Languages Area CQI Report, Fall 2019

On January 14, 2020 we met to review the CQI reports for CS 3350 Automata (two instructors, so two reports), CS 3360 Programming Languages (ditto) and CS 4342 Databases.

Attending were Cheon, Mejia, Villanueva, and Ward. Dr. Kreinovich later concurred by email.

First we discussed the assessment instruments and criteria. Some of the assessments were suboptimal, and we identified ways to do them better next time, but concluded that we could still trust the results. There was discussion as to the appropriate criterion for judging whether each outcome was satisfied. While the traditional criterion is "an average of 70% performance or higher" on the relevant assessment instruments, we adjusted that in two cases. First for CS 4342 a D, not a C, is a passing grade (also true in CS 3360), so 60% was used in that class. Second, for one section of CS 3360 the grading scale is always very harsh, with 80% being the A cut-off, so for that section also a 60% criterion was used. However these were interpreted flexibly; any outcome where the score was at or near substandard, by any criterion, was discussed by the committee.

Second, we reviewed the reports in detail. Some typos and clarity issues were noted, and revisions requested of the authors.

Third, we reviewed all outcomes. In sum:

- For CS 3350, all outcomes were met for both sections.
- For CS 3360, most outcomes were met.
- For CS 4342, all outcomes were met.

Fourth, we discussed needed mitigations and improvements, and voted on recommendations. The resulting recommendations are as follows:

CS 3350

- We recommend that outcomes 1b, 1c, 1d, and 1e be reworded to say "describe", since "understand" is not measurable
- We recommend that outcome 2a be split into its three component outcomes
- We recommend that attention be paid to giving students a working knowledge of these models. While we were unable to formulate this desire as a specific necessary outcome, we encourage instructors to have more examples of applications rather than teaching students to just memorize and simulate standard conversion algorithms. We note that FSA and CFGs are important in Software Engineering, Networks, and Programming Languages, and think that it should be easy to find interesting examples that help the students learn to work with these models in ways that give them a deeper understanding that they can use when opportunities to do so arise later in the curriculum or in the workplace.

CS 3360

- We recommend that outcome 2c (met marginally in one section, assessed at the wrong level in the other) be moved to a Level 1 outcome: "Describe ways to formally specify the dynamic semantics of small subsets of programming languages, such as expressions and control structures." Students do not have the mathematical background to attain any level of mastery of axiomatic and denotational semantics, nor do they generally need such mastery. This is any case truly a graduate-level topic.

- We recommend improving teaching relative to 1b (stages), 2a and 2b (BNF), 2d (specifically dynamic/static scope), and 2e (specifically subprograms as parameters), all problematic or marginal in one section or another, by following the specific recommendation in the CS 3360 reports.
- We recommend a new level-2 outcome: "Be able to write programs to solve simple problems in a purely functional language." This is because an important aim and major topic of the course is helping students become stronger programmers, including improving the ability to look at problems in different ways.
- We recommend another level-2 outcome: "Be able to write programs to solve simple problems in a scripting language," for the same reasons.

CS 4342

- We recommend that teaching relating to outcome 1b strengthened by following the recommendations in that report.

CS3350 Automata, Computability, and Formal Languages
Course Report
Fall 2019
Vladik Kreinovich

The class started with 50 students. One student dropped the class.

Of the 49 students that took the final exam, the grades are as follows: 23 As, 19 Bs, 6 Cs, and 1 D.

As a criterion, we used the average grades on students on the corresponding questions on midterm exams and on the final exam. All the exams and homeworks are available on class website http://www.cs.utep.edu/vladik/cs3350.19a

Learning Outcomes

Level 3 Outcomes: Synthesis and Evaluation

Level 3 outcomes are those in which the student can apply the material in new situations. This is the highest level of mastery. Upon successful completion of this course, students will be able to:

3a. Compare regular, context-free, recursive, and recursively enumerable languages.

* Tested by Problem 10 from Test 1, average grade 68/100; by Problem 2 from Test 2, average grade 75/100; by Problem 1-2 from Test 3, average grade 82/100; by Problem 2, 3, and 4 from the Final Exam, average grades 80/100, 82/100, and 85/100.

3b. Compare FA, PDA, and Turing machines.

* Tested by Problem 10 from Test 1, average grade 68/100; by Problem 2 from Test 2, average grade 75/100; by Problems 1-2 and 5 from Test 3, average grades 82/100 and 95/100; by Problems 2 and 3 from the Final Exam, average grades 80/100 and 82/100.

Level 2 Outcomes: Application and Analysis

Level 2 outcomes are those in which the student can apply the material in familiar situations, e.g., can work a problem of familiar structure with minor changes in the details. Upon successful completion of this course, students will be able to:
2a. Convert a non-deterministic FA (respectively transition graph) into an equivalent deterministic FA, convert a transition graph or NFA into an equivalent regular expression, and convert a regular expression into an equivalent FA.

* Tested by Problem 7 and 8-9, Test 1, average grades 80/100 and 69/100; by Problem 1 from Final Exam, average grade 74/100.

2b. Construct a regular expression (respectively a context-free grammar) for a regular language (respectively context-free language).

* Tested by Problem 8-9 from Test 1, average grade 69/100; by Problem 4 from Test 2, average grade 71/100; by Problems 1 and 2 from Final Exam, average grades 74/100 and 80/100.

2c. Convert a context-free grammar into an equivalent pushdown automaton.

* Tested by Problem 1 from Test 2, average grade 68/100; by Problem 2 from the Final Exam, average grade 80/100.

2d. Construct a context-free grammar for a given context-free language.

* Tested by Problems 2 and 4 from Test 2, average grades 75/100 and 71/100; by Problem 2 from the final exam, average grade 80/100.

2e. Design an algorithm for a machine model to simulate another model.

* Tested by Problem 2 from Test 2, average grade 75/100; by Problems 3 and 5 from Test 3, average grades 75/100 and 87/100; by Problems 2 and 3 from Final Exam, average grades 80/100 and 82/100.

2f. Build simple Turing machines.

* Tested by Problems 4 and 5 from Test 3, average grades 75/100 and 95/100; by Problem 3 from Final Exam, average grade 82/100.

2g. Prove formally properties of languages or computational models.

* Tested by Problem 10 from Test 1, average grade 68/100; by Problems 1-2, 8, and 9 from Test 3, average grades 82/100, 89/100, and 88/100; by Problems 2, 3, and 4 from Final Exam, average grades 80/100, 82/100, and 85/100.

2h. Apply a parsing algorithm.

* Tested by Problem 5 from Test 2, average grade 90/100; by Problem 2 from Final Exam, average grade 80/1000.

2i. Build a parse tree or a derivation from a context-free grammar.

* Tested by Problems 2, 4, and 6 from Test 2, average grades 75/100, 71/100, and 72/100; by Problem 2 from the Final Exam, average grade 80/100.

2j. Use the closure properties in arguments about languages.

* Tested by Problem 4 from Test 1, average grade 94/100.

Level 1 Outcomes: Knowledge and Comprehension
Level 1 outcomes are those in which the student has been exposed to the terms and concepts at a basic level and can supply basic definitions. The material has been presented only at a superficial level. Upon successful completion of this course, students will:

1a. Be familiar with the implications of Church-Turing thesis.

* Tested by Problem 10 from Test 3, average grade 90/100; by Problem 4 from Final Exam, average grade 85/100

1b. Understand that there are problems for which an algorithm exists, and problems for which there are no algorithms (non-recursive, non-recursively enumerable languages) and understand the implications of such results.

* Tested by Problem 9 on Test 3, average grade 88/100; by Problem 4 from Final Exam, average grade 85/100.

1c. Understand and explain the diagonalization process as used in proofs about computability.

* Tested by Problem 9 on Test 3, average grade 88/100; by Problem 4 from Final Exam, average grade 85/100.

1d. Understand the difference between feasible and non-feasible algorithms, understand the limitations of the current formalization of feasibility as polynomial-time

* Tested by Problem 6 from Test 3, average grade 87/100; by Problem 4 from Final Exam, average grade 85/100.

1e. Understand the main ideas behind the concepts of NP and NP-hardness, know examples of NP-hard problems

* Tested by Problem 7 from Test 3, average grade 76/100; by Problem 4 from Final Exam, average grade 85/100.

ANALYSIS OF THE RESULTS

Overall, the results are satisfactory. In all topics, we got at least C level average, and in most topics, B level.

On average, the results are somewhat better than in Fall 2017 -- which is to be expected, since, in contrast to Fall 2017, the class was divided into two smaller sections, and smaller sections usually lead to better learning. However, the improvement was miniscule, much smaller than I expected.

This is not due to the complexity of the topic, since on the final exam, the first megaquestion about finite automata -- the easiest part of the test -- got 74/100, while all other questions that cover much more complex topics got B average. A possible explanation comes from student evaluations: several students mentioned that they did not appreciate the need for studying this topic until well into the semester. This makes sense: to get to real applications, it is necessary to go through a lot of technical details. It is desirable to have more examples of applications explained in the beginning of the class.

On the negative side, my feeling is that the students' attitude somewhat worsened. In the past semester, after a not very successful test, I would again review the difficult part of the material and give students quizzes to make sure that they got it this time -- and it worked well, according to the quizzes, their understanding improved. This semester, when I tried that, the average grade on the quiz was almost the same as the grade on the same question in the original exam. I attribute this to the same phenomenon as mentioned above: students not understanding the need for this material. Hopefully, more emphasis on understanding the need for this course will help.

Also, in light of the need to put more emphasis in our curriculum on parallelism -- mentioned by the accreditors -- as an experiment, I taught the students some elements of parallelism theory. The

corresponding metaquestion was added, as a extra credit question, to the final exam. Half of the students tried to answer it. One student got it all right, most other students got some extra credit for answering part of this metaquestion. Taking into account that we spent only a little less than two class periods on this topic, this shows that students are quite capable of learning it. On the other hand, once we got a more detailed list of what exactly accreditors want, we realized that what they want is not theory -- so maybe we should just add a little bit of parallelism to this class without spending too much time on it. The corresponding material is already in the textbook that students use, so this addition should be relatively easy. Probably we should add this to the list of outcomes. If this can help in our response to accreditors, then definitely we should do it. In general, with the addition of the new parallelism required course this addition to Automata outcomes will be well understood by the students.

P.S. I am still thinking about other suggestions on modifying the wording of the outcomes made by Daniel Mejia, who taught another version of this class. They all sound reasonable, but I continue thinking since I feel that some caution is needed, we do not want to solidify the material too much by making outcomes too detailed.

Course Number: CS 3350
Course Title: Automata, Computability, and Formal Languages
Course Instructor: Daniel Mejia

Grade Distribution
The class enrollment was 35 students at census day and throughout the entire semester. No students dropped the course.

Grade	Pass				Fail
	A	B	C	D*	F
No. of students	11/35	15/35	6/35	1/35	2/35
Percentage	31.4%	42.8%	17.1%	2.8%	5.7%
Total	33/35 (94.3%)				2/35 (5.7%)

*D is a passing grade in CS 3350.

Out of 35 students who were enrolled in the class 33 students (94.3%) passed, with a D or greater. One student (2.8%) passed the class by receiving a D. 2 students (5.7%) failed, including one who did not take the final exam. Of the failures, one student was unable to show that the material was understood; and, one student last attended the course on October 2, 2019 – due to certain restrictions, this student could not be dropped from the course.

Learning Outcomes
Assessment was done using exams and in-class quizzes only. Homework was assigned as part of the class but was intended for students to use as practice problems and not assessment.

For each outcome I measured the average score obtained by students on the exam or quiz questions. The nominal acceptance value was 70%; any outcome that was assessed and received less than this was omitted from this report. Most outcomes had multiple questions throughout the semester mapped to them, and for these outcomes I picked a few most relevant, representative, or recent questions.

Level 1: Knowledge and Comprehension
Level 1 outcomes are those in which the student has been exposed to the terms and concepts at a basic level and can supply basic definitions. The material has been presented only at a superficial level.
Upon successful completion of this course, students will:

a. Be familiar with the implications of Church-Turing thesis.
   Quiz 5, Q9 – 75%
   Final, Q21 – 93%
   Good

b. Understand that there are problems for which an algorithm exists, and problems for which there are no algorithms (non-recursive, non-recursively enumerable languages) and understand the implications of such results.
   Final, Q15 – 93%

Excellent

c. Understand and explain the diagonalization process as used in proofs about computability.
Final, Q14– 88%
Good

d. Understand the difference between feasible and non-feasible algorithms, understand the limitations of the current formalization of feasibility as polynomial-time.
Quiz 5, Q8-9 – 75%
Final, Q15 – 93%
Good

e. Understand the main ideas behind the concepts of NP and NP-hardness, know examples of NP-hard problems.
Quiz 6 Q1-5– 74%
Final, Q15 – 93%
Good

Level 2: Application and Analysis
Level 2 outcomes are those in which the student can apply the material in familiar situations, e.g., can work a problem of familiar structure with minor changes in the details.
Upon successful completion of this course, students will be able to:

a. Convert a non-deterministic FA (respectively transition graph) into an equivalent deterministic FA, convert a transition graph or NFA into an equivalent regular expression, and convert a regular expression into an equivalent FA.
Quiz 3, Q1 – 80% (NFA -> DFA)
Exam 1, Q8 – 81% (NFA -> Reg Exp)
Exam 3, Q5 – 77% (NFA -> DFA)
Final, Q4 – 91% (NFA -> Reg Exp)
Final, Q2 – 77% (Reg Exp -> FA)
Satisfactory

b. Construct a regular expression (respectively a context-free grammar) for a regular language (respectively context-free language).
Exam 1, Q8 – 81%
Exam 3, Q9 – 89%
Final, Q4 – 91%
Good

c. Convert a context-free grammar into an equivalent pushdown automaton.
Exam 2, Q1 – 90%
Final, Q8 – 86%
Good

d. Construct a context-free grammar for a given context-free language.

Final, Q6 – 97%
Excellent

e. Design an algorithm for a machine model to simulate another model.
Final, Q12 – 76%
Satisfactory

f. Build simple Turing machines.
Final, Q12 – 76%
Satisfactory

g. Prove formally properties of languages or computational models.
Exam 2, Q7 – 70%
Exam 3, Q2 – 80%
Final, Q5 – 85%
Good

h. Apply a parsing algorithm.
Exam 2, Q6 – 87%
Final, Q2 – 77%
Final, Q9 – 98%
Good

i. Build a parse tree or a derivation from a context-free grammar.
Exam 3, Q9 – 83%
Final, Q6 – 97%
Good

j. Use the closure properties in arguments about languages.
Exam 1, Q6 – 89%
Exam 3, Q5 – 77%
Final, Q2 – 77%
Satisfactory

Level 3: Synthesis and Evaluation
Level 3 outcomes are those in which the student can apply the material in new situations. This is the highest level of mastery.
Upon successful completion of this course, students will be able to:

a. Compare regular, context-free, recursive, and recursively enumerable languages.
Quiz 5, Q7-8 – 75%
Final, Q15 – 93%
Good

b. Compare FA, PDA, and Turing machines.
Quiz 5, Q1-5 – 75%
Final, Q15 – 93%

Good

Analysis of the Results, and Instructor's 2020 Recommendations

The questions were mapped to the learning outcomes based on similar problems submitted by Dr. Vladik Kreinovich in the CQI for CS 3350 in Fall 2015. The averages indicate that the students who passed this class have satisfactory knowledge of all the necessary topics. Most of the outcomes were assessed multiple times – the first few times of assessment for some topics had lower results (unsatisfactory) than those nearing the end of the course. After continuous explanation, additional practice, and re-assessment the scores for the learning outcomes generally improved throughout. Most explanations were based directly on the required text *(Siper, 2006)* for the course.

Recommendations

- Adjust the learning outcomes to be more explicit:
    - Outcome 2e:
        - Current: Design an algorithm for a machine model to simulate another model.
        - *Proposed (Or similar to): Design an algorithm of a machine model, such as a State Machine, to simulate another machine model, such as a Turing Machine.*
    - Outcome 2h:
        - Current: Apply a parsing algorithm
        - Possibly consider adding additional context to this outcome. It does not appear to be clearly defined as to what the parsing algorithm should be applied to. Moreover, students may not be able to determine if they have mastered this skill.
- Split the learning outcomes with multiple points of assessment
    - Current: Convert a non-deterministic FA (respectively transition graph) into an equivalent deterministic FA, convert a transition graph or NFA into an equivalent regular expression, and convert a regular expression into an equivalent FA.
        - Create three outcomes from this single outcome
- Adjust the following outcome to align more with a level-1 outcome (Covered at a superficial level)
    - Current: Understand and explain the diagonalization process as used in proofs about computability.
        - Having students "explain" this process requires more than a level-1 coverage
    - Proposed: Understand the diagonalization process as used in proofs about computability.
- Explicitly state the significance that Automata, computability and formalization concepts have in Computer Science in learning outcomes

- Expand the explanation of regular, context-free, recursive, and recursively enumerable languages and the significance it has in Computer Science to be explicitly stated as part of learning outcomes.

- Expand the explanation of FA, PDA, and TM to show the importance and the role it can play in high-level Computer Science problems; include how they can be abstracted/generalized and easily transitioned to similar topics in Advanced Object-Oriented Programming.

- Expand on the importance of this course and how the material covered, and knowledge gained can be extended beyond this course. Describe the significance of knowing Automata theory and how the algorithms to describe such DFA, NFA, PDA, and TMs can be used elsewhere, especially in a non-theoretical Computer Science course.

Course Outcomes: CS 3360, Fall 2019

Course Number: CS3360
Course Title: Design and Implementation of Programming Languages
Course Instructor: Yoonsik Cheon
Instructional Assistant: Adrian Gomez Rodriguez (10 hours/week)

Course Description

In this course the students study concepts and examples of programming languages with the goal of acquiring the tools necessary for critical evaluation and rapid mastery of programming languages and constructs. The course attempts to balance theory and hands-on experience. The course surveys the constructs and capabilities typically found in modern programming languages, and several languages of different programming paradigms are presented in detail. By gaining an understanding on the range of possibilities likely to be encountered in a language, students will be prepared to learn new languages quickly throughout their careers. By understanding the implications of design alternatives, students will be better able to anticipate the problems likely to arise in using a new language. The presentation of design alternatives and trade-offs lays the groundwork for future advanced study of compilers and programming language semantics. To instantiate the discussion of general programming language characteristics, several languages will be presented in more detail: e.g., PHP (a Web scripting language), AspectJ (an aspect and object-oriented language), Haskell (a functional language), and Prolog (a logic-programming language). Students will gain practical experience with each programming paradigm by completing a programming project in each of the chosen languages.

Grade Distribution

The class started with 40+ students enrolled, and several students dropped before the course drop date. The table below summarizes the distribution of final letter grades that the remaining 36 students earned.

Grade	Pass				Failure
	A	B	C	D*	F
No. of students	12	9	9	3	3
Percentage	33%	25%	25%	8%	8%
Total	83% (30/36)			8% (3/36)	8% (3/36)

*D is a passing grade in CS 3360.

Out of 36 students who stayed in class, 33 students (92%) passed; 30 students (83%) earned a C or better. Three students (8%) failed, and they didn't take the final exam. Three students (8%) barely passed the course by earning a D.

Learning Outcomes

Assessment was done using exams and written homework assignments only. There were three programming assignments (PHP, AspectJ, Haskell), and they are not used for the assessment because the assignments require multiple skills including programming skills and are hard to map to specific, individual learning outcomes. There were frequent (12) on-line quizzes upon reading assignments, and they are not used either for the assessment.

For each outcome I measured the average score earned by students on the exam/homework questions mapped to the outcome, and the nominal acceptance value was 70% even if a D (60%) is a passing grade

in this course. Most outcomes had multiple questions mapped to them, and for these outcomes I picked a few most relevant, representative, or recent questions.

Level 1: Knowledge and Comprehension
Level 1 outcomes are those in which the student has been exposed to the terms and concepts at a basic level and can supply basic definitions. The material has been presented only at a superficial level. Upon successful completion of this course, students will be able to:

1a. Describe broad trends in the history of development of programming languages.
Met (Exam 1 question 2, 98%)

1b. Explain the stages of programming language interpretation and compilation.
Met (Exam 1 q1c 93%)

1c. Understand data and control abstractions of programming languages.
Met (84%; Exam 1 q10 72%, hw3 q4 97%, hw3 q5 84%)

1d. Understand how attribute grammars describe static semantics.
Met (Exam 1 q1d 76%%)

Level 2: Application and Analysis
Level 2 outcomes are those in which the student can apply the material in familiar situations, e.g., can work a problem of familiar structure with minor changes in the details. Upon successful completion of this course, students will be able to:

2a. Define syntax of a small context-free grammar in BNF and EBNF.
Met (82%; HW1 q4 94%, HW1 q6 85%, HW1 q8 67%)

2b. Define the syntax of a small subset of a programming language using BNF or EBNF.
Met Marginally (71%; Exam 1 q7 68%; hw1 q7 73%)

2c. Formally describe the dynamic semantics of small subset of programming languages, e.g., expressions and control structures.
Not met (Final q5 63%)

2d. Compare different approaches to naming, storage bindings, typing, scope, and data type.
Met (87%; hw2 q1 99%, hw2 q2 93%, hw2 q6 83%, hw2 q7 69%, hw2 q8 90%)

2e. Analyze design dimensions of subprograms, including parameter passing methods, sub-programs as parameters, and overload subprograms.
Met (75%; Exam 1 q8 68%, Final q7 81%)

Level 3: Synthesis and Evaluation
Level 3 outcomes are those in which the student can apply the material in new situations. This is the highest level of mastery. Upon successful completion of this course, students will be able to:

3a. Evaluate modern, representative programming languages critically with respect to design concepts, design alternatives and trade-offs, and implementation considerations for scope, binding, data types, expressions, control structures, subprograms, abstract data types, objects, concurrency structures, and exception handling.
Met (83%; hw2 q5 79%, hw3 q6 94%, hw3 q8 93%, hw3 q9 82%, hw3 q10 63%, hw3 q11 87%)

3b. Choose a suitable programming paradigm and language for a given problem or domain.

Met (86%; Exam 1 q3 73%; Final q2 98%)

All outcomes except for one were met, either marginally (70%-74%) or fully (75% or above). Outcome 2c (formal semantics; 63%) was not met, and outcome 2b (BNF; 71%) was met marginally.

Observations

1. The outcome 2c (formal semantics) was not met. It seems that students are not equipped with necessary mathematical backgrounds, especially first-order predicate logic, needed for the formal treatment of programming languages, i.e., axiomatic semantics.
2. I continued in incorporating the recommendations made in Fall 2015 and Fall 2017, including:
    - Spend more time on PHP to get students familiar with a new language, style, platform, and development environments, and have more in-class labs.
    - Incorporate pair programming to all programming assignments.
    - Adopt the application-driven approach.

    I adopted elements of the so-called *application-driven approach* in which new language features are introduced in the context of specific applications (programming assignments). I used this approach in teaching all three new languages, and all programming assignments were allowed to be done in pairs. Since Fall 2015, passing/success rates of students have been improved significantly and consistently as shown in the following table.

	A	B	C	D	F
Fall 2015	21% (8/39)	21 (8/39)	23% (9/39)	23% (9/39)	13% (5/39)
Fall 2017	33% (15/46)	30% (14/46)	26% (12/46)	7% (3/46)	4% (2/46)
Fall 2019	33% (12/36)	25% (9/36)	25% (9/36)	8% (3/36)	8% (3/36)

3. I was lucky to have an IA who was not only knowledgeable on the subject but also proactive in helping students, e.g., by contacting students who missed classes and offering help to those who were struggling.

Recommendations

1. Pay a special attention to the outcome 2c (formal semantics), and consider teaching formal semantics after logic programming (Prolog). This will provide a bit of background on first-order predicate logic which axiomatic semantics are based on.
2. Continue practicing/using: application-driven approach, pair programing, streamlined assignments, and the mobile/web quiz app.

Course Outcomes: CS 3360, Section 2 (Ward), Fall 2019

January 9, 2020

Course Number: CS3360
Course Title: Design and Implementation of Programming Languages

Course Description
In this course the students study concepts and examples of programming languages with the goal of acquiring the tools necessary for critical evaluation and rapid mastery of programming languages and constructs. The course attempts to balance theory and hands-on experience. The course surveys the constructs and capabilities typically found in modern programming languages, and several languages of different programming paradigms are presented in detail. By gaining an understanding on the range of possibilities likely to be encountered in a language, students will be prepared to learn new languages quickly throughout their careers. By understanding the implications of design alternatives, students will be better able to anticipate the problems likely to arise in using a new language. The presentation of design alternatives and trade-offs lays the groundwork for future advanced study of compilers and programming language semantics. To instantiate the discussion of general programming language characteristics, several languages are presented in more detail. Students will gain practical experience with each programming paradigm by completing a programming project in each of the chosen languages.

This report describes the outcome for Section 2, instructor Nigel Ward.

This semester there were, for the first time, two sections. The learning targets were the same and the coverage almost the same, but the teaching style was different. In particular, in this section all the programming assignments were done as individual projects, most of the programming assignments were smaller in scope, there were no daily quizzes, and more languages were introduced, namely PHP (a Web scripting language), bash (another scripting language), Haskell (a functional language), Prolog (a logic-programming language), and as a new experiment, Go, an imperative language with some interesting language features and the ability to easily give interesting assignments in concurrency.

Grade Distribution

The class started with 50 students enrolled. A few added and dropped (mostly dropped) in the first week, and then several more before the course drop date. 1 more had a true medical situation so at his request I dropped him. 3 stopped coming and did not take the final but neglected to drop, so they earned Fs. 1 decided to retake the class in the Spring but continued attending and even took the final; he also earned an F. The table below summarizes the distribution of final letter grades as submitted in Goldmine.

Grade	Pass				Fail
	A	B	C	D*	F
No. of students	8/42	17/42	9/42	4/42	4/42
Percentage	19%	40%	21%	10%	10%
Total	81% (34/42)			10%	10%

*D is a passing grade in CS 3360.

Assessment Method

Formal assessment was done using final exam questions only. However my comments below also reflect my observations of their performance on the 2 tests, 5 quizzes, 10 programming assignments, 7 homework problemsets, and several inclass exercises.

For each outcome I measured the average score obtained by students on the question relating to the outcome. I use a nominal acceptance value of 55%, rather than the traditional 70%, because my grading scale is non-traditional. In general, I give assignments and pose questions at a difficulty level such that typically 80% performance is an A. On final exams in particular I always try to challenge even the most advanced students, so almost all the questions had some element of synthesis. This semester on the final exam the raw score of the top performer, a very good student, was 76%, so clearly the traditional 70% criterion would not be appropriate.

Overall, interpretation of results is complicated by the fact that this semester's students were, overall, I feel, slightly weaker this semester than the previous two times I taught it. There were a good dozen students who were generally well engaged, but many others were consistently unprepared or often absent. The grade distribution was also weaker than in past semesters. Thinking that this may have reflected unintended increases in the difficulty of the assignments this semester, I lowered all thresholds by 0.04 points, e.g. I set the A/B threshold to 76%. These seemed right and also brought the grade distribution a little closer to that seen in the past. However conceivably this was too generous; without this adjustment there would have been two more Fs, and the percentages reported below would likely have been about 5% higher. However doing so would not have substantially affected the observations and recommendations.

Of the 39 students who took the final exam, I examined performance of 10. I discarded the top and bottom as potential outliers, and to obtain 10 samples that spanned the range, selected the students whose final exam scores were the 2nd highest, the 6th, the 10th, … and the 36th.

Learning Outcomes

Level 1: Knowledge and Comprehension Level 1 outcomes are those in which the student has been exposed to the terms and concepts at a basic level and can supply basic definitions. The material has been presented only at a superficial level. Upon successful completion of this course, students will be able to:

1.a. Describe broad trends in the history of development of programming languages.
Satisfactory (Final Exam, question 1, 68%)

1.b. Explain the stages of programming language interpretation and compilation.
Problematic (Final Exam, question 2 50%, question 21 40%)

1.c. Understand data and control abstractions of programming languages.
Satisfactory (Final Exam, question 3, 62%)

1.d. Understand how attribute grammars describe static semantics.
Satisfactory (Final q4, 74%)

Level 2: Application and Analysis Level 2 outcomes are those in which the student can apply the material in familiar situations, e.g., can work a problem of familiar structure with minor changes in the details. Upon successful completion of this course, students will be able to:

2a. Define syntax of a small context-free grammar in BNF and EBNF.
Problematic (Final q5 48%)

2.b. Define the syntax of a small subset of a programming language using BNF or EBNF.
Satisfactory (Final q6, 65%)

2.c. Formally describe the dynamic semantics of small subsets of programming languages, e.g., expressions and control structures.

Borderline Satisfactory (Final q8 100%, q9 60%, but q9 evaluated only Level-1 knowledge)

2.d. Compare different approaches to naming, storage bindings, typing, scope, and data types.
Mostly Satisfactory (Final q7 93%, q10 25%, q11 64%, q28 55%, q29 45%)

2.e. Analyze design dimensions of subprograms, including parameter passing methods, sub-programs as parameters, and overloaded subprograms.
Borderline Problematic (Final q14 55%, q15 53%)

Level 3: Synthesis and Evaluation Level 3 outcomes are those in which the student can apply the material in new situations. This is the highest level of mastery. Upon successful completion of this course, students will be able to:

3.a. Evaluate modern, representative programming languages critically w.r.t. design concepts, design alternatives and trade-offs, and implementation considerations for scope, binding, data types, expressions, control structures, subprograms, abstract data types, objects, concurrency structures, and exception handling.
Mostly Satisfactory (Final q16 80%, q17 10%, q18 40%, q20 72%)

3.b. Choose a suitable programming paradigm and language for a given problem or domain.
Satisfactory (Final q31 64%)

Observations and Outcome-Specific Recommendations

In summary, performance this semester was **generally satisfactory, with significant exceptions**.

Regarding **1b** (stages), the results may be misleading because both questions required synthesis, and question 21 also required understanding Go syntax. Nevertheless they indicate that covering this once in the first week of class and revisiting it once later was not sufficient: this topic needs reinforcement.

Regarding **2a** (writing BNF), the problem I gave them was to write a natural language grammar, which was intended to enable them to demonstrate a deep knowledge, but added some unpracticed complexity that doubtless confused some. Nevertheless performance on this topic was low overall. More problems sets are needed. In addition weaker students should be strongly encouraged to take Automata before this class.

Regarding one **2d** topic, dynamic scope, question 29 revealed that less than half the students could show understanding of it on the final. This was disappointing as they all seemed to understand it on the homework and the inclass exercise. While I could probably have demonstrated understanding had I recorded measurements of those results, clearly one inclass exercise and two homeworks on this from two different perspectives was not enough: they need more practice.

Regarding **2e (**subprograms), numerous conceptual-level discussions and exercises were clearly not enough. Maybe they further need to have programming exercises to drive these points home.

Regarding **3a** (critical evaluation), two of the measurements, q17 and q18, presupposed that the students remembered some basic points of Go syntax. Clearly many did not, so in future there should be an assessment instrument that doesn't depend on other knowledge.

Other Context and Comments

Some things were better this semester: a smaller class size, a better room, and a very good IA. Dr. Cheon was kind enough to handle the PHP lecture and in-class session, which I'm sure also helped.

Some things were worse: having only one IA limited the number of homework assignments I could give and get them feedback on, and 9am seemed to be a bad time slot, as many students were often late.

Regarding the languages taught.

- Bash was liked for its power
- Haskell was liked for the different way of thinking
- Prolog was neither much liked nor disliked: the assignments were small and fairly easy to complete.
- PHP was clearly a struggle. Many did not complete the PHP assignment. Since this is taught in DB (required now) and in Web based computing, it's less essential to keep it in this course.
- Go is an interesting small language, with good documentation and learning support. Students did fine on the first two assignments but many did little or nothing on the third. This may have been simply because it came late in the semester and they had other pressures.

General Recommendations

1) In future the specific weaknesses identified above be addressed, as suggested above.
2) Although performance on the attribute grammars test question was acceptable, overall their understanding of this was very weak. This outcome was put there, I believe, because 1) it reinforces CFG concepts, 2) it's a mathematically elegant way to introduce static semantics. Unfortunately half the class had not taken Automata first, so this put them at a disadvantage. Further, probably no more than 5-6 of the students got any benefit from the mathematical elegance. It is not clear that including attribute grammars has much value for the students. This should be dropped as an outcome, and only taught if the instructor feels that it has value for the other outcomes.
3) There is currently no outcome for strengthening general programming skills, but this was for me an important aim for the class. I'd like to see it added as a standard outcome, despite the difficulty of measuring it.
4) There is also currently no outcome for acquiring the ability to quickly learn a new language. Again this could be added as an outcome.
5) The fact that only 38 of the 50 students who started the course persisted and succeeded clearly shows some fundamental problem. We note that we investigated this in the last CQI cycle and made recommendations in our "Languages Area Report, Fall 2017" (dated April 6, 2018) and revisited them in the ad hoc 3360 report on the Spring 2019 drop situation, dated May 20, 2019. This problem has clearly not gone away, and we urge the department to seriously consider our earlier recommendations, repeated below:

    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".*

University of Texas at El Paso, Department of Computer Science
CS 4342 Database Management - Fall 2019

CQI- Analysis of Outcomes

Instruction Team

Instructor: Natalia Villanueva Rosales
Teaching Assistant: Daniel Ornelas (10 hours per week).
Instructional Assistant: Adrián Gómez Rodríguez (10 hours per week).

Course Catalog Description

Introduction to database fundamentals, modeling, the use of database management systems for applications, and current trends for data management including: relational algebra, entity-relationship models, relational data models, semi-structured data models, schema design, query processing, data integrity, privacy, security and data analytics.

Grading

1. Exams 50%.
2. Project and Assignments including presentations 40%.
3. Class participation and activities 10%.

Grade Distribution

The class started with 55 students enrolled, 49 UG and 6 Graduates in the cross-listed graduate course. One student dropped during the first two weeks and the spot was used by a student in the waiting list. The table below summarizes the distribution of final letter grades that the 49 students earned.

Grade	Pass				Failure
	A	B	C	D	F
No. of students	15	22	10	0	2
Percentage	30.6%	44.8%	20.40%	0%	4.0%
Total	96% (47/49)				4.0% (2/49)

Out of 49 students registered in the class, 47 students (96%) passed and earned a C or better. Two students (4%) earned an F after taking the final exam.

Assessment Method

Formal assessment was done through midterm, final exams, quizzes, assignments and related homework. Comments and recommendations are also informed by meetings with students' team twice per semester, office hours and meetings with TA and IA. In total, students had a midterm and a final exam, four (large) quizzes, weekly feedback and weekly (short) quizzes. Short quizzes are not considered in the assessment method as well as some homework and activities that were graded but served the purpose of providing context on the level of understanding of students throughout the semester. In this assessment I use 60%

as a nominal acceptance value given that a student can pass with a D and use the average and the most representative and recent questions to analyze the progress of the class throughout the semester.

An important element of this course is the design and development of a team project that enables students to apply the topics introduced in the course in a real-world application. The course project also aims to improve students' teamwork, writing, and communication skills through the creation of progress reports and presentations. Thus, evaluated tasks related to the course project are mapped in the course outcomes and included in the assessment. It is important to note that while the project provides an excellent opportunity to obtain skills needed in this course, there is no lab associated with the course. Students are required to meet regularly with the instruction team (instructor, TA and IA) to review progress and weekly on-class and homework exercises are provided for students. In my experience, the project provides an excellent opportunity for students to develop technical and professional skills but requires commitment of time from students, which makes this course challenging for some of them.

Most of the questions of exams were implemented in Blackboard and the item analysis was used, which provides average for each question but does not allow to disaggregate the analysis through groups. The Blackboard shell includes six additional students that are a graduate students registered in the cross-listed class of this course. Their grades are two As, two Bs and two Cs, and thus will not change significantly the outcomes. If disaggregation by courses is allowed by Blackboard, and update to the report will be provided.

Course Outcomes

Divided into the following three broad levels of Bloom's taxonomy:

Level 1: Knowledge and Comprehension.

Level 1 outcomes are those in which the student has been exposed to the terms and concepts at a basic level and can supply basic definitions. The material has been presented only at a superficial level. Upon successful completion of the course, students will be able to:

1a. Describe and compare data models (e.g., Entity-Relationship model, relational model, semistructured model), how they have been used in the past, and how they are currently used for data management.
Midterm Q1, Q8, Q10.
Final Q2, Q3.
Average: 80.6% with 91.5% average in the final exam questions.
Met.

1b. Describe the components of a database system, the most common designs for core database system components including the query optimizer query executor, storage manager, access methods, and transaction processor their most common design, and give examples of their use.
Final Q43, Q44, Q45.
Average: 64.33%
Marginally Met.

1c. Cite the basic goals, functions, and models of database systems.

Final Q46.
Average: 75%
Met.

1d. Identify database languages and interfaces for data management.
Final Q4.
Assignment 3 (81%).
Average 70.5%. Assignment weight should be higher given the complexity of the task.
Met

1e. Critique an information application with regard to satisfying user information needs.
Final Q31.
Assignment 3.
Average 84.5%.
Met.

1f. Explain uses of declarative queries.
Queries homework.
Final Q41, Q47.
Average 80.33%.
Met.

1g. Identify database architectures (e.g., centralized, distributed, web-based).
Final Q48.
Average 88%.
Met.

1h. Identify current trends of data management paradigms.
Final Q58.
Presentation on data paradigms.
Average 71%.
Met.

1i. Describe technical solutions to the challenges in information privacy, integrity, security, and preservation.
Final Q53.
Average 87%.
Met.

1j. Identify major database management systems functions and describe their role in a database system.
Final Q52.
Average 74%.
Met.

1k. Identify the careers/roles associated with information management
Final Q49, Q50.
Average 96.5%.
Met.

Level 2: Application and Analysis.
Level 2 outcomes are those in which the student can apply the material in familiar situations, e.g., can work a problem of familiar structure with minor changes in the details. Upon successful completion of the course, students will be able to:

2a. Demonstrate uses of explicitly stored metadata/schema associated with data.
Data models homework.
Average 0.76%.
Met.

2b. Use relational algebra and set theory that are supported in the relational model.
Relational algebra exercise.
Final Q23, Q26, Q27, Q42.
Average 65.53%.
Marginally met.

2c. Use a relational query language (e.g. SQL) to elicit information from a database.
Quiz on SQL basics.
Final Q29, Q30, Q33, Q37, Q39, Q50.
Assignment 2, Assignment 3.
Average 74.2%.
Met.

2d. Normalize a database using the 1st, 2nd, and 3rd normal forms.
Midterm Q1, normalization section.
Final Q28, Q34, Q35, normalization section.
Average 74.3%
Met.

2e. Justify the use of relational or non-relational data management systems based on the requirements of an applications.
Assignment 1.
Average 81%.
Met.

2f. Demonstrate the ability to work in teams
Final presentation.
Teamwork evaluation.
Average 86.5%.
Met.

Level 3: Synthesis and Evaluation.

Level 3 outcomes are those in which the student can apply the material in new situations. This is the highest level of mastery. Upon successful completion of the course, students will be able to:

3a. Design a database system from a problem statement to a conceptual, high-level data model (e.g., Entity-Relationship) using standard notation and modeling principles.
Midterm Q2, Q4, Q7, Q11-Q26.
Final Q7, Q9-Q12, Q14-Q17.
Assignments 1-3.
Average 82.2% with final questions above 90% average.
Met.

3b. Design a relational data model from a conceptual data model.
Midterm Q6, Q27-Q31.
Final Q8, Q18-Q22.
Assignment 1-3.
Homework ER to relational model.
Average 70%.
Met.

3c. Design and implement a relational data model in a relational database schema using a database management system.
Midterm Q5.
Assignment 2 &3.
Average 85%.
Met.

3d. Design and implement an interface for a database system applying best practices for usability, privacy and security.
Final Q6.
Assignment 3.
Average 89.5%.
Met.

Analysis and Recommendations

All the outcomes were satisfactorily met. Two outcomes (9.5%; 1b and 2b) were marginally met with the threshold of 60% and would have not be met with a threshold of 70%. Given the focus on the course on providing students with skills and knowledge to achieve outcomes in level 3, it is somehow expected that students will not memorize concepts (1b, describe components of DBMS). It is recommended to reinforce concepts and do more assessments throughout the semester for this outcome.

Relational algebra and set theory (2b) is always a hard topic for students, but still was met through providing many exercises. More focus on this area is recommended as well as provide more background needed for those students that may lack of it.

All teams submitted a functional database management system consisting on a backend and frontend and high-quality documents at the end of the semester which is reflected in a higher average in outcomes at the applications and synthesis level. It is recommended to continue with project-based learning for this course to overcome the lack of a lab. The role of prepared TAs and IAs is key in this course to provide constructive and timely feedback during the semester, identify students at risk and provide ad-hoc sessions needed, specially with the growth of enrollment of this class resulting from its change to a course course.

The team project provides the opportunity for students to develop professional skills. Through the interaction with students, it was noticed that teamwork is the most challenging skill to develop. Providing more tools to successfully communicate and participate in teams is recommended. Including sessions for teamwork development skills would be desired but hard to achieve given the amount of topics to be covered.

The great advantage to use Blackboard is the ability to provide immediate feedback at the price of having very limited options to create questions. I noticed that a few questions were confusing and had to re-grade manually. It is recommended to continue to include written examinations and work with Blackboard Central providers at UTEP to find additional tools for assessment.