to know a lot.
- No, I am not getting what I expetec, but not because of lack of preparation, it's because of time. I think that assignments require more time than I have.
- No, howver I also did not "thrive" in Arch 1, Adv O-O, Automat. Yes you should test prog. Skills.
- No, but not because of the lack of Arch 1, AOOP or Automata. Yes (test programming skills).
- No. Yes, AOOP, Automat and Arch would have prepare me a lot better for this course. A lot of the material was introduced to me for the first time in this class and I think it affected my performance compared to those that have already taken those courses. While they are important, I do not think this course requires extreme knowledge/skills in programming, because it is more theoretical and it gets down to the details of how the languages work.

Summary: there is a lot of diversity of opinion, with some support for each of the interventions suggested.

Looking at the comments by students who said they were thriving, all of the proposed interventions had some support. Requiring Arch 1 was mentioned somewhat more than the others, and many mentioned that having taken it had been helpful.

Appendix 4: Statistics on dropped/at-risk students versus a sampling of other students (Spring 2018

Comparing the 11 students discussed in Appendix 2 above to a random sample of 11 others in the class, and to responses on initial survey of all students.

	dropped or at-risk	other (of 11)	all*
CS 1,2, 3 at: UTEP – mixed – EPCC	6-1-4	9-1-1	
Grade in CS 3: A-B-C	2-2-7	5-1-5	
Previous W/F in PL: no- yes	9-2	11-0	
Automata: taken-taking-not yet^	7-0-4	10-1-0	46-6-10
Arch: taken-taking-not yet	0-0-11	8-3-0	28-4-30
Adv. OOP: taken-taking-not yet+	7-0-4	11-0-0	45-4-13

^ including with the best grade so far of F, + including with the best grade so far of D or F

*based on the Spring 2018 semester-start survey. There were 61 respondents of 62 enrolled. I compute the results above as if the missing respondent had taken none of the three classes being investigated.

Appendix 5: Statistics on likelihood of "failing" (dropping or at-risk-of-F), Spring 2018

	With*	Without
Automata	**13%** (7/52)	**40%** (4/10)
Arch	**0%** (0/32)	**36%** (11/30)
Adv. OOP	**14%** (7/49)	**31%** (4/13)

* includes taking concurrently

Appendix 6: Simulated Effects of Two Possible Prerequisite Changes, based on Spring 2018 Statistics

Possibility 1: new pre-requisite: Architecture

Benefit: reduce failure rate from 18% to 0%

Cost: delay 34 students in their degree progress

Possibility 2: new pre-requisite: Architecture (may be taken simultaneously)

Benefit: same

Cost: delay 30 students

Possibility 3: new pre-requisite: Architecture or a B-or-better in CS 2302

Benefit: reduce failure rate from 18% to 6%

Cost: delay perhaps 15 students by a semester

Note: These are based on simulations, assuming that nothing else changes. In practice, the cost will be less, as students change their course-picking strategy and to take Arch earlier. The benefit may also be less, if some of the effects of Arch are due to the way it selects and sorts students, rather than purely in educating them.