2013-14 CptS Student Work Assessment Report

Chris Hundhausen

Chair, CS & EECS Assessment Committee

1. Background

Per the schedule set forth in our Assessment Manual, each academic year we select a subset of student work in targeted courses for assessment relative to a set of student outcomes. Table 1 presents the courses and student outcomes targeted in the 2013-14 academic year.

Course	Instructor	Enrollment	Targeted Outcomes
CptS 260 "Computer Architecture" (Sp14)	Eric Wang	57	A,C
CptS 350 "Algorithms" (Fa13)	Zhe Dang	57	A, B
CptS 355 "Programming Languages" (Sp14)	Seth Long	54	B, C, I
CptS 423 "Senior Design Project II" (Sp14)	Sakire Arslan-Ay	29	B, C, I

Table 1. Courses and outcomes targeted for assessment during 2013-14 academic year

Our new sampling strategy was developed in the Fall of 2013 in response to our most recent ABET visit. At that time, we decided to collect student work samples in CptS 350, which was offered only in the fall semester of 2013-14. However, due to the late notice, we were unable to collect student work in that course in the fall of 2013. Thus, in 2013-14, we were able to obtain coursework from only three out of the four targeted courses: CptS 260, CptS 355, and CptS 423.

2. Assessment Methodology

Figure 5 (p. 7) in the Assessment Manual presents the process by which we assess student work samples. On May 12, 2014, the CS Assessment Committee (Chris Hundhausen, Ananth Kalyanaraman, Larry Holder, and Carl Hauser) met with the instructors of CptS 260 (Eric Wang), CptS 355 (Seth Long), and CptS 423 (Sakire Arslan-Ay). The meeting with each individual instructor lasted roughly 45 minutes to one hour. To prepare for each individual meeting, course instructors were asked to furnish the Assessment Committee with samples of all student work for 2-3 course deliverables of their choosing.

At each individual meeting, instructors first described each deliverable they chose for assessment. They then projected selected student work samples onto a large screen for inspection and discussion. Although the Assessment Manual states that each committee member was to independently rate these samples on the 4-point assessment scale, we opted to deviate from this process because it quickly became apparent that we would not be able to assess the volume of student work supplied by each instructor. Instead, we decided to focus the discussions on the following question:

> *Are students who are passed the course with a C or higher demonstrating a "capable" level of competence (3 on our 4 point scale) in these assignments with respect to the targeted outcomes?*

Given this focus, the Assessment Committee's preliminary discussions with instructors considered the work of "borderline" students: those who performed just well enough to pass the class with a C or better. After preliminary discussions about these borderline cases and strategies for assessing them relative to our 4-point scoring scale, we decided that the most efficient course of action was to have each instructor prepare a brief assessment report following the meeting. The reports were to focus on assessing the work samples of students at the cusp of passing the course relative to our 4-point scoring scale, in order to determine whether these students were demonstrating "capable" (3 point) performance. If these students

achieved capable performance, we reasoned that any student who passed the course must have also attained "capable" performance.

The instructors were asked to write these reports in the weeks following the meeting, and to submit them to the Assessment Committee Chair for analysis.

3. Results

Two of the three instructors of targeted courses (Eric Wang and Sakire Arslan-Ay) submitted their reports. Because Seth Long, the CptS 355 instructor, left the university in the summer of 2014, I had Carl Hauser, who normally teaches CptS 355, write the assessment report results for CptS 355.

CptS 260

Eric Wang's assessment of CptS 260 is included as Appendix A of this report. Eric assessed the work samples of 19 students near the pass/fail boundary: 16 who received a B, B-, C+ or C in the course, and another 3 who received a C- in the course. His report presents tables that provide his ratings of these students' work on three course assignments (two programming assignments and the final exam, constituting 50% of students' overall course grade) with respect to outcomes A and C. For each passing student grade level (C or above), Table 2 summarizes Eric's ratings for each Outcome.

Student Grade Level	*N*	Outcome A	Outcome C
B	11	2.74	2.87
B-	2	2.30	2.29
C+	1	1.95	1.84
C	2	2.48	2.44

Table 2. Average rating of 15 students' work on three assignments in CptS 260

As can be seen, under Eric's ratings, students who passed the course with a B or C generally did reach the "capable" threshold. In trying to interpret these results, I consulted with Eric, who had this to say about his ratings:

> Keep in mind that we considered only the less-capable students in the class, on the most challenging of the [course] deliverables. If their ratings had been mostly "capable", they would have earned higher grades (and thus excluded themselves)… Below a grade of B, students frequently exhibit 2 pt ("Needs Improvement") performance. In our estimation, that's why they're not A students.

When pressed further about the relationship between his course grades and his ABET assessments, Eric stated the following:

> It could also be the case that [I am] lenient in assignment grades to undergrads, but I was demanding in my assessment on the 4-pt scale…Hence a great deal of student work is passing by [my] grading standards, but deos not meet my threshold of being "capable."

These explanations account for Eric's generally low ratings of B and C students on the 4-point assessment scale. At the same time, they the issue of the need to normalize assessment ratings across courses and instructors—an issue that we discuss in Section 4 below.

CptS 355

Carl Hauser's assessment of CptS 355 is included as Appendix B of this report. Carl considered th e work of 13 students submitted for two course deliverables: a programming assignment that involved the construction of a simple PostScript interpreter, and a "quiz" with small Scheme programming problems. With respect to Outcomes B, C, and I, Carl assessed the work of eight students who passed the course with a C (7) or C+ (1), along with the work of five of the 15 students who passed the course with a B. For each student grade level considered, Table 3 summarizes Carl's rating for each Outcome.

Student Grade Level	*N*	Outcome B	Outcome C	Outcome I
B	5	2.75	2.67	2.89
C/C+	8	2.75	2.67	2.89

Table 3. Average rating of 15 students' work on three assignments in CptS 260

As was the case for CptS 260, CptS 355 students who passed the course with a B or C generally did not attain the 3.0 "capable" threshold. In fact, no difference can be observed between the outcomes of B-level and C-level students with respect to Outcomes B, C, and I. The interpretation of the data offered by Carl is that, even though students in the course did not always exhibit competency, they often did. Noting that students often learn at different rates, Carl did not exhibit concern that 355 students "sometimes missed the mark."

CptS 423

Based on the Assessment Committee's discussions with Sakire Arslan-Ay, we decided to have her perform a more detailed assessment of work of the two lowest-performing senior design teams: Team Gondor and Team Red Dragon. Students on these two teams received grades of C, C+, B-, A-, C, B, and A. Thus, all students on these teams passed the course with a C or better. Our rationale was that if students on these low-achieving teams could achieve target performance levels with respect to the three targeted outcomes (B, C, and I), then students on the other, higher-achieving teams must also have achieved target performance levels.

Sakire's assessment reports of these two teams are included as Appendix C and of this report. In addition to considering the final reports of these two teams (included as Appendices D and E of this report), Sakire's assessment considered other relevant deliverables and facts, as noted in her report.

Table 2 presents a summary of Sakire's assessment. As the table indicates, the two lowest-achieving senior design teams all performed above the "capable" threshold with respect to the three target outcomes. We therefore conclude that all students in the course who achieved a C or higher also performed above the "capable" threshold.

Team	B	C	I
Gondor	3.17	3.0	3.5
Red Dragon	3.25	3.78	3.5

Table 2. Average ratings of two lowest-achieving senior design teams with respect to Outcomes B, C, and I

4. Discussion and Recommendations

Section 2A of our Assessment Manual (p. 6) specifies that the average ratings of student work with respect to each targeted outcome should be 3.0 ("capable"). Given the data we have on students in CptS 260, 355, and 423, we can draw the following conclusions:

1. Average ratings of the work of low-performing students fell above the 3.0 threshold for Outcomes B and I (in CptS 423 only).
2. Average ratings of the work of low-performing students fell below the 3.0 threshold for Outcomes A, C, and I (in CptS 355 only).

We also observed that, since we had course instructors, and not the Assessment Committee, ultimately assign ratings to student work in two out of the three cases (Carl Hauser, who assessed 355 student work, is an Assessment Committee member), it was likely that the ratings were not assigned reliably. This possibility was confirmed by the comments of Eric Wang (CptS 260 instructor), who stated that he used a more stringent scale in his ABET ratings than he did in his grading.

Given these results and observations, the Assessment Committee proposes the following recommendations, to be reviewed by the faculty at the 2014 Faculty Retreat in August:

1. Per the Assessment Manual (Figure 6, p. 8, put Outcomes A and C on watch for the 2014-15 assessment cycle. (Since Outcome I was assessed to be well above 3.0 in CptS 423, and nearly at 3.0 in CptS 355, we will not put it on watch.)
2. Consider whether to require average assessment ratings to meet a threshold of 3.0. Perhaps a threshold of 2.75 would be more reasonable, given the variability in raters and also the variability in student work
3. In order to achieve more uniformity in assessment ratings, consider going back to the assessment approach outlined in our Assessment Manual: Require instructors of targeted courses to furnish the Assessment Committee with the work of all C-level students for two to three course deliverables. At the assessment meeting, the instructor should describe the assignment and identify components to be assessed relative to targeted outcomes. Members of the Assessment Committee should then perform the assessments independently. A discussion should follow, in order to ensure that the assessment ratings were furnished according to consistent criteria.

.

Appendix A:

Eric Wang's Assessment Report for CptS 260

Assessment of the Pass/Fail (B/C–) Boundary for Cpt S 260 Introduction to Computer Architecture (Spring 2014) for ABET Accreditation

Eric Wang

27 May 2014

Methodology

In previous semesters, we have used sampling to make the assessment task manageable, choosing exactly 2 students per tier for 3 tiers, representing good (A), average (B), and poor (C) students, based on their final course grades. All assessment is performed on 3 selected course deliverables.

For the Spring 2014 semester, we present instead a **comprehensive** summary of every student around the "pass/fail" boundary. This comprises the following final course grades:

- Pass (16 students): average (B/B–) and poor (C+, C)
- Fail (3 students): C–

We assess 3 selected course deliverables. The result of this assessment is a histogram of their performance indicators over each course deliverable. From this, we infer some general statements for each performance indicator.

The 3 selected course deliverables are:

• HW5 Quicksort	200/1200 HW pts × 50% = 1/12	= 10/120 of grade
• HW9 Frogger	300/1200 HW pts × 50% = 1/8	= 15/120 of grade
• Final Exam	30% of grade	= 36/120 of grade
	Total	= 61/120 of grade

These 3 deliverables represent 50.8% of the course grade only.[1] Furthermore, these 3 course deliverables were selected in large part because of their challenging nature (including two semester projects). Hence, some students underperformed on these deliverables compared to their overall coursework, and thus earned a higher course grade than their performance here may indicate.

Since we deliberately restrict our focus to the students adjacent to the "pass/fail" boundary, the "pass" tier comprises B–C students (only), who tend to score at best 3 on our 4-point scale.

[1] The remaining 49.2% was spread over 8 course deliverables, not considered here: 7 HWs × 100 pts each (on average) = 35/120, plus a Midterm Exam at 20% = 24/120.

For each deliverable, we present:

a) The major portions of each deliverable (each portion is assessed separately);
b) a brief explanation of the performance indicators relevant to each portion;
c) the individual assessments for all 19 students (with identities obfuscated);
d) a histogram of performance indicator sums for two tiers of "pass" and "fail".

Note that the histograms simply sum the 4–point scale for all students in each tier, so it depicts average assessment. For HWs (but not the Final Exam), we extend the scale with 0s for students who ignored or made no attempt on a deliverable or portion thereof, as these may reflect external factors other than a true lack of comprehension or ability. Table cells are color-coded on our 4-point scale as shown below. For histogram table cells, we show the color of the ceiling $\lceil$ *total / # of non-0 students* $\rceil$, rather than rounding or truncating.

4	exemplary
3	capable
2	needs improvement
1	unsatisfactory
0	*no attempt made*

1. Deliverable: HW5 Quicksort

This is a programming assignment using MIPS assembly language. It is the 1st of 2 semester projects, worth twice as much as a typical homework. It has two major portions:

5:123	implementation	(swap, partition, quicksort) Provided: *brutepart*, *brutesort*
5:45	test harness	(quicktest)

- The 1st portion [123] is to implement the quicksort algorithm for an array of integers in memory. For convenience in testing (due to the relative tedium of generating formatted output in MIPS assembly), two "pre-existing code" files were provided, which were simple test loops that called the student's implementation and pretty-printed the results.
- The 2nd portion [45] is to implement a test harness, which consists of (a) a declarative array of test cases, each one using "object-like" (actually, C struct) memory layout and notation, and (b) a loop over this array.

HW5 covers *(or not)* these performance indicators:

A1. Design an algorithm and data structure;

C4. Implement the selected approach

C5. Validate a solution

A2. Understand computer hardware architecture at instruction level

- [all] Visualize stack frame and stack pointer register $sp
- [all] Visualize arrays in memory, pointers into arrays, pointer arithmetic

A3. *Continuous and* discrete mathematics to analyze correctness and performance

- [all] Discrete math only: subscript-to-address conversion, iteration over array, fencepost computation

A4. Synthesize modeling, simulation, prototyping to refine design concept

- [all] Pseudo-code as prototype
- [45] Test case memory layout as a template for memory allocation

A5. Implement using multiple paradigms

- *[123] MIPS assembly language is imperative only.*
- [45] We (deliberately) extend MIPS with object-like memory management.

A6. Use pre-existing code libraries

- [123] Sample test files (call as function; link into executable)
- *[45] none*

A7. Modern development environments, debuggers

- MARS compiler: Detects compile-time errors
- MARS debugger

A8. Develop test plan

- [all] Incremental testing
- [123] Use sample test files
- [45] Write own test cases

C1. Recognize when theory dictates that a solution is impractical or impossible

- [123] Recursion must make problem instance smaller
- [45] Array of variable-length “objects” complicates iteration

C2. Generates multiple design concepts

C3. Determines tradeoffs among design alternatives and selects an approach

- We generally see only the final submitted version of a programming assignment.

p.i.	A1	A2	A3	A4	A5	A6	A7	A8	C1	C2	C3	C4	C5	#	
pass	37	37	33	24	*n/a*	25	36	35	31	*n/a*	*n/a*	36	35	**15**	**Gr**
5:123	3	3	2	1	*n/a*	2	2	1	0	*n/a*	*n/a*	1	1	b14	B
5:123	1	1	2	1	*n/a*	2	1	1	1	*n/a*	*n/a*	1	1	f68	B
5:123	1	2	2	2	*n/a*	0	2	2	1	*n/a*	*n/a*	2	1	w01	B
5:123	3	3	3	1	*n/a*	0	3	3	3	*n/a*	*n/a*	3	3	b47	B
5:123	2	2	2	1	*n/a*	0	2	3	2	*n/a*	*n/a*	2	3	m50	B
5:123	3	3	3	1	*n/a*	3	3	3	3	*n/a*	*n/a*	3	3	b69	B
5:123	4	4	3	4	*n/a*	3	3	4	4	*n/a*	*n/a*	4	4	f30	B
5:123	4	3	3	3	*n/a*	3	3	4	4	*n/a*	*n/a*	4	4	m15	B
5:123	2	3	2	1	*n/a*	3	3	2	2	*n/a*	*n/a*	3	2	z69	B
5:123	2	2	2	1	*n/a*	3	3	2	3	*n/a*	*n/a*	3	3	l38	B
5:123	3	2	3	3	*n/a*	0	3	3	3	*n/a*	*n/a*	3	3	f10	B
5:123	3	3	2	1	*n/a*	3	2	3	2	*n/a*	*n/a*	3	3	h64	B-
5:123	3	3	2	1	*n/a*	3	2	2	3	*n/a*	*n/a*	2	2	c74	B-
5:123	2	1	2	1	*n/a*	0	2	1	0	*n/a*	*n/a*	1	1	c85	C+
5:123	0	0	0	0	*n/a*	0	0	0	0	*n/a*	*n/a*	0	0	s47	C
5:123	1	2	0	2	*n/a*	0	2	1	0	*n/a*	*n/a*	1	1	l36	C
5:123	1	2	2	1	*n/a*	0	1	1	0	*n/a*	*n/a*	1	1	h55	C-
5:123	0	0	0	0	*n/a*	0	0	0	0	*n/a*	*n/a*	0	0	s85	C-
5:123	0	0	0	0	*n/a*	0	0	0	0	*n/a*	*n/a*	0	0	l00	C-
fail	1	2	2	1	*n/a*	0	1	1	0	*n/a*	*n/a*	1	1	**1**	

p.i.	A1	A2	A3	A4	A5	A6	A7	A8	C1	C2	C3	C4	C5	#	
pass	8	8	8	6	8	*n/a*	8	8	6	*n/a*	*n/a*	8	8	**2**	**Gr**
5:45	0	0	0	0	0	*n/a*	0	0	0	*n/a*	*n/a*	0	0	b14	B
5:45	0	0	0	0	0	*n/a*	0	0	0	*n/a*	*n/a*	0	0	f68	B
5:45	0	0	0	0	0	*n/a*	0	0	0	*n/a*	*n/a*	0	0	w01	B
5:45	0	0	0	0	0	*n/a*	0	0	0	*n/a*	*n/a*	0	0	b47	B
5:45	0	0	0	0	0	*n/a*	0	0	0	*n/a*	*n/a*	0	2	m50	B
5:45	0	0	0	0	0	*n/a*	0	0	0	*n/a*	*n/a*	0	0	b69	B
5:45	4	4	4	3	4	*n/a*	4	4	3	*n/a*	*n/a*	4	3	f30	B
5:45	4	4	4	3	4	*n/a*	4	4	3	*n/a*	*n/a*	4	3	m15	B
5:45	0	0	0	0	0	*n/a*	0	0	0	*n/a*	*n/a*	0	0	z69	B
5:45	0	0	0	0	0	*n/a*	0	0	0	*n/a*	*n/a*	0	0	l38	B
5:45	0	0	0	0	0	*n/a*	0	0	0	*n/a*	*n/a*	0	0	f10	B
5:45	0	0	0	0	0	*n/a*	0	0	0	*n/a*	*n/a*	0	0	h64	B-
5:45	0	0	0	0	0	*n/a*	0	0	0	*n/a*	*n/a*	0	0	c74	B-
5:45	0	0	0	0	0	*n/a*	0	0	0	*n/a*	*n/a*	0	0	c85	C+
5:45	0	0	0	0	0	*n/a*	0	0	0	*n/a*	*n/a*	0	0	s47	C
5:45	0	0	0	0	0	*n/a*	0	0	0	*n/a*	*n/a*	0	0	l36	C
5:45	0	0	0	0	0	*n/a*	0	0	0	*n/a*	*n/a*	0	0	h55	C-
5:45	0	0	0	0	0	*n/a*	0	0	0	*n/a*	*n/a*	0	0	s85	C-
5:45	0	0	0	0	0	*n/a*	0	0	0	*n/a*	*n/a*	0	0	l00	C-
fail	0	0	0	0	0	*n/a*	0	0	0	*n/a*	*n/a*	0	0	**0**	

The histograms for HW5 are as follows.

tier	p.i.	A1	A2	A3	A4	A5	A6	A7	A8	C1	C2	C3	C4	C5	#
pass	**5:123**	37	37	33	24	*n/a*	25	36	35	31	*n/a*	*n/a*	36	35	15
fail	**5:123**	1	2	2	1	*n/a*	0	1	1	0	*n/a*	*n/a*	1	1	1
pass	**5:45**	8	8	8	6	8	*n/a*	8	8	6	*n/a*	*n/a*	8	8	2
fail	**5:45**	0	0	0	0	0	*n/a*	0	0	0	*n/a*	*n/a*	0	0	0

On average, the "pass" tier (15 of 16 students) exhibited > 2.0-point performance on all performance indicators for both portions, except for these:

- A4 *Synthesize the results of … prototyping.* (24 / 15 = 1.6 points) We apply this item only to the writing and use of pseudo-code in C or similar, as a "scaffold" for their ensuing MIPS code. Conclusion: We did not emphasize this mindset strongly enough to cause a significant shift in program development habits among students in these tiers.
- A6 *Use pre-existing code … as components of programs.* (25 / 15 = 1.67 points) Some students simply made too little progress to have testable code, so they did not even try to link the provided test functions. Hence, the low score here could indicate deeper failures in programming or debugging skill (A1, A2, A7, A8, C5), or insufficient effort.
- C1 *Recognize when theory dictates that a solution is impractical.* (31 / 15 = 2.07, barely above 2 "needs improvement") Some students don't understand recursion, and made elementary programming errors. HW5 is the most strenuous usage of recursion in this course, so it best reveals this weakness.

Comparing the "pass" and "fail" tiers for the 1st portion [5:123], we can roughly infer that students who pass have > 2.0-point performance across all performance indicators (except those noted above), while students who fail showed ≤ 2.0-point performance, or completely ignored the assignment.

For the 2nd portion [5:45], only 2 of 19 students attempted it at all, and both did well. The others made no attempt, due to factors including: insufficient progress in the 1st portion; lack of time; simple oversight *(based on ensuing student emails);* or apathy. No students attempted it and failed, so we can't conclude that they were deficient, except in a most general sense that the difficulty threshold prevented trivial attempts.

2. Deliverable: HW9 Frogger

This is a programming assignment using MIPS assembly language. It is the 2nd of 2 semester projects, worth three times as much as a typical homework. The gist of HW9 is to implement a rudimentary "1-line" version of the Frogger arcade video game, using memory-mapped keyboard input and output to a display window, and interrupt handling for keystrokes. It comprises four major portions:

9:126	memory-mapped I/O to DISPLAY, from KEYBOARD
9:345	game representation and logic
9:79	main loop and shared-memory global variable
9:8	interrupt handler

HW9's sub-portions cover the following performance indicators.

p.i.	A1	A2	A3	A4	A5	A6	A7	A8	C1	C2	C3	C4	C5
9:126			*n/a*	*n/a*	*n/a*				*n/a*	*n/a*	*n/a*		
9:345			*n/a*	*n/a*	*n/a*	*n/a*			*n/a*	*n/a*	*n/a*		
9:79			*n/a*	*n/a*		*n/a*				*n/a*	*n/a*		
9:8			*n/a*	*n/a*				*n/a*		*n/a*	*n/a*		

A1, A7, C4, C5 are generally relevant to all portions of a programming assignment.

A2. Understand computer hardware architecture at instruction level

- [126] Memory-mapped I/O
- [345][79] Shared memory using global variable
- [8] Interrupt status/cause registers and kernel code

A3. Continuous and discrete mathematics to analyze correctness and performance

- *Only trivial incrementing is needed throughout.*

A4. Synthesize modeling, simulation, prototyping to refine design concept

- *The finished design is already provided in the HW9 assignment handout.*

A5. Implement using multiple paradigms

- *[126][345] MIPS assembly language is imperative only.*
- [79][8] Main loop + interrupt handler is a special case of **two-thread** programming, which requires similar cognitive effort.

A6. Use pre-existing code libraries

- [126] MIPS memory-mapped IO presents a fixed hardware/software interface, which must be followed.
- *[345][79] none*
- [8] Interrupt handler sample code is given in textbook. Students must understand, copy, and modify it.

A8. Develop test plan

- [126][345][79] Incremental testing using direct calls instead of interrupts
- *[8] Cannot assess due to lack of visible traces*

C1. Recognizes when theory dictates that a solution is impractical or impossible

- *[126][345] not applicable*
- [79] Main "thread" is disjoint from interrupt handler "thread", no direct calls
- [8] Must determine cause, ignore exceptions, save all registers

The raw student assessments are as follows. For consistency, we present the students in the same order throughout, even if several of them made no attempt.

p.i.	A1	A2	A3	A4	A5	A6	A7	A8	C1	C2	C3	C4	C5	#	
pass	37	35	*n/a*	*n/a*	*n/a*	42	41	40	*n/a*	*n/a*	*n/a*	42	38	**14**	**Gr**
9:126	3	3	*n/a*	*n/a*	*n/a*	3	3	3	*n/a*	*n/a*	*n/a*	4	3	b14	B
9:126	3	3	*n/a*	*n/a*	*n/a*	3	3	3	*n/a*	*n/a*	*n/a*	3	3	f68	B
9:126	4	4	*n/a*	*n/a*	*n/a*	4	3	4	*n/a*	*n/a*	*n/a*	4	4	w01	B
9:126	3	3	*n/a*	*n/a*	*n/a*	3	3	3	*n/a*	*n/a*	*n/a*	3	3	b47	B
9:126	3	2	*n/a*	*n/a*	*n/a*	3	3	3	*n/a*	*n/a*	*n/a*	3	3	m50	B
9:126	2	3	*n/a*	*n/a*	*n/a*	3	3	3	*n/a*	*n/a*	*n/a*	3	3	b69	B
9:126	2	2	*n/a*	*n/a*	*n/a*	3	3	2	*n/a*	*n/a*	*n/a*	3	2	f30	B
9:126	3	2	*n/a*	*n/a*	*n/a*	3	3	3	*n/a*	*n/a*	*n/a*	3	3	m15	B
9:126	3	2	*n/a*	*n/a*	*n/a*	3	3	4	*n/a*	*n/a*	*n/a*	3	3	z69	B
9:126	2	2	*n/a*	*n/a*	*n/a*	3	3	3	*n/a*	*n/a*	*n/a*	3	3	l38	B
9:126	0	0	*n/a*	*n/a*	*n/a*	0	0	0	*n/a*	*n/a*	*n/a*	0	0	f10	B
9:126	2	2	*n/a*	*n/a*	*n/a*	3	3	2	*n/a*	*n/a*	*n/a*	3	2	h64	B-
9:126	2	3	*n/a*	*n/a*	*n/a*	3	3	2	*n/a*	*n/a*	*n/a*	2	2	c74	B-
9:126	2	1	*n/a*	*n/a*	*n/a*	2	2	2	*n/a*	*n/a*	*n/a*	2	1	c85	C+
9:126	3	3	*n/a*	*n/a*	*n/a*	3	3	3	*n/a*	*n/a*	*n/a*	3	3	s47	C
9:126	0	0	*n/a*	*n/a*	*n/a*	0	0	0	*n/a*	*n/a*	*n/a*	0	0	l36	C
9:126	2	1	*n/a*	*n/a*	*n/a*	2	2	1	*n/a*	*n/a*	*n/a*	2	1	h55	C-
9:126	0	0	*n/a*	*n/a*	*n/a*	0	0	0	*n/a*	*n/a*	*n/a*	0	0	s85	C-
9:126	2	3	*n/a*	*n/a*	*n/a*	3	3	3	*n/a*	*n/a*	*n/a*	3	3	l00	C-
fail	4	4	*n/a*	*n/a*	*n/a*	5	5	4	*n/a*	*n/a*	*n/a*	5	4	**2**	

p.i.	A1	A2	A3	A4	A5	A6	A7	A8	C1	C2	C3	C4	C5	#	
pass	42	40	0	0	0	0	42	41	*n/a*	*n/a*	*n/a*	42	40	**14**	**Gr**
9:345	3	3	*n/a*	*n/a*	*n/a*	*n/a*	3	3	*n/a*	*n/a*	*n/a*	3	3	b14	B
9:345	3	3	*n/a*	*n/a*	*n/a*	*n/a*	3	3	*n/a*	*n/a*	*n/a*	3	3	f68	B
9:345	4	2	*n/a*	*n/a*	*n/a*	*n/a*	4	4	*n/a*	*n/a*	*n/a*	4	4	w01	B
9:345	3	3	*n/a*	*n/a*	*n/a*	*n/a*	3	3	*n/a*	*n/a*	*n/a*	3	3	b47	B
9:345	3	3	*n/a*	*n/a*	*n/a*	*n/a*	3	3	*n/a*	*n/a*	*n/a*	3	3	m50	B
9:345	3	3	*n/a*	*n/a*	*n/a*	*n/a*	3	3	*n/a*	*n/a*	*n/a*	3	3	b69	B
9:345	3	3	*n/a*	*n/a*	*n/a*	*n/a*	3	3	*n/a*	*n/a*	*n/a*	3	3	f30	B
9:345	3	2	*n/a*	*n/a*	*n/a*	*n/a*	3	3	*n/a*	*n/a*	*n/a*	3	3	m15	B
9:345	3	3	*n/a*	*n/a*	*n/a*	*n/a*	3	4	*n/a*	*n/a*	*n/a*	3	3	z69	B
9:345	3	3	*n/a*	*n/a*	*n/a*	*n/a*	3	3	*n/a*	*n/a*	*n/a*	3	3	l38	B
9:345	0	0	*n/a*	*n/a*	*n/a*	*n/a*	0	0	*n/a*	*n/a*	*n/a*	0	0	f10	B
9:345	2	3	*n/a*	*n/a*	*n/a*	*n/a*	2	2	*n/a*	*n/a*	*n/a*	2	2	h64	B-
9:345	3	3	*n/a*	*n/a*	*n/a*	*n/a*	3	2	*n/a*	*n/a*	*n/a*	3	2	c74	B-
9:345	3	3	*n/a*	*n/a*	*n/a*	*n/a*	3	2	*n/a*	*n/a*	*n/a*	3	2	c85	C+
9:345	3	3	*n/a*	*n/a*	*n/a*	*n/a*	3	3	*n/a*	*n/a*	*n/a*	3	3	s47	C
9:345	0	0	*n/a*	*n/a*	*n/a*	*n/a*	0	0	*n/a*	*n/a*	*n/a*	0	0	l36	C
9:345	2	2	*n/a*	*n/a*	*n/a*	*n/a*	3	1	*n/a*	*n/a*	*n/a*	2	1	h55	C-
9:345	0	0	*n/a*	*n/a*	*n/a*	*n/a*	0	0	*n/a*	*n/a*	*n/a*	0	0	s85	C-
9:345	2	2	*n/a*	*n/a*	*n/a*	*n/a*	3	2	*n/a*	*n/a*	*n/a*	3	2	l00	C-
fail	4	4	*n/a*	*n/a*	*n/a*	*n/a*	6	3	*n/a*	*n/a*	*n/a*	5	3	**2**	

p.i.	A1	A2	A3	A4	A5	A6	A7	A8	C1	C2	C3	C4	C5	#	
pass	36	33	*n/a*	*n/a*	35	*n/a*	38	38	36	*n/a*	*n/a*	36	37	**14**	**Gr**
9:79	3	2	*n/a*	*n/a*	3	*n/a*	3	3	3	*n/a*	*n/a*	3	3	b14	B
9:79	3	3	*n/a*	*n/a*	3	*n/a*	3	3	3	*n/a*	*n/a*	3	3	f68	B
9:79	4	4	*n/a*	*n/a*	4	*n/a*	4	4	4	*n/a*	*n/a*	4	4	w01	B
9:79	3	3	*n/a*	*n/a*	3	*n/a*	3	3	3	*n/a*	*n/a*	3	3	b47	B
9:79	2	1	*n/a*	*n/a*	1	*n/a*	1	2	2	*n/a*	*n/a*	2	2	m50	B
9:79	2	3	*n/a*	*n/a*	3	*n/a*	3	3	3	*n/a*	*n/a*	3	3	b69	B
9:79	2	2	*n/a*	*n/a*	2	*n/a*	3	3	2	*n/a*	*n/a*	2	3	f30	B
9:79	2	3	*n/a*	*n/a*	3	*n/a*	3	3	3	*n/a*	*n/a*	2	3	m15	B
9:79	3	2	*n/a*	*n/a*	3	*n/a*	3	4	3	*n/a*	*n/a*	2	2	z69	B
9:79	3	3	*n/a*	*n/a*	3	*n/a*	3	3	3	*n/a*	*n/a*	3	3	l38	B
9:79	0	0	*n/a*	*n/a*	0	*n/a*	0	0	0	*n/a*	*n/a*	0	0	f10	B
9:79	2	2	*n/a*	*n/a*	3	*n/a*	2	1	3	*n/a*	*n/a*	2	2	h64	B-
9:79	2	2	*n/a*	*n/a*	1	*n/a*	2	1	1	*n/a*	*n/a*	2	1	c74	B-
9:79	2	1	*n/a*	*n/a*	1	*n/a*	2	2	1	*n/a*	*n/a*	2	2	c85	C+
9:79	3	2	*n/a*	*n/a*	2	*n/a*	3	3	2	*n/a*	*n/a*	3	3	s47	C
9:79	0	0	*n/a*	*n/a*	0	*n/a*	0	0	0	*n/a*	*n/a*	0	0	l36	C
9:79	2	2	*n/a*	*n/a*	2	*n/a*	3	2	2	*n/a*	*n/a*	2	2	h55	C-
9:79	0	0	*n/a*	*n/a*	0	*n/a*	0	0	0	*n/a*	*n/a*	0	0	s85	C-
9:79	2	3	*n/a*	*n/a*	2	*n/a*	3	2	2	*n/a*	*n/a*	3	2	l00	C-
fail	4	5	*n/a*	*n/a*	4	*n/a*	6	4	4	*n/a*	*n/a*	5	4	**2**	

p.i.	A1	A2	A3	A4	A5	A6	A7	A8	C1	C2	C3	C4	C5	#	
pass	35	27	*n/a*	*n/a*	35	30	36	*n/a*	*n/a*	*n/a*	*n/a*	34	33	**14**	**Gr**
9:8	3	2	*n/a*	*n/a*	3	2	3	*n/a*	3	*n/a*	*n/a*	3	3	b14	B
9:8	3	2	*n/a*	*n/a*	3	3	1	*n/a*	3	*n/a*	*n/a*	2	2	f68	B
9:8	4	4	*n/a*	*n/a*	4	4	4	*n/a*	4	*n/a*	*n/a*	4	4	w01	B
9:8	3	2	*n/a*	*n/a*	3	2	3	*n/a*	3	*n/a*	*n/a*	3	3	b47	B
9:8	0	0	*n/a*	*n/a*	0	0	0	*n/a*	0	*n/a*	*n/a*	0	0	m50	B
9:8	3	3	*n/a*	*n/a*	3	3	3	*n/a*	3	*n/a*	*n/a*	3	3	b69	B
9:8	2	2	*n/a*	*n/a*	3	2	3	*n/a*	2	*n/a*	*n/a*	2	2	f30	B
9:8	3	2	*n/a*	*n/a*	3	3	3	*n/a*	3	*n/a*	*n/a*	3	3	m15	B
9:8	2	2	*n/a*	*n/a*	2	3	3	*n/a*	3	*n/a*	*n/a*	3	2	z69	B
9:8	2	1	*n/a*	*n/a*	1	1	3	*n/a*	2	*n/a*	*n/a*	1	1	l38	B
9:8	0	0	*n/a*	*n/a*	0	0	0	*n/a*	0	*n/a*	*n/a*	0	0	f10	B
9:8	3	2	*n/a*	*n/a*	3	2	3	*n/a*	3	*n/a*	*n/a*	3	3	h64	B-
9:8	2	2	*n/a*	*n/a*	3	2	2	*n/a*	2	*n/a*	*n/a*	2	2	c74	B-
9:8	2	1	*n/a*	*n/a*	1	1	2	*n/a*	1	*n/a*	*n/a*	2	2	c85	C+
9:8	3	2	*n/a*	*n/a*	3	2	3	*n/a*	3	*n/a*	*n/a*	3	3	s47	C
9:8	0	0	*n/a*	*n/a*	0	0	0	*n/a*	0	*n/a*	*n/a*	0	0	l36	C
9:8	3	2	*n/a*	*n/a*	3	2	3	*n/a*	3	*n/a*	*n/a*	3	2	h55	C-
9:8	0	0	*n/a*	*n/a*	0	0	0	*n/a*	0	*n/a*	*n/a*	0	0	s85	C-
9:8	2	3	*n/a*	*n/a*	3	3	3	*n/a*	3	*n/a*	*n/a*	3	3	l00	C-
fail	5	5	*n/a*	*n/a*	6	5	6	*n/a*	6	*n/a*	*n/a*	6	5	**2**	

We emphasized several times in class that HW9 carried a substantial impact on the course grade: at ¼ of the total HW points with a weight of 50%, it contributed 12.5% of the total course score, or roughly 2/3 of a letter grade. Student participation was relatively higher, with valid submissions from 14 of 16 in the "pass" tier and 2 of 3 in the "fail". (Of the remaining 3 students, 1 ignored HW9 only, and the other 2 ignored all HWs for the entire semester, which was a borderline failing strategy.)

The performance histogram for HW9 is as follows. First, we group by HW9 portions:

tier	p.i.	A1	A2	A3	A4	A5	A6	A7	A8	C1	C2	C3	C4	C5	#
pass	**9:126**	37	35	*n/a*	*n/a*	*n/a*	42	41	40	*n/a*	*n/a*	*n/a*	42	38	**14**
fail	**9:126**	4	4	*n/a*	*n/a*	*n/a*	5	5	4	*n/a*	*n/a*	*n/a*	5	4	**2**
pass	**9:345**	42	40	*n/a*	*n/a*	*n/a*	*n/a*	42	41	*n/a*	*n/a*	*n/a*	42	40	**14**
fail	**9:345**	4	4	*n/a*	*n/a*	*n/a*	*n/a*	6	3	*n/a*	*n/a*	*n/a*	5	3	**2**
pass	**9:79**	36	33	*n/a*	*n/a*	35	*n/a*	38	38	36	*n/a*	*n/a*	36	37	**14**
fail	**9:79**	4	5	*n/a*	*n/a*	4	*n/a*	6	4	4	*n/a*	*n/a*	5	4	**2**
pass	**9:8**	35	27	*n/a*	*n/a*	35	30	36	*n/a*	35	*n/a*	*n/a*	34	33	**14**
fail	**9:8**	5	5	*n/a*	*n/a*	6	5	6	*n/a*	6	*n/a*	*n/a*	6	5	**2**

Next, we show the same table, but grouped by tier.

tier	p.i.	A1	A2	A3	A4	A5	A6	A7	A8	C1	C2	C3	C4	C5	#
pass	**9:126**	37	35	*n/a*	*n/a*	*n/a*	42	41	40	*n/a*	*n/a*	*n/a*	42	38	**14**
	9:345	42	40	*n/a*	*n/a*	*n/a*	*n/a*	42	41	*n/a*	*n/a*	*n/a*	42	40	
	9:79	36	33	*n/a*	*n/a*	35	*n/a*	38	38	36	*n/a*	*n/a*	36	37	
	9:8	35	27	*n/a*	*n/a*	35	30	36	*n/a*	35	*n/a*	*n/a*	34	33	
fail	**9:126**	4	4	*n/a*	*n/a*	*n/a*	5	5	4	*n/a*	*n/a*	*n/a*	5	4	**2**
	9:345	4	4	*n/a*	*n/a*	*n/a*	*n/a*	6	3	*n/a*	*n/a*	*n/a*	5	3	
	9:79	4	5	*n/a*	*n/a*	4	*n/a*	6	4	4	*n/a*	*n/a*	5	4	
	9:8	5	5	*n/a*	*n/a*	6	5	6	*n/a*	6	*n/a*	*n/a*	6	5	

Roughly, we infer that the "pass" tier achieves average performance scores in the half-closed interval (2.0, 3.0] over all relevant performance indicators, except for:

- A2 *Demonstrate understanding of computer hardware architecture.* (27 / 14 = barely under 2.0) For HW9 portion 9:8, the interrupt handler, this item pertains to knowledge of the MIPS interrupt/exception coprocessor 0, and how to extract the needed information from the Status and Cause registers on that coprocessor. The teaching method relies on using example code given in the textbook. We infer that we may need to add emphasis to this topic, and/or provide more example code not in the textbook.

In contrast, the "fail" tier averages ≤ 2.0 points for about half of the performance indicators. In particular:

- A1 *Design an algorithm and data structure.* Weakness here generally indicates poor programming ability, which tends to manifest throughout all other programming assignments, and the programming questions on the Midterm and Final Exams.
- A2 *Demonstrate understanding of computer hardware architecture at the instruction level.* For 9:126, this denotes a failure to adhere to the strict MIPS MMIO interface, or to learn it adequately from lectures and the textbook. For 9:345, it indicates a deficiency in performing simple array read/write operations.
- A5 *Implement … using multiple paradigms.* For portion 9:79, it covers the cognitive task of properly treating a main loop disjointly from an interrupt handler, using a shared global variable as a data bridge between them. This is similar to programming with two threads, in that the student must broaden the scope of his or her thinking to a similar extent. See also Final Exam F6.

- A8 *Develop a test plan;* C5 *Validate a solution.* Students in the "fail" tier tend to be poor in testing their code, or in devising an effective testing methodology. More deeply, the 2 students in this tier made far less than a reasonable effort in development, so they simply didn't have sufficient code that was ready to be tested. This indicates a deeper inability in programming.

3. Deliverable: Final Exam

This is a closed-book in-class exam, to be completed in one sitting, in two hours total. There are seven problems:

F1	short answers
F2	MIPS bitfields and datapath highlighting
F3	cache performance and direct-mapped cache
F4	MIPS assembly programming: pointers
F5	MIPS assembly programming: recursion
F6	MIPS assembly programming: event handler + main loop
F7	numeric representation: two's complement, float

These problems cover the performance indicators shown in yellow.

p.i.	A1	A2	A3	A4	A5	A6	A7	A8	C1	C2	C3	C4	C5
F1	*n/a*		*n/a*	*n/a*	*n/a*	*n/a*	*n/a*	*n/a*	*n/a*	*n/a*	*n/a*	*n/a*	*n/a*
F2	*n/a*		*n/a*	*n/a*	*n/a*	*n/a*	*n/a*	*n/a*	*n/a*	*n/a*	*n/a*		*n/a*
F3	*n/a*			*n/a*	*n/a*	*n/a*	*n/a*	*n/a*	*n/a*	*n/a*	*n/a*	*n/a*	*n/a*
F4				*n/a*	*n/a*	*n/a*	*n/a*	*n/a*	*n/a*	*n/a*	*n/a*		*n/a*
F5			*n/a*	*n/a*	*n/a*	*n/a*	*n/a*	*n/a*		*n/a*	*n/a*		*n/a*
F6			*n/a*	*n/a*		*n/a*	*n/a*	*n/a*		*n/a*	*n/a*		*n/a*
F7	*n/a*			*n/a*	*n/a*	*n/a*	*n/a*	*n/a*	*n/a*	*n/a*	*n/a*	*n/a*	*n/a*

Performance indicators not covered (A4, A6–A8, C2, C3, C5) are mostly process-based (design, implementation, debugging), which are not relevant to a finite-time written exam. For conciseness, we elide all irrelevant columns, and group relevant columns by exam problem.

	F1	F2		F3		F4				F5				F6					F7			
	A2	A2	C4	A2	A3	A1	A2	A3	C4	A1	A2	C1	C4	A1	A2	A5	C1	C4	A2	A3		
	39	45	45	37	42	41	39	45	37	53	50	53	50	43	45	39	43	41	31	42	**16**	**Gr**
pass	2	3	3	3	4	3	3	3	3	4	4	4	4	2	2	2	2	3	2	3	b14	B
	2	3	3	2	2	3	2	3	3	4	4	4	4	2	3	2	3	3	2	2	f68	B
	2	2	3	2	1	4	4	4	3	4	3	4	3	4	3	4	4	3	2	3	w01	B
	3	3	3	3	4	2	2	3	2	3	3	3	3	4	4	4	4	4	2	3	b47	B
	3	4	3	3	3	4	4	4	4	4	4	4	4	4	4	4	4	3	2	2	m50	B
	2	3	3	2	2	3	2	2	2	3	2	3	3	1	2	1	1	1	1	3	b69	B
	2	3	3	2	3	2	3	2	1	3	2	3	3	1	2	1	1	1	2	3	f30	B
	3	1	2	2	2	1	2	2	1	4	4	4	3	4	4	4	4	4	3	3	m15	B
	2	2	3	1	3	1	2	2	2	3	3	3	3	2	2	3	3	3	2	2	z69	B
	3	4	3	2	2	3	2	3	3	2	3	3	3	3	3	3	3	2	2	3	l38	B
	3	3	3	3	4	3	3	3	2	3	3	3	3	4	4	4	4	4	1	2	f10	B
	2	3	3	2	2	2	2	3	1	3	3	2	3	2	3	1	1	2	1	2	h64	B-
	3	2	2	2	1	2	2	3	3	3	3	3	3	3	2	3	3	2	2	2	c74	B-
	2	2	2	3	3	3	2	3	2	3	2	3	3	2	2	1	3	2	3	3	c85	C+
	2	3	2	3	4	3	2	2	3	4	4	4	3	3	2	1	2	2	1	3	s47	C
	3	4	4	2	2	2	2	3	2	3	3	3	2	2	3	1	1	2	3	3	l36	C
fail	3	4	4	2	1	2	2	2	2	2	2	2	1	1	2	1	1	1	1	2	h55	C-
	3	4	3	2	2	1	1	1	1	4	4	4	4	3	2	3	3	3	2	2	s85	C-
	1	1	2	2	3	1	1	1	1	1	1	1	1	1	2	1	1	2	1	2	l00	C-
	7	9	9	6	6	4	4	4	4	7	7	7	6	5	6	5	5	6	4	6	**3**	

The histogram for the Final Exam consists of the first and last rows in the above table. The "pass" tier averaged > 2.0-point performance for all problems and all performance indicators, except one.

- Problem F7 (numeric representations), A2. *Demonstrate understanding of computer hardware architecture.* It shows that the students' grasp of IEEE 754 floating-point representation is borderline weak (just below 2.0 "needs improvement"). We conclude that the classroom instruction and homework assignments should emphasize this topic more.
- The "pass" tier showed a > 3.0-point performance on problem F5, which is a programming problem to write a Fibonacci-like doubly-recursive function. We conclude that our strong emphasis on teaching recursive functions, with many examples, was generally successful in imprinting this technique across most students.

The “fail” tier averaged ≤ 2.0-point performance on about half of the problems. These were the 3 MIPS programming problems, suggesting a general deficiency in ability to program.

- F4 is a MIPS programming problem to write a function that searches and updates an array of characters using (registers that behave as) “pointers-to-char”. Students who are weak in thinking abstractly with pointers tend to slide down to the “fail” tier, or the bottom half of the “pass” tier.
- F5 is as described above. Even in the “fail” tier, we see relatively better performance for this problem, which suggests that they too benefitted from the class emphasis here.
- F6 is a MIPS programming problem that has two disjoint portions, a main loop and an “event handler” function, which share the same global variable. It is a greatly simplified version of HW9’s interrupt handler that eliminates the interrupt handler’s details. It demands a similar cognitive ability to mentally separate the two code portions and envision how they can each achieve shared work while never directly calling each other. Students who did poorly on this concept *and* at least 1 other major area (either another Final Exam problem, or the HWs) tended to fall to or below the “fail” boundary.
- F7 is as above. Both “pass” and “fail” tiers showed similar weakness here, which suggests that the classroom instruction should be improved.

Summary

Overall, considering all 3 course deliverables, we can summarize the results thus:

- Students who average > 2.0-point performance on all 3 deliverables are in the “pass” tier.
    - If they are barely above 2.0, they are in the C/C+ grade range.
    - If they are substantially above 2.0 (and near 3.0) performance, they move up into the B grade range.
- Students who average ≤ 2.0-point performance on all 3 deliverables are in the “fail” tier.
- Conversely, students in the “fail” tier either averaged ≤ 2.0-point performance on all 3 deliverables, or ignored 2 of the 3 deliverables.

Appendix B:

Carl Hauser's Assessment Report for CptS 355

Introduction

I was asked to do outcomes assessment for the Spring 2014 offering of CptS 355 – Design of Programming Languages because the instructor has moved on from WSU. CptS 355 is a required course in the BSCptS and BACptS programs. I have taught CptS 355 in Fall semesters for many years. This was the first Spring offering in a long time and the adjunct faculty member who taught the course generally used the curriculum and assignments I have used, so I am familiar with these assignments and know what I would expect in them.

Methodology

The two assignment selected for outcomes evaluation are a simplified-postscript interpreter, which is the fourth programming assignment in the course and is designated H4 below, and the Scheme take-home quiz, designated SQ below. The scheme quiz comes about 1/3 of the way through the course and H4 is the next-to-last assignment of the semester. H4 is based on an earlier assignment and requires the students to modify their earlier code (or alternatively, code supplied by the instructor) to support a form of static scoping for postscript variables. The Scheme Quiz addresses basic recursion patterns for lists as well as definition and use of higher-order functions.

For CptS 355 the identified outcomes for assessment this year were (along with the indicators that the program has identified):

- B: Ability to analyze a problem and identify and define the computing requirements appropriate to its solution. Indicators:
    - B1. Identifies technical requirements and specifications for a design project.
    - B2. Constructs a problem statement that articulates what constitutes a solution.
    - B3. Identifies measurable parameters that characterize a problem and its solution.
    - Outcome B is not relevant for SQ; for H4 there is a weak connection to indicators B2 and B3 and stronger connection to the overall outcome "analyze a problem and identify and define computing requirements appropriate to its solution"

- C: Ability to design, implement and evaluate a computer-based system, process, component or program to meet desired needs. Indicators:
    - C1. Recognizes when theory dictates that a solution is impractical or impossible.
    - C2. Generates multiple design concepts.
    - C3. Determines and articulates tradeoffs among design alternatives and selects an approach or, as appropriate, approaches to solve the problem.
    - C4. Implements the selected approach, or approaches, to obtain a solution.
    - C5. Validates a solution, identifying reasons for differences between expected and actual behavior.
    - Outcome C is relevant for both SQ and H4. For SQ, the indicators C3 and C5 are implicit

and C4 is the main one (is it a solution and do you know it is a solution). For H4, all of C3, C4, and C5 are relevant.

- I: Ability to use current techniques, skills, and tools necessary for computing practice. Indicators:
  - I1. Uses current programming languages, operating systems, and software libraries to implement computing solutions.
  - I2. Uses current tools to create documents and presentations.
  - I3. Uses current tools to manage data and processes related to development of computing solutions.
  - I4. Evaluates which techniques or tools are most appropriate to complete a specific computing task.
  - For both SQ and H4, the relevant indicators are I1 and I4.

Assignments are rated, for each outcome on the scale:

1. Unsatisfactory
2. Needs Improvement
3. Competent
4. Exemplary

My method was to take into account the identified indicators when coming up with the overall rating for a given assignment and a given outcome.

In the final grades for the course there are 1 C+, 10 Cs, one D+ and one D. The student work provided by the instructor includes at least one of H4 and SQ for 7 of the students receiving Cs and the student receiving a C+. There are no assignments for the other 3 students who received Cs or the students receiving a D or D+. I do not know why there are so many missing assignments – it may be that they were not submitted by the instructor for some reason, or it seems likely that at least some of the students never turned them in. The fact that among students receiving a B grade or better, the fraction for which both assignments are available to be evaluated tends to support this hypothesis. (Note that in the offerings of this course that I teach students are required to get a minimum score on the programming projects in order to receive a C or higher grade – missing assignments along with low performance on other assignments make it difficult to pass under these rules.)

In addition to evaluating the work related to these low grades (C or above is considered sufficient to meet the program's requirements for successful completion of this course) I also sampled the work of students receiving B grades – about 1/3 of those, select by taking every 3rd student alphabetically. Those results are tabulated separately below.

Observations of student work

Students are identified as Si and my remarks about their work is indented below in the format Assignment-Outcome: Rating – remarks.

Remarks and ratings for students receiving course grade C or C+

S1

H4-B: 3 -- uses language features such as lists and dicts to implement necessary part of the interpreter.

H4-C: 2 -- has not fully met the specifications for the interpreter; has incorrect behavior on some inputs

H4-I: 3 -- uses language features correctly but not well in all cases

S2

SQ-C: 2 -- had the general right idea for the simpler problems; did not use correct scheme syntax a lot of the time. Was not able to do the last two problems (merge2 and mergeN)

SQ-I: 1 -- not sufficiently able to use simple recursive techniques to solve even slightly complicated problems.

S3

SQ-C: 1 -- no idea what to do for any of the problems

SQ-I: 1 -- threw something on the page

S4

H4-B: 2 -- uses language features such as lists and dicts to implement necessary part of the interpreter.

H4-C: 1 -- has not fully met the specifications for the interpreter; does not understand the correct behavior of static scoping

H4-I: 3 -- uses language features correctly but not well in all cases

SQ-C: 4 -- correct solutions to problems using both basic recursion and higher-order functions

SQ-I: 4 -- defined his own aux function that could have been assumed to exist to get full credit

S5

H4-B: 1 -- has thrown together bits of language without understanding what they do

H4-C: 1 -- the program does not even meet the requirement to handle a command-line switch and do different things based on its value

H4-I: 2 -- doesn't look like he knows what many things do above the level of basic syntax of python

SQ-C: 3 -- handled the basic recursion problems with some difficulty. Not the higher-order problems

SQ-I: 3 -- did basic recursion but not well -- less-than-optimal base case choices, etc.

S6

H4-B: 3 -- uses language features such as lists and dicts to implement necessary part of the interpreter.

H4-C: 2 -- has not fully met the specifications for the interpreter; has incorrect behavior on some inputs

H4-I: 3 -- uses language features correctly but not well in all cases

SQ-C: 4 -- solutions are all correct

SQ-I: 4 -- language is used appropriately

S7 (C+)

H4-B: 3 -- uses language features such as lists and dicts to implement necessary part of the interpreter.

H4-C: 2 -- has not fully met the specifications for the interpreter; has incorrect behavior on some inputs; does not fully understand what the problem is or what constitutes a solution.

H4-I: 3 -- uses language features correctly but not well in all cases

SQ-C: 2 -- did basic recursion ok but no answers for the solutions involving HOFs

SQ-I: 2 -- does not have a good sense of good usage verus correct but bad usage (cons x y) vs (append (list x) y)

S8

SQ-C: 2 -- has a rough idea of the requirements for the simple recursive functions; did not understand merge2 problem and understood the repl problem as update-in-place rather than compute a new value

SQ-I: 2 -- does not use the language in the way that was taught in class; instead uses untaught features to accomplish bizarre results

Remarks and ratings for students receiving course grade B

There were no B- grades. Here are the ratings and remarks for a sample of Bs (every 3rd one alphabetically);

S9

SQ-C: 3 -- well done but didn't tackle the last problem because claimed it was not covered in class (1st to mention this)

SQ-I: 4

S10

H4-B: 2 -- replicates 100s of lines of code to handle a few lines of difference. Inappropriate use of language.

H4-C: 3 -- appears to understand what is required to meet the specifications of the problem

H4-I: 3 -- uses language features correctly but not well in all cases

SQ-C: 3 -- correct idea for all the problems attempted; did not do mergeN and didn't see how to use HOF to solve it; the first to

recognize that the conditions given in the problem eliminated the need for some error checks and to take advantage of that.

SQ-I: 3 -- couldn't get parens right for the merge2 function but had the right idea.

S11

H4-B: 3 -- uses language features such as lists and dicts to implement necessary part of the interpreter.

H4-C: 2 -- does not fully understand the problem or what constitutes a solution

H4-I: 3 -- uses language features correctly but not well in all cases

SQ-C: 2 -- code shows signs of beginning to understand but has not actually run the code and seen that there are mistakes galore

SQ-I: 1 -- uses library functions without understanding what they do (for example map)

S12

H4-B: 3 -- uses language features such as lists and dicts to implement necessary part of the interpreter.

H4-C: 2 -- does not fully understand the problem or what constitutes a solution

H4-I: 3 -- uses language features correctly but not well in all cases

SQ-C: 4 -- correct solutions and correct language use

SQ-I: 4

S13

H4-B: 3 -- uses language features such as lists and dicts to implement necessary part of the interpreter.

H4-C: 3 -- appears to understand the static scoping definition and its implementation

H4-I: 3 -- uses language features correctly but not well in all cases

SQ-C: 2 -- basic understanding of the recursive approach and HOFs but has not been tested. Uses update in place instead of functional approach for repl.

SQ-I: 2 -- syntax errors and generally missing attention to detail in the code

Tabulated Results

Outcome	B	C	I	
C Grades				
S1-H4	3	2	3	
S2-SQ		2	1	
S3-SQ		1	1	
S4-H4	2	1	3	
S4-SQ		4	4	
S5-H4	1	1	2	
S5-SQ		3	3	
S6-H4	3	2	3	
S6-SQ		4	4	
S7-H4	3	2	3	C+
S7-SQ		2	2	C+
S8-SQ		2	2	
There were 3 more students who received C grades and one each who received a D and D+ for whom there was no data on these assignments, etiher because the instructor didn't deliver it or they didn't turn it in. I can't tell from this distance which holds.				
B Grades				
S9-SQ		3	4	
S10-H4	2	3	3	
S10-SQ		3	3	
S11-H4	3	2	3	
S11-SQ		2	1	
S12-H4	3	2	3	
S12-SQ		4	4	
S13-H4	3	3	3	
S13-SQ		2	2	

Conclusion

For students receiving C or C+ grades in the course, it appears that based on these two assignments the program's goal of having students exhibit traits associated with the outcomes at the "Competent" level were not met in this offering of CptS 355. Of course, there are many more opportunities for students to demonstrate the capabilities but I would read this as suggesting that at least some students are at best borderline. My interpretation of the data for the students receiving B grades is that while on these two assignments they don't always exhibit competency, most of the time they do. Since it is natural that while students are still learning a subject they will progress toward competency at different rates it does not surprise or bother me that students receiving B grades at this stage of their education are missing the mark sometimes.

Appendix C:

Sakire Arslan-Ay's Assessment Report for CptS 423 Team Gondor

CptS421 -Assignment 1: Project Description and Clarification (B2)
B2: 3.5 (Section II summarizes project goals and expected outcomes. Description is brief but clear.)

CptS421 -Assignment 2: Project Requirements Specifications (B1)
B1: 3 (Sections II.1 and II.3 identifies the functional and non-functional requirements of the software. Descriptions are brief and don't provide adequate detail. Some requirements are omitted.)

CptS421 -Assignment 3: Solution Approach (I1)
I1: 3 (The last paragraph in Section I identifies the programming language and software libraries used for the development of the software.)

CptS421 -Assignment 4: Alpha Prototype Description (C1,C4)
C1: 3.5 (In Section II (under subtitles "Preliminary Results") various design decisions have been analyzed based on preliminary tests. For example Section II.4 discusses that the measurements obtained through gestures are not practical for CAD processing, therefore need to be scaled.)
C4: 3 (Section III describes the demonstration of the alpha prototype to the project mentor and instructor.)

CptS423 -Assignment 3: Final Report (C1, C2, C3, C4, C5, I2, I3, I4)
C1: 3 (First paragraph of Section IV.1.1) (Invoking principles of cohesion and coupling in Section IV.1.2)
C2: N/A to this group's project
C3: 3 (First paragraph of Section IV.1.1) (Invoking principles of cohesion and coupling in Section IV.1.2)
C4: 2.5 (Section 5 describes prototype; Section 6 articulates test cases.)
C5: 3 (Section 6 describes test cases. Instructor reports that student team identified reasons for failed tests and corrected the software so that they passed for the final report.)
I2: 3 (The writing assignments have been created using MS Word and MS Power Point. Various diagrams and charts in the assignments have been created using MS Visio and similar tools)
I3: 4 (The senior design team used the Socialcast platform for team collaboration, discussions and information sharing. https://eecs-wsu-edu.socialcast.com
The senior design team has maintained the project code on the EECS Github server for code reviews, and code management. https://github.eecs.wsu.edu/)
I4: N/A to this group's project

Appendix C:

Sakire Arslan-Ay's Assessment Report for CptS 423 Team Red Dragon

CptS421 -Assignment 1: Project Description and Clarification (B2,I4)
B2: 3.5 (Section II summarizes project goals and expected outcomes.)
I4: 3.5 (Section I evaluates alternative frameworks and platforms for the development of the software.)

CptS421 -Assignment 2: Project Requirements Specifications (B1)
B1: 3 (Sections II.1 and II.3 identify the functional and non-functional requirements of the software. All major requirements are listed, however descriptions are very brief and don't provide adequate detail.)

CptS421 -Assignment 3: Solution Approach (C1,C2,C3)
C1: 3.5 (Section III last paragraph explains why relational model would not work to store the index data structure)
C2: 4 (Section II.1 discusses the software design patterns that would suit the best for the software. Section II.2 explains the design approaches for each software subsystem. Section III discusses two different approaches for the index data structure.)
C3: 4 (In Section II the tradeoffs between different design choices are discussed. For example in Section II.2.2.2 the advantages and disadvantages for different data storage solutions are explained and the incentive for the selected solution is given.)

CptS421 -Assignment 4: Alpha Prototype Description (C3,C4)
C3: 4 (Section II discusses various changes made to the initial design and why those changes were made. For example in Section II.1.2, the performance optimizations on the A* search algorithm is explained.)
C4: 3.5 (Section III describes the demonstration of the alpha prototype to the project mentor and instructor.)

CptS423 -Assignment 3: Final Report (C4, C5, I2, I3)
C4: 3.5 (Section V describes the final prototype; Section VI articulates successful and failed test cases.)
C5: 3.5 (Section 6 describes test cases. The failed tests were not repeated.)
I2: 3 (The writing assignments have been created using MS Word and MS Power Point. Various diagrams and charts in the assignments have been created using MS Visio and similar tools)
I3: 4 (The senior design team used the Socialcast platform for team collaboration, discussions and information sharing. https://eecs-wsu-edu.socialcast.com
The senior design team has maintained the project code on the EECS Github server for code reviews, and code management. https://github.eecs.wsu.edu/)

Appendix D:

CptS 423 Team Gondor's Final Report

Development of an Interactive CAD modeling system using RGB-D Type Sensor

Final Report

School of Mechanical and Materials Engineering

Mentor(s):
Dr. Gaurav Ameta

Team Gondor
Michael Belay
Young Chon
Lyle Dallas
Brian Fleck
Karan Sharma

CptS 423 Software Design Project II

Spring 2014

Instructor: Sakire Arslan Ay

TABLE OF CONTENTS

I. Executive Summary

The state of our project is near complete. We've completed and tested the major functionalities that align with our requirements thus far. The only thing left is to tweak the program and make sure it works flawlessly each time.

Our GUI was the biggest improvement from when we did our test cases. Before, we just had the Skeleton Viewer with a list of shapes that the user created in that session. Now, we've implemented an OpenGL world that tracks your head rotates the OpenGL camera (so the original coordinates of the shape would be preserved). Our goal was to eradicate keyword conflicts so we have our software show the command the user said. If the command is wrong, the user can say "cancel" and/or "go back" to start over. The "Shape Selection" portion integrated right in the main window so the User can select whichever shape they'd like to edit/manipulate. We've managed to create our shapes in OpenGL so we could bypass the OpenSCAD window and have OpenGL show what shapes are being created/used in real-time. With OpenSCAD, the file being modified would need to be saved at regular intervals and then recompiled to show the differences from the previous edit, to the new edit.

The data translation is working flawlessly. 3D and 2D shapes are now implemented and the program can render and display both. The shapes that are stored in our object manager are managed with the original scale and condition the user creates it. From there, the user is free to edit/manipulate the shape however they want.

II. Introduction

Our project implements an interactive 3D geometric modeling system that utilizes the Microsoft Kinect sensor for getting user input through gestures and audio commands. The system gathers the user's joint data and compares the positions of the joints relative to the Kinect. This data is then encapsulated and translated into raw CAD object parameters, which is then converted into a polygon point data for OpenGL rendering on the 3D interface. The CAD object data are also translated into OpenSCAD commands to allow viewing and editing of the models in OpenSCAD window. Our software can be used by anyone to design basic CAD models. This will take away from the complexity of traditional CAD tools and thus increase overall usability.

The main goal of this project will be to make simple gestures that map to commands in the CAD program. The basic commands that we would need to map to be grouped into two groups; environment movement and object manipulation. Environment movement will consist of rotate, pan, zoom, which are the basic movements in most CAD programs. For object manipulation we would extruding, scaling, cutting, transformations, and moving. These different commands sometimes have different ways that they function, such as the extrude command having either set or no value for incrementing or decrementing a measurement.

III.System Requirements Specification

III.1 Stakeholders

Stakeholders for the system range between students in academia and engineers in industry. Many engineers may find that the traditional mouse and keyboard inputs seem to restrict CAD users to design objects in a certain way. The functional requirement change traditional input into a more natural way to describe and design objects and provide a new way for our stakeholders to use CAD software.

LII.2 Functional requirements

The Kinect provides a crucial hardware component that our system is dependent on. Without being able to control the RGB-D sensor, the system would not be map out gestures, along with many other components provide with the Kinect. Using an application programming interface (API) provided with the Kinect software Development kit (Kinect SDK), allows the software development team the ability to gather data from the Kinect. A clear understanding of all the capabilities that the Kinect provides along with its libraries provided by the Kinect SDK defines the functional correctly since it relays information to other components gathered from the user.

IV. 1 Software Design

IV.1.1 Overview

Our interactive CAD modeling software uses a closed multi-layered system architecture. Due to the complex nature of the software interacting with many components, it seems reasonable to break the software into subsystems that allows the development team to handle tackle the project. The system is closed, thus establishing that communication between layers are restricted and limited to only the layers below and above the component. There are three major layers in the system that allows for cohesion and low coupling, as seen in Figure 1.

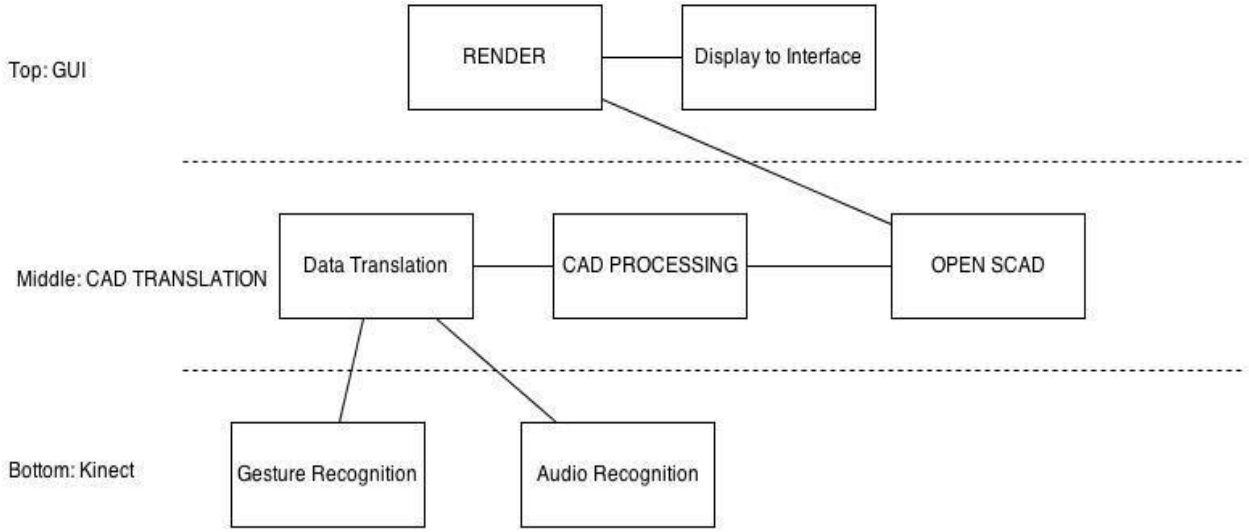


Figure 1

The top layer is mostly for GUI control and visual feedback for the user. The components will render information given from OpenSCAD to display on the user interface. This layer is at the top, since it relies on information provided by the layer below it to supply the users with visual feedback. The middle layer is what encapsulate and converts data collected into CAD objects. Taking information from the layer below it, the middle layer takes this data and translate and transport it to openSCAD for processing. The information produced by openSCAD will be sent to the top layer for display. The bottom layer is strictly for gathering user input from a Microsoft Kinect. This data is encapsulated and then transported up to the middle layer.

IV.1.2 Subsystem Decomposition

Our program is comprised of six subsystems in total. These subsystems are the Audio and Gesture Recognition, Data Translation, CAD processing, Render, GUI, and the OpenSCAD. This decomposition for the OpenSCAD and Kinect were done because they already existed outside of our program as third party tools. The Audio and Gesture subsystem is made because we need something to interact with the Kinect in such a way that we can pull raw data to be used later in the program. The Data Translation translates the raw data from the Kinect into something that is given to the OpenSCAD program. The CAD processing handles the interactions that goes on with the OpenSCAD program by taking the data we give it and passing it to the OpenSCAD. The Render subsystem is used to render items that are currently made in the program with information passed from the CAD processing. The GUI subsystem displays the user the rendered items that have been made, and a skeleton structure to help them see where in the workspace they are when manipulating items.

The Gesture and Audio have a high cohesion, as its process all revolve around using the Kinect to gather input for our program. The subsystem only dealt with the input from the Kinect and nothing else, and once it has that input will pass it on to other subsystems that require the information. The Data Translation subsystem has high cohesion as it only generates input for the CAD program. The CAD program has a high cohesion, as it is tasked with generating input for only the Openscad subsystem. The render only work on rendering the objects that are passed to it from the Openscad, this mean the subsystem has high cohesion. The GUI takes information from the Render subsystem and displays the created objects to the user, and also provide feedback to the user for errors made, objects being created, and their current coordinate location. This means it has a high cohesion as well, as its processes all focus on providing information to the user.

The coupling of most of the subsystems is low for most of them. The Gesture and Audio went on to interact with the Data Translation subsystem by sending it data from the Kinect, which in turn organize the data in a way that can be used by the CAD processing subsystem. The Gesture and Audio subsystem, as well as the CAD processing will have low coupling due to this, but the Data Translation has slightly higher coupling than the others. This is because it interacts with both the Gesture and Audio subsystems. The Render will have a low coupling as well as it only interacts with the GUI outside of the CAD Processing. The GUI will have low coupling since it only receives information from the Render subsystem.

1. Audio Recognizer

The Audio recognizer gets the raw data from the Kinect and process it and convert it to text file.

a. Interface Description

The interface gets audio as input from the user and outputs a text file.

Services Provided:

1.) Gathering audio
Service provided to: [Audio Recognizer subsystem]
Description: [It captures the audio from user .The input is audio and the output is a text file.

1. Gesture Recognition

The gesture recognizer gets the raw data from the Kinect and process it and convert it to text file.

a) Interface Description
The interface gets gesture as input from the user and outputs a text file.

Services Provided:
1.) Gathering gesture
Service provided to: Gesture Recognizer
Description: It captures the gesture from user. The input is gesture and the output is a text file.

1. Data Translation

(I)This subsystem will take the data provided by the Gesture and Audio subsystems and put it into a data structure that will allow the CAD processing to find the needed information it needs to execute commands.

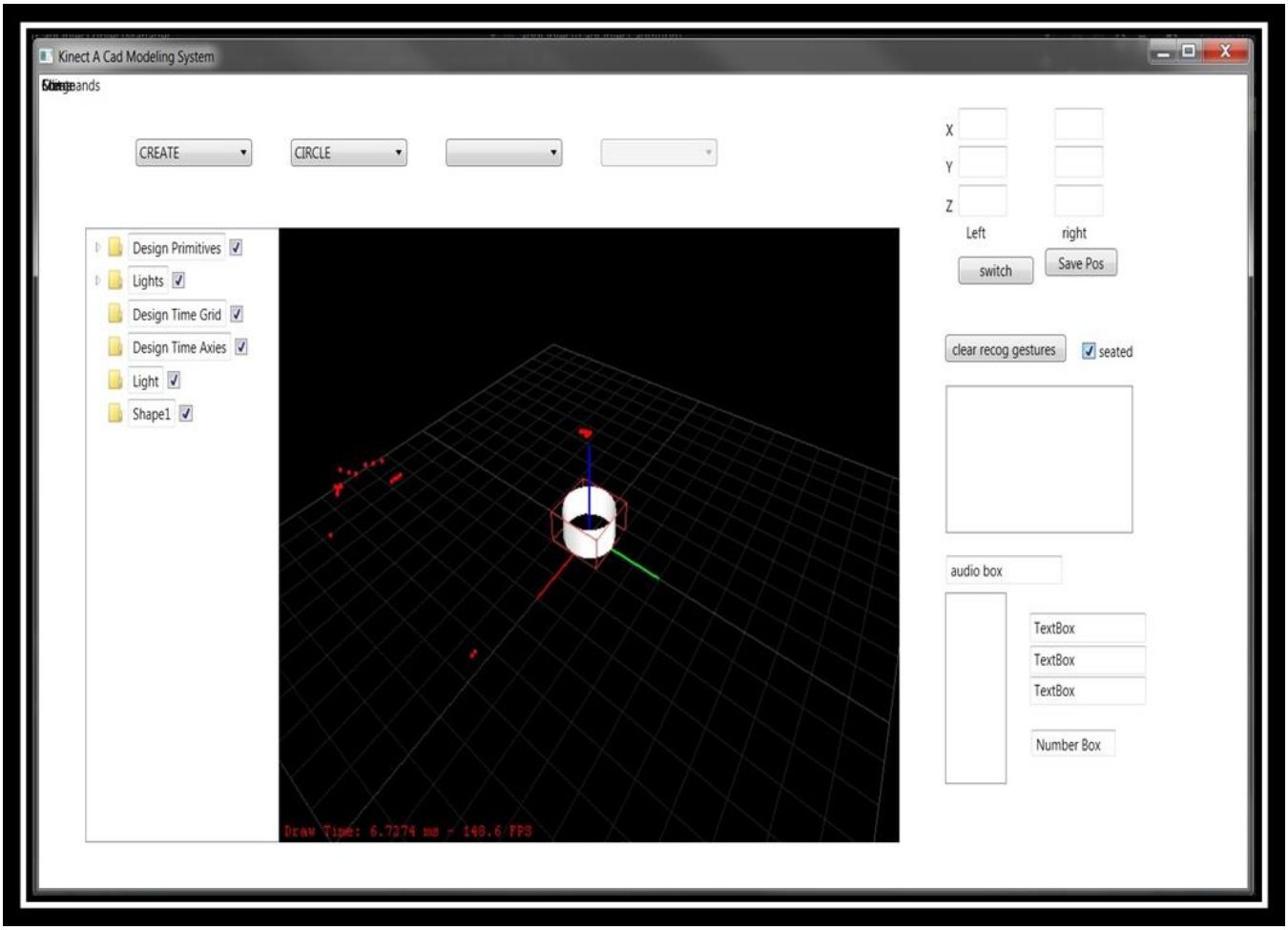
(A)***Concepts and Algorithms Generated:*** The concepts generated for this subsystem are that this subsystem will have a class that details all the different variables the CAD program needs to run. This will consist of finding keywords from the audio and mapping gestures to specific commands in the CAD program.

(B) Interface Description: The subsystem will require data from the Kinect from either the gesture recognition or the audio recognition subsystems. This data will be taken by the program and processed using keywords or gestures recognized by the Kinect. These recognized items will be placed into a class that can be passed to the CAD processing to be used.

V. Description of Final Prototype

On our final prototype we were able to create shapes such as a circle, rectangle, cube etc. And we were successful in implementing the commands like extrude, translate, cut etc. using gestures and audio commands. Using the Microsoft Kinect to retrieve the data the user input is received using gestures into Kinect, it provides crucial user information for our application. User's data collected by the Kinect, is relayed to our application as skeletons. Our interface housed all the components that are passed from the Kinect and processed by the data translation and initializes them during the software loading .The initialization processes establishes the connection between our software any Kinect connected to the computer the application is running on. These data structures are displayed through our graphical user interface (GUI), as it provides immediate feedback once a Kinect is connected. On the GUI the shapes that are created using the gesture and voice commands are also displayed.

One typical usage scenario could be school of mechanical engineering. On our first semester of senior design class we interviewed several students from mechanical engineering major and most of them find using cad software in the form of a keyboard hard .Our application will be useful in transforming a complicated usage of cad to natural hand gestures and audio commands to create complicated shapes.



VI. Results

VI.1. Prototype Test Results

Unit Tests:

Gestures

Test Case Identifier: 1-001 -- public abstract class gesture
Test Title/Name: Test Create Class - rightHandObserver
Test Summary/Description: This test creates a new instance of a gesture public abstract class gesture by creating and initializing a class “rightHandObserver” that inherits from gesture.
Assumptions: This tests assumes that rightHandObserver inherits from gesture.
Input Specification: Public rightHandObserver testGesture
Output Specification:

Expected output: The object “testGesture” should be initialized and have space created in the systems memory. The class should inherit all the methods and member variables.

Actual Output: Class was created

Test Result (Pass/Fail):PASS

Test Case Identifier: 1-002 -- public virtual void add
Test Title/Name: Test Add Gesture Points
Test Summary/Description: This tests ensures the ability to store points from a given any gesture class.
Assumptions: A gesture class has been properly initialized.
Input Specification: Kinect sensor is on, with a user present.
Output Specification:

Expected output: Once a skeleton has been displayed, the points of the gesture that has been created should be stored within the class in the member “listings” and invokes the method “Look For Gestures”

Actual Output: Points are added to the class, LookForGestures is called.

Test Result (Pass/Fail): PASS
Pass/Fail Criteria: There should be at most 20 listings within the structure, consistently updating the points.

Test Case Identifier: 1-003 -- public abstract void LookForGestures
Test Title/Name: Test Look For Gestures - Swipe Detected
Test Summary/Description: This test is specific to a given gesture class. The look for gesture performs arithmetic operations within the listings of points, and returns if a gesture has been detected..
Assumptions: rightHandObserver testSwipe has been created and initialized.

Input Specification: A user, and Kinect. The user must swipe in a horizontal motion from the user's right to left.
Output Specification:
Expected output: A gestureIsDetected function should be triggered.
Actual Output: gestureIsDetected function printed "RIGHT HAND SWIPE"
Test Result (Pass/Fail): PASS
Pass/Fail Criteria: The swipes may occur from any direction. Diagonal swipes from right to left were detected.

Test Case Identifier: 1-004 -- public abstract void LookForGestures
Test Title/Name: Test Look For Gestures - Swipe NOT Detected
Test Summary/Description: This test is specific to a given gesture class. The look for gesture performs arithmetic operations within the listings of points, and should not return anything while a gesture has not been detected...
Assumptions: rightHandObserver testSwipe has been created and initialized.
Input Specification: A user, and Kinect. The user must NOT swipe in a horizontal motion from the user's right to left.
Output Specification:
Expected output: A gestureIsDetected function should not be triggered.
Actual Output: nothing occurs
Test Result (Pass/Fail): PASS

Test Case Identifier: 1-005 -- protected void gestureIsDetected
Test Title/Name: Test Gesture Is Detected - Normal
Test Summary/Description: gestureIsDetected is triggered when a gesture has been recognized. The function then ensures that a gesture has not been recently detected.
Assumptions: rightHandObserver testSwipe has been created and initialized.
Input Specification: A user, and Kinect. The user must swipe in a horizontal motion from the user's right to left.
Output Specification:
Expected output: gestureIsDetected should update a text box
Actual Output: gestureIsDetected function printed "RIGHT HAND SWIPE"
Test Result (Pass/Fail): PASS
Pass/Fail Criteria: First swipe should be detected. then a long pause and another swipe should also be recognized.

Test Case Identifier: 1-006 -- protected void gestureIsDetected
Test Title/Name: Test Gesture Is Detected - Too Close
Test Summary/Description: This function ensures that a swipe does not get triggered more than once from an already existing swipe
Input Specification: A user, and Kinect. The user must swipe in a horizontal motion from the user's right to left.
Output Specification:
Expected output: gestureIsDetected should update a text box ONCE

Actual Output: gestureIsDetected function printed “RIGHT HAND SWIPE”
Test Result (Pass/Fail): PASS
Pass/Fail Criteria: Too close is considered detecting a gesture within one second of the last gesture.

Audio

Test Case Identifier: 2-001 - Grammar Conflict Testing for “move”.
Create a grammar such that "move" does not have any conflicts with other words.
Summary: This test case involved testing "move" for conflicts with other words in the grammar. A conflict would involve you saying a word and the speech recognizer picking up a different word.
Assumptions: The Kinect sensor is not picking up noise from any other source but the tester.
Input Specification: "Move" should be tested 10 times against the audio sensor.
Output Specification:
Expected output: Its expected that the Kinect would be able to understand "move" without any conflict for 9 of the 10 tests. The confidence associated with the word should range from 7.5 - 9.
Actual Output: "Move" had 0 conflicts with the rest of the words in the grammar.
Test Result (Pass/Fail): For 10 out of 10 checks, "move" had 0 conflicts. The confidence levels were within the appropriate range.
Pass/Fail Criteria: Passed.

Test Case Identifier: 2-002 Grammar Conflict Testing for “translate.
Create a grammar such that "translate" does not have any conflicts with other words.
Summary: This test case involved testing "translate" for conflicts with other words in the grammar. A conflict would involve you saying a word and the speech recognizer picking up a different word.
Assumptions: The Kinect sensor is not picking up noise from any other source but the tester.
Input Specification: "Translate" should be tested 10 times against the audio sensor.
Output Specification:
Expected output: Its expected that the Kinect would be able to understand "translate" without any conflict for 9 of the 10 tests. The confidence associated with the word should range from 7.5 - 9.
Actual Output: "Translate" had 0 conflicts with the rest of the words in the grammar.
Test Result (Pass/Fail): For 10 out of 10 checks, "translate" had 0 conflicts. The confidence levels were within the appropriate range.
Pass/Fail Criteria: Passed.

Test Case Identifier: 2-003 Grammar Conflict Testing for “previous”.
Create a grammar such that "previous" does not have any conflicts with other words.
Summary: This test case involved testing "previous" for conflicts with other words in the grammar. A conflict would involve you saying a word and the speech recognizer picking up a different word.
Assumptions: The Kinect sensor is not picking up noise from any other source but the tester.

Input Specification: "Previous" should be tested 10 times against the audio sensor.
Output Specification:

Expected output: Its expected that the Kinect would be able to understand "Previous" without any conflict for 9 of the 10 tests. The confidence associated with the word should range from 7.5 - 9.

Actual Output: "Previous" had 0 conflicts with the rest of the words in the grammar.

Test Result (Pass/Fail): For 10 out of 10 checks, "previous" had 0 conflicts. The confidence levels were within the appropriate range.
Pass/Fail Criteria: Passed.

Test Case Identifier: 2-004 Grammar Conflict Testing for “next”.
Create a grammar such that "next" does not have any conflicts with other words.
Summary: This test case involved testing "next" for conflicts with other words in the grammar. A conflict would involve you saying a word and the speech recognizer picking up a different word.
Assumptions: The Kinect sensor is not picking up noise from any other source but the tester.
Input Specification: "Next" should be tested 10 times against the audio sensor.
Output Specification:

Expected output: Its expected that the Kinect would be able to understand "Next" without any conflict for 9 of the 10 tests. The confidence associated with the word should range from 7.5 - 9.

Actual Output: "Next" had 0 conflicts with the rest of the words in the grammar.

Test Result (Pass/Fail): For 10 out of 10 checks, "next" had 0 conflicts. The confidence levels were within the appropriate range.
Pass/Fail Criteria: Passed.

Test Case Identifier: 2-005 Grammar Conflict Testing for “select”.
Create a grammar such that "set" did have conflict with “select”.
Summary: This test case involved testing "set" for conflicts with other words in the grammar. A conflict would involve you saying a word and the speech recognizer picking up a different word.
Assumptions: The Kinect sensor is not picking up noise from any other source but the tester.
Input Specification: "Set" should be tested 10 times against the audio sensor.
Output Specification:

Expected output: Its expected that the Kinect would be able to understand "set" without any conflict for 9 of the 10 tests. The confidence associated with the word should range from 7.5 - 9.

Actual Output: "Set" had 9 conflicts with “select”.

Test Result (Pass/Fail): For 10 out of 10 checks, "Set" had 9 conflicts. The confidence levels were within the appropriate range.
Pass/Fail Criteria: Failed.

Test Case Identifier: 2-006 Grammar Conflict Testing for “set”.
Create a grammar such that "select" did have conflict with “set”.
Summary: This test case involved testing "select" for conflicts with other words in the grammar. A conflict would involve you saying a word and the speech recognizer picking up a different word.

Assumptions: The Kinect sensor is not picking up noise from any other source but the tester.
Input Specification: "select" should be tested 10 times against the audio sensor.
Output Specification:

Expected output: Its expected that the Kinect would be able to understand "select" without any conflict for 9 of the 10 tests. The confidence associated with the word should range from 7.5 - 9.

Actual Output: "Select" had 3 conflicts with “set”. The confidence levels were within the appropriate range.

Test Result (Pass/Fail): For 10 out of 10 checks, "Set" had 3 conflicts. The confidence levels were within the appropriate range.
Pass/Fail Criteria: Failed.

Test Case Identifier: 2-007 Grammar Conflict Testing for “circle”.
Create a grammar such that "circle" does not have any conflicts with other words.
Summary: This test case involved testing "circle" for conflicts with other words in the grammar. A conflict would involve you saying a word and the speech recognizer picking up a different word.
Assumptions: The Kinect sensor is not picking up noise from any other source but the tester.
Input Specification: "Circle" should be tested 10 times against the audio sensor.
Output Specification:

Expected output: Its expected that the Kinect would be able to understand "circle" without any conflict for 9 of the 10 tests. The confidence associated with the word should range from 7.5 - 9.

Actual Output: "Circle" had 0 conflicts with the rest of the words in the grammar.

Test Result (Pass/Fail): For 10 out of 10 checks, "circle" had 0 conflicts. The confidence levels were within the appropriate range.
Pass/Fail Criteria: Passed.

Test Case Identifier: 2-008 Grammar Conflict Testing for “square”.
Create a grammar such that "square" does not have any conflicts with other words.
Summary: This test case involved testing "square" for conflicts with other words in the grammar. A conflict would involve you saying a word and the speech recognizer picking up a different word.
Assumptions: The Kinect sensor is not picking up noise from any other source but the tester.
Input Specification: "Square" should be tested 10 times against the audio sensor.
Output Specification:

Expected output: Its expected that the Kinect would be able to understand "square" without any conflict for 9 of the 10 tests. The confidence associated with the word should range from 7.5 - 9.

Actual Output: "Square" had 0 conflicts with the rest of the words in the grammar.

Test Result (Pass/Fail): For 10 out of 10 checks, "square" had 0 conflicts. The confidence levels were within the appropriate range.
Pass/Fail Criteria: Passed.

Test Case Identifier: 2-009 Grammar Conflict Testing for “shutdown”.

Create a grammar such that "shutdown" does not have any conflicts with other words.
Summary: This test case involved testing "shutdown" for conflicts with other words in the grammar. A conflict would involve you saying a word and the speech recognizer picking up a different word.
Assumptions: The Kinect sensor is not picking up noise from any other source but the tester.
Input Specification: "Shutdown" should be tested 10 times against the audio sensor.
Output Specification:

Expected output: Its expected that the Kinect would be able to understand "shutdown" without any conflict for 9 of the 10 tests. The confidence associated with the word should range from 7.5 - 9.

Actual Output: "Shutdown" had 0 conflicts with the rest of the words in the grammar.

Test Result (Pass/Fail): For 10 out of 10 checks, "shutdown" had 0 conflicts. The confidence levels were within the appropriate range.
Pass/Fail Criteria: Passed.

Test Case Identifier: 2-010 Grammar Conflict Testing for “initialize”.
Create a grammar such that "initialize" does not have any conflicts with other words.
Summary: This test case involved testing "initialize" for conflicts with other words in the grammar. A conflict would involve you saying a word and the speech recognizer picking up a different word.
Assumptions: The Kinect sensor is not picking up noise from any other source but the tester.
Input Specification: "Initialize" should be tested 10 times against the audio sensor.
Output Specification:

Expected output: Its expected that the Kinect would be able to understand "initialize" without any conflict for 9 of the 10 tests. The confidence associated with the word should range from 7.5 - 9.

Actual Output: "Initialize" had 0 conflicts with the rest of the words in the grammar.

Test Result (Pass/Fail): For 10 out of 10 checks, "initialize" had 0 conflicts. The confidence levels were within the appropriate range.
Pass/Fail Criteria: Passed.

Test Case Identifier: 2-011 Grammar Conflict Testing for “cut”.
Create a grammar such that "cut" does not have any conflicts with other words.
Summary: This test case involved testing "cut" for conflicts with other words in the grammar. A conflict would involve you saying a word and the speech recognizer picking up a different word.
Assumptions: The Kinect sensor is not picking up noise from any other source but the tester.
Input Specification: "cut" should be tested 10 times against the audio sensor.
Output Specification:

Expected output: Its expected that the Kinect would be able to understand "cut" without any conflict for 9 of the 10 tests. The confidence associated with the word should range from 7.5 - 9.

Actual Output: "cut" had 0 conflicts with the rest of the words in the grammar.

Test Result (Pass/Fail): For 10 out of 10 checks, "cut" had 0 conflicts. The confidence levels were within the appropriate range.
Pass/Fail Criteria: Passed.

Test Case Identifier: 2-012 Grammar Conflict Testing for “choose”.
Create a grammar such that “choose” does not have any conflicts with other words.
Summary: This test case involved testing “choose” for conflicts with other words in the grammar. A conflict would involve you saying a word and the speech recognizer picking up a different word.
Assumptions: The Kinect sensor is not picking up noise from any other source but the tester.
Input Specification: “choose” should be tested 10 times against the audio sensor.
Output Specification:

Expected output: Its expected that the Kinect would be able to understand “choose” without any conflict for 9 of the 10 tests. The confidence associated with the word should range from 7.5 - 9.

Actual Output: “Choose” had 0 conflicts with the rest of the words in the grammar.

Test Result (Pass/Fail): For 10 out of 10 checks, “choose” had 0 conflicts. The confidence levels were within the appropriate range.
Pass/Fail Criteria: Passed.

Test Case Identifier: 2-013 Grammar Conflict Testing for “zero.
Create a grammar such that “zero” does not have any conflicts with other numbers.
Summary: This test case involved testing “zero” for conflicts with other numbers in the grammar. A conflict would involve you saying a number and the speech recognizer picking up a different number.
Assumptions: The Kinect sensor is not picking up noise from any other source but the tester.
Input Specification: “Zero” should be tested 10 times against the audio sensor.
Output Specification:

Expected output: Its expected that the Kinect would be able to understand “zero” without any conflict for 9 of the 10 tests. The confidence associated with the number should range from 7.5 - 9.

Actual Output: “Zero” had 0 conflicts with the rest of the numbers in the grammar.

Test Result (Pass/Fail): For 10 out of 10 checks, “zero” had 0 conflicts. The confidence levels were within the appropriate range.
Pass/Fail Criteria: Passed.

Test Case Identifier: 2-014 Grammar Conflict Testing for “one”.
Create a grammar such that “one” does not have any conflicts with other numbers.
Summary: This test case involved testing “one” for conflicts with other numbers in the grammar. A conflict would involve you saying a number and the speech recognizer picking up a different number.
Assumptions: The Kinect sensor is not picking up noise from any other source but the tester.
Input Specification: “One” should be tested 10 times against the audio sensor.
Output Specification:

Expected output: Its expected that the Kinect would be able to understand “one” without any conflict for 9 of the 10 tests. The confidence associated with the number should range from 7.5 - 9.

Actual Output: “One” had 0 conflicts with the rest of the numbers in the grammar.

Test Result (Pass/Fail): For 10 out of 10 checks, “one” had 0 conflicts. The confidence levels

were within the appropriate range.
Pass/Fail Criteria: Passed.

Test Case Identifier: 2-015 Grammar Conflict Testing for “two”.
Create a grammar such that “two” does not have any conflicts with other numbers.
Summary: This test case involved testing “two” for conflicts with other numbers in the grammar. A conflict would involve you saying a number and the speech recognizer picking up a different number.
Assumptions: The Kinect sensor is not picking up noise from any other source but the tester.
Input Specification: “Two” should be tested 10 times against the audio sensor.
Output Specification:

> **Expected output:** Its expected that the Kinect would be able to understand “two” without any conflict for 9 of the 10 tests. The confidence associated with the number should range from 7.5 - 9.
> **Actual Output:** “Two” had 0 conflicts with the rest of the numbers in the grammar.

Test Result (Pass/Fail): For 10 out of 10 checks, “two” had 0 conflicts. The confidence levels were within the appropriate range.
Pass/Fail Criteria: Passed.

Test Case Identifier: 2-016 Grammar Conflict Testing for “three”.
Create a grammar such that “three” does not have any conflicts with other numbers.
Summary: This test case involved testing “three” for conflicts with other numbers in the grammar. A conflict would involve you saying a number and the speech recognizer picking up a different number.
Assumptions: The Kinect sensor is not picking up noise from any other source but the tester.
Input Specification: “Three” should be tested 10 times against the audio sensor.
Output Specification:

> **Expected output:** Its expected that the Kinect would be able to understand “zero” without any conflict for 9 of the 10 tests. The confidence associated with the number should range from 7.5 - 9.
> **Actual Output:** “Three” had 0 conflicts with the rest of the numbers in the grammar.

Test Result (Pass/Fail): For 10 out of 10 checks, “three” had 0 conflicts. The confidence levels were within the appropriate range.
Pass/Fail Criteria: Passed.

Test Case Identifier: 2-017 Grammar Conflict Testing for “four”.
Create a grammar such that “four” does not have any conflicts with other numbers.
Summary: This test case involved testing “four” for conflicts with other numbers in the grammar. A conflict would involve you saying a number and the speech recognizer picking up a different number.
Assumptions: The Kinect sensor is not picking up noise from any other source but the tester.
Input Specification: “Four” should be tested 10 times against the audio sensor.
Output Specification:

> **Expected output:** Its expected that the Kinect would be able to understand “zero” without any conflict for 9 of the 10 tests. The confidence associated with the number

should range from 7.5 - 9.

Actual Output: “Four” had 0 conflicts with the rest of the numbers in the grammar.

Test Result (Pass/Fail): For 10 out of 10 checks, “four” had 0 conflicts. The confidence levels were within the appropriate range.

Pass/Fail Criteria: Passed.

Test Case Identifier: 2-018 Grammar Conflict Testing for “five”.

Create a grammar such that “five” does not have any conflicts with other numbers.

Summary: This test case involved testing “five” for conflicts with other numbers in the grammar. A conflict would involve you saying a number and the speech recognizer picking up a different number.

Assumptions: The Kinect sensor is not picking up noise from any other source but the tester.

Input Specification: “Five” should be tested 10 times against the audio sensor.

Output Specification:

Expected output: Its expected that the Kinect would be able to understand “five” without any conflict for 9 of the 10 tests. The confidence associated with the number should range from 7.5 - 9.

Actual Output: “Five” had 0 conflicts with the rest of the numbers in the grammar.

Test Result (Pass/Fail): For 10 out of 10 checks, “five” had 0 conflicts. The confidence levels were within the appropriate range.

Pass/Fail Criteria: Passed.

Test Case Identifier: 2-019 Grammar Conflict Testing for “six”.

Create a grammar such that “six” does not have any conflicts with other numbers.

Summary: This test case involved testing “six” for conflicts with other numbers in the grammar. A conflict would involve you saying a number and the speech recognizer picking up a different number.

Assumptions: The Kinect sensor is not picking up noise from any other source but the tester.

Input Specification: “Six” should be tested 10 times against the audio sensor.

Output Specification:

Expected output: Its expected that the Kinect would be able to understand “zero” without any conflict for 9 of the 10 tests. The confidence associated with the number should range from 7.5 - 9.

Actual Output: “Six” had 0 conflicts with the rest of the numbers in the grammar.

Test Result (Pass/Fail): For 10 out of 10 checks, “six” had 0 conflicts. The confidence levels were within the appropriate range.

Pass/Fail Criteria: Passed.

Test Case Identifier: 2-020 Grammar Conflict Testing for “seven”.

Create a grammar such that “seven” does not have any conflicts with other numbers.

Summary: This test case involved testing “seven” for conflicts with other numbers in the grammar. A conflict would involve you saying a number and the speech recognizer picking up a different number.

Assumptions: The Kinect sensor is not picking up noise from any other source but the tester.

Input Specification: “Seven” should be tested 10 times against the audio sensor.

Output Specification:

Expected output: Its expected that the Kinect would be able to understand “seven” without any conflict for 9 of the 10 tests. The confidence associated with the number should range from 7.5 - 9.

Actual Output: “Seven” had 0 conflicts with the rest of the numbers in the grammar.

Test Result (Pass/Fail): For 10 out of 10 checks, “seven” had 0 conflicts. The confidence levels were within the appropriate range.

Pass/Fail Criteria: Passed.

Test Case Identifier: 2-021 Grammar Conflict Testing for “eight”.

Create a grammar such that “eight” does not have any conflicts with other numbers.

Summary: This test case involved testing “eight” for conflicts with other numbers in the grammar. A conflict would involve you saying a number and the speech recognizer picking up a different number.

Assumptions: The Kinect sensor is not picking up noise from any other source but the tester.

Input Specification: “Eight” should be tested 10 times against the audio sensor.

Output Specification:

Expected output: Its expected that the Kinect would be able to understand “zero” without any conflict for 9 of the 10 tests. The confidence associated with the number should range from 7.5 - 9.

Actual Output: “Eight” had 0 conflicts with the rest of the numbers in the grammar.

Test Result (Pass/Fail): For 10 out of 10 checks, “eight” had 0 conflicts. The confidence levels were within the appropriate range.

Pass/Fail Criteria: Passed.

Test Case Identifier: 2-022 Grammar Conflict Testing for “nine”.

Create a grammar such that “nine” does not have any conflicts with other numbers.

Summary: This test case involved testing “nine” for conflicts with other numbers in the grammar. A conflict would involve you saying a number and the speech recognizer picking up a different number.

Assumptions: The Kinect sensor is not picking up noise from any other source but the tester.

Input Specification: “Nine” should be tested 10 times against the audio sensor.

Output Specification:

Expected output: Its expected that the Kinect would be able to understand “nine” without any conflict for 9 of the 10 tests. The confidence associated with the number should range from 7.5 - 9.

Actual Output: “Nine” had 0 conflicts with the rest of the numbers in the grammar.

Test Result (Pass/Fail): For 10 out of 10 checks, “nine” had 0 conflicts. The confidence levels were within the appropriate range.

Pass/Fail Criteria: Passed.

Objects

Test Case Identifier: 3-001 -- public Circle

Create cadObjects Circles correctly from user input.

Summary: This test was done for testing if Circles were properly constructed by our cadObject classes.
Assumptions: Valid coordinates are given.
Input Specification: Origin from the user hand position..
Output Specification:
Expected output: An origin that waits for an endpoint to finish defining the shape
Actual Output: An origin that waits for an endpoint to finish defining the shape
Test Result (Pass/Fail): Passed**.**

Test Case Identifier: 3-002 -- public void Circle.endPoint
Create a Circle correctly from user input.
Summary: This test was done for testing if Circle properly constructed by our cadObject classes when given the endpoint, or radius, by the user.
Assumptions: Given point not equal to the origin
Input Specification: Right hand position.
Output Specification:
Expected output: A circle with 72 points space five degrees apart at a distance of radius from the origin.
Actual Output: A circle with 72 points space five degrees apart at a distance of radius from the origin.
Test Result (Pass/Fail): Passed.

Test Case Identifier: 3-003 -- public Square
Create cadObjects Square correctly from user input.
Summary: This test was done for testing if Squares were properly constructed by our cadObject classes.
Assumptions: Valid coordinates are given.
Input Specification: Origin from the user hand position..
Output Specification:
Expected output: An origin that waits for an endpoint to finish defining the shape
Actual Output: An origin that waits for an endpoint to finish defining the shape
Test Result (Pass/Fail): Passed**.**

Test Case Identifier: 3-004 -- public void Square.endPoint
Create a Square correctly from user input.
Summary: This test was done for testing if Circle properly constructed by our cadObject classes when given two endpoint for the bottom left and upper right
Assumptions: Given points not equal to the origin, and also don't have the same x or y value.
Input Specification: Right hand position.
Output Specification:
Expected output: A square with 4 points, two from the user and two generated.
Actual Output: A square with 4 points, two from the user and two generated.
Test Result (Pass/Fail): Passed.

Test Case Identifier: 3-005 -- public void cadObject.GetPoints()
Return points from a given shape that are stored in the class.
Summary: This test was done for returning points from an object so they could be used for different functions such as extruding or cuts.
Assumptions: Shape, either a square or circle, was properly constructed.
Input Specification: Existing shape.
Output Specification:

Expected output: A list of points corresponding to a shape.
Actual Output: A list of points corresponding to a shape.

Test Result (Pass/Fail): Passed.

Test Case Identifier: 2-005 -- cadObject.Translate
Move all the points in a given shape to a direction given by the user.
Summary: A shape is given a point to translate to, and adds the direction to all its points so it
Assumptions: Shape was properly constructed.
Input Specification: Existing shape and a hand position.
Output Specification:

Expected output:A shape with points properly translated in the proper direction
Actual Output:A list of shapes translated in the proper direction but not quite the right distance.

Test Result (Pass/Fail): Passed.
This test passed because the object managed to translate in the proper direction, it just didn't have the proper scale from the Kinect coordinates to the openGL coordinates.

System Tests:

Test Case Identifier: 5-001 OpenGL Rendering
Render shapes with OpenGL in such a way that the user can Identify them
Summary: This test was aimed at seeing if the program could render shapes in a way that the user would accurately see the shapes they created and have them display to the screen.
Assumptions: Proper shapes parameters were put in and all input by the user was valid. Shapes existed prior to rendering.
Input Specification: Squares and/or Circle cadObjects for the program to render.
Output Specification:

Expected output: We were expecting a square and/or circle to show up on the screen
Actual Output: Squares and circle showed up on the specific tests done for them, however sizes were not always as expected.
[You may include tables, graphs, images to illustrate your results, if needed.]

Test Result (Pass/Fail): Passed
Pass/Fail Criteria: The sizes don't have to do with the rendering to a certain extent. There is a scaling issue between coordinates given from the Kinect and those used for openGL, however we still generate shapes correctly which was the desired outcome.

Test Case Identifier: 5-002 -- void ProcessFrame

Test Title/Name: Test Process Frame - No User
Test Summary/Description: This test will display no skeletons to the screen
Assumptions: Kinect and skeleton processing enabled.
Input Specification: A Kinect
Output Specification:
Expected output: The GUI should not display any skeleton data.
Actual Output: No skeleton data has been displayed.
Test Result (Pass/Fail): PASS

Test Case Identifier: 5-003 -- void ProcessFrame
Test Title/Name: Test Process Frame - User
Test Summary/Description: This test will display unique skeletons to the screen
Assumptions: Kinect and skeleton processing enabled.
Input Specification: A Kinect and a user in the field of view
Output Specification:
Expected output: The GUI should display skeleton data corresponding to what the user is doing
Actual Output: An accurate skeleton has been displayed
Test Result (Pass/Fail): PASS
Pass/Fail Criteria: If there are more than one user, multiple skeletons should displayed

Test Case Identifier: 5-004 -- private void OpenGLControl_OpenGLDraw
Test Title/Name: Test OpengGLDraw Square
Test Summary/Description: Depending on the user's skeleton data, a square should render inbetween the audio commands square and set.
Input Specification: A Kinect and a user in the field of view
Output Specification:
Expected output: The GUI should display a white square while the object is being created.
Actual Output: An immediate visual feedback of a square being created.
Test Result (Pass/Fail): PASS

Test Case Identifier: 6-001 Initialize
This test involved the correct creation of our gesture and audio sensors.
Summary: This test case involves testing whether the Kinect sensors are being correctly initialized.
Assumptions: The computer connected to the Kinect has all the correct drivers installed. Kinect is being detected by the computer.
Input Specification: "Move" should be tested 10 times against the audio sensor.
Output Specification:
Expected output: That audio and gesture recognition is being correctly set up and functional.
Actual Output: Both audio and gesture recognition objects and sensors are initialized

correctly and accessible to the rest of our correlating functions.
Test Result (Pass/Fail): The test passed because our sensors and recognizer objects were being initialized.
Pass/Fail Criteria: Passed.

Test Case Identifier: 6-002 Start
This test involved testing if our system was running correctly at the beginning of runtime.
Summary: This test case involves whether all of our initial functionality works as the system is starting to run.
Assumptions: There is a Kinect plugged into the computer doing the testing.
Input Specification: The Kinect is on and the user is present.
Output Specification:

Expected output: The main window should display with a video feed showing what the Kinect sensor sees and a skeletal model for each user in range. After 5 seconds (it takes 5 seconds for audio to be initialized) the audio box should start outputting responses from the audio sensor.

Actual Output: Main window showed up within acceptable speed requirements. The user was correctly being displayed on the screen with a skeleton. Audio was correclty initialized and proper output was shown in the audio box.

Test Result (Pass/Fail): The test passed because our system started correctly with all proper functionalities.
Pass/Fail Criteria: Passed.

Test Cases Omitted:

Gestures: The following functions have been omitted due to superclass and inheritance redundancy.

rightHandObserver.LookForGestures()
rightHandObserver.gestureIsDetected()
rightHandObserver.add()
leftHandObserver.LookForGestures()
leftHandObserver.gestureIsDetected()
leftHandObserver.add()
twoHandGesture.LookForGestures()
twoHandGesture.gestureIsDetected()
twoHandGesture.add()

Speech Recognition: Our speech recognition had the following functionalities omitted due to redundancy.

SRESpeechRecognized()
SRESpeechHypothesized()
SRESpeechRejected()

VI.2. Additional Results

No additional results.

VII. Limitations and Recommendations:

There are a few limitations regarding our project. One of the significant ones is how terrible the Microsoft Kinect microphones are at noise cancelling. During our demonstrations, we were able to see just how awkwardly the audio commands worked since our surroundings were noisy. The best way to solve this is to upgrade to the new Kinect 2.0. The new Kinect has 3 noise cancelling microphones in the back, and one microphone in the front that focuses on the user. This will benefit our program in the sense that it only has to focus on audio from one main microphone, and can process and cancel the rest of the noise it is picking up.

Another limitation is the preciseness of the gestures. The current Kinect sensors have a 640x480 camera. This is too low of a resolution for preciseness, it's more of a "pick up whatever you can and we'll use that to track gestures". The Kinect 2.0 has dual 1920x1080 cameras. This is the current high definition standard that has a stunning amount of clarity. This clarity is what will help the program in picking up miniscule gestures or pinches to fine tune the shapes.

The new Kinect 2.0 is highly recommended to continue this project further. The benefits include a modern, and updated SDK with more API's. There are hardly any cons other than the cost of the new Kinect.

VIII. Conclusions and Future Work

In conclusion, the project was a difficult one in the beginning. As we broke it down into much smaller components did we know how to approach our project. None of the group members had any experience with CAD systems, or OpenGL graphics capabilities. Ultimately, the final prototype works wonderfully and hopefully it will stir up the innovative markets.

The future work for this project entails working with 3D printers. Yes, we are able to create shapes with voice commands and gestures. Now we need a way to commercialize this product. With the patents on 3D printing expiring at the end of the year, there will be cheaper and more “made for home” 3D printers that aren’t ridiculously expensive. We hope that once the user is done making the shape, they will be able to 3D print it. This will be great for children as they have the freedom to create almost anything, and then print their creation to see how it actually looks. We’re hoping this creates an interest for kids to become engineers in the future.

IX. Acknowledgements

First and foremost, we would like to thank Dr. Sakire Arslan-Ay. She stepped up to guide us when our mentor was unreachable for assistance. This project wouldn't be where it is if it wasn't for her incredible knowledge and a sense of design.

X. References

Channel 9 - Microsoft; http://channel9.msdn.com

Coding4Fun; http://c4fKinect.codeplex.com

Software Testing - A Craftman's Approach ; Binder

Everything OpenGL - http://www.opengl-tutorial.org/

STL File Origns - http://bastech.com/sla/techtips/STLfiles.asp

OpenGl Programming Guide - http://glprogramming.com/red/

Kinect 2 AutoCAD - http://through-the-interface.typepad.com/through_the_interface/2011/11/au-2011-samples-integrating-microsoft-Kinect-with-autocad.html

OpenGL Skeleton - http://www.holmes3d.net/graphics/skeleton/

XI. Appendix A – Team Information

(Right to left): Young Chon, Lyle Dallas, Karan Sharma, Michael Belay, and Brian Fleck.

XII. Appendix B - Project Management

Tuesday: Weekly meetings for about an hour with team members to see where we are after the weekend and what we will be trying to accomplish this week.

Thursday: Weekly meetings with the instructor.

- This meeting was the most beneficial as we got professional feedback on the project. It was at these meetings we decided on new routes to take on merging different sections of the project and getting it work correctly.

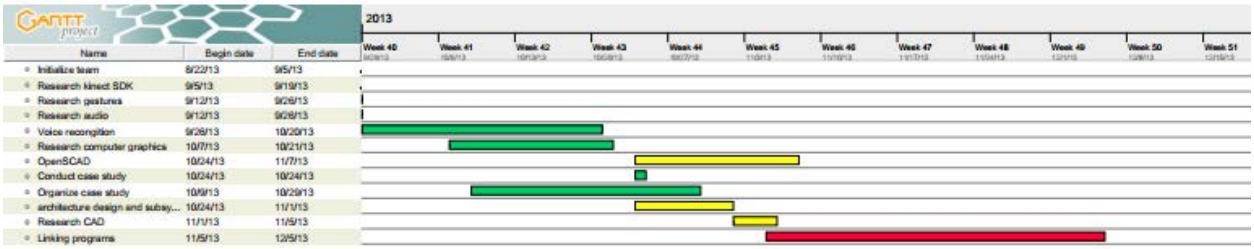
Saturday: Weekly meetings with some team members who were working together on respective portions of the project.

- During these meetings we mostly worked on the new things we talked about on Thursday.

Sunday: Project related team activities. Whether it be trying new implementations or helping test new/old ones.

- We finished up Saturday's work so some of the others could test the functionalities and see whether or not they were beneficial to our project or not.

The Gantt Chart from October 31st, 2013:



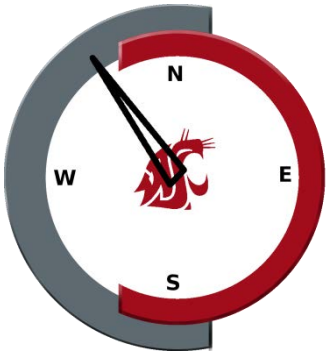
We didn't create a Gantt Chart for the second semester since the previous semester gave us our own sections to work on, and we followed through on those.

Appendix E:

CptS 423 Team Red Dragon's Final Report

Project Lost

Final Report



Mentor:

Andy O'Fallon

Team Red Dragons

Beau Schwieso

David Barajas

Joey Smith

CptS 423 Software Design Project II
Spring 2014

TABLE OF CONTENTS

I. Executive Summary

At its current state, Project Lost implements the basic requirements that we established in the first semester of its development. These requirements include navigating between buildings, inside them, user account creation, and student schedule importation. While time constraints prevented implementation, the groundwork has been put into place for some of the secondary objectives.

Of the secondary objectives, the most desired on our part was being able to change the parameter used to deterministically seek the destination. The primary hindrance for this feature was the time constraints on data gathering. Future updates will build on this groundwork by implementing features to allow users to navigate using handicap accessible features, and limiting paths to prevent steep hills.

II. Introduction

Project Lost implements graph theory and global positioning software to provide students of the Washington State University campus with a more detailed and accurate approach to path finding. The main objective is to create a more precise and intuitive solution than the campus map or Google maps could provide outdoors, while providing indoor navigation as well.

Project Lost is a way to increase students' ability to navigate through Washington State University's campus by not only making information regarding the locations of buildings and classrooms more readily accessible but also by not restricting them to sidewalks and paths. In this way students save themselves the need to find unknown buildings and rooms on a map, as well as finding a viable route to their destination. Essentially, Project Lost provides the information about campus, to incoming students and faculty, that students and faculty would normally acquire with experience.

Project Lost sets itself apart from other navigational software by giving the application the ability to not only traverse various types of terrain (i.e. Fields), but also the ability of directing users through structures, taking them directly to their classroom. The scope of the project has been to implement navigation throughout the central area of campus. The fundamental component of our application deterministically guides the user from point A to point B. This is achieved through a modified version of the A* algorithm which deterministically searches for the destination across a grid of nodes that span the entire Pullman campus.

III. System Requirements Specification

III.1. Stakeholders

Our stakeholders would be the group members, Sakire, Andy O'Fallon, and EECS. Our clients would be anyone that would be willing to download the app for use, this would include, but not be limited to WSU students and faculty. Our stakeholder's request that this application be responsive and not 'heavy' (using too much memory from the phone) along with the ability to dynamically change based on everyday data. We also are expected to deliver an application that will have the ability to search for classes and add them to build the student's 'profile'. Our client's request that this application be very user friendly, easy to navigate, easy to sign up and easy to maintain (from semester to semester). They also would like the ability to have the application work differently based on the user's preferences; meaning whether or not they care about avoiding busy paths.

III.2. Functional Requirements

A list of our current functional requirements: Execution on all major phone platforms, a searchable class database, ability to import class schedule, ability to import classes manually, navigation over a map interface, Red Dragon account creation, indoor building navigation using floor plans, path finding has the ability to determine shortest/least busy/flattest routes, server side calculation of traffic, ability to export class schedule to calendar, web page to sign up and account management, ability to gather user feedback, and ability for user to define a manual starting location.

IV. Software Design

This section should describe the final design of your software system.
Revise sections II, III, and IV in your “Solution Approach” document to reveal the changes in your project design and include them here:

IV.1. Architecture Design

IV.1.1. Overview

Project Lost employs a Model, View, Controller architecture. An MVC architecture is the best solution for Project Lost because the limited resources available to an application on a smart phone require that it be as efficient as possible. The MVC architecture allows us to circumvent this problem altogether by moving a portion of the load to a third party, in this case the server.

The application, located on the device, is largely only responsible for the view and controller portion of the MVC. This is represented by the UI and logic in the graph. Three versions of the UI will be developed, one for each mobile device. This is not only necessary to deploy, but we are going to design each view differently to conform to the look and feel of the rest of the operating system. The backend will perform path finding from their current GPS location to the desired destination. The phone will need to store the base node network grid on the smart phone itself, but this will still have a small separation from the logic.

With the exception of the node network, the server is responsible for storing the vital information too complicated or time consuming to compute on a mobile device. The server is responsible for querying WSU’s AD for students and their schedules. With these schedules it computes the projected traffic levels in different areas of campus, it also readily makes available these schedules for students to import without the student having to wait for the server to be queried.

We also use a factory pattern to provide pluggable Storage types, GPS types, and Map path finding algorithms. This way we can provide a generic implementation and change it out should one implementation work better than another.

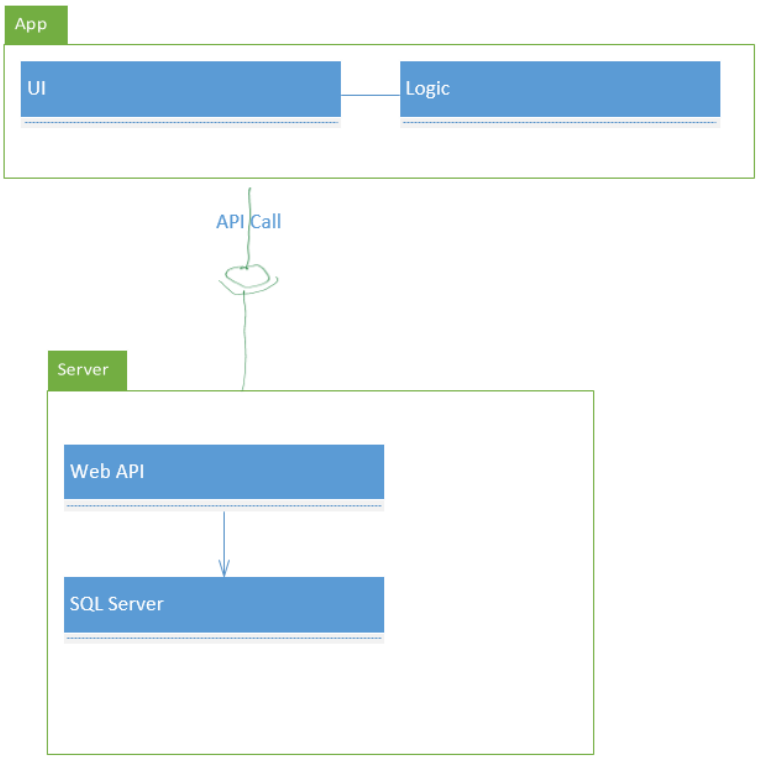


Figure IV.1.1 - Basic System Parts

IV.1.2. Subsystem Decomposition

Our system will take advantage of the MVC style design pattern. Since our goal was to not have to rewrite the backend logic for each mobile device we created, we use the help of the Xamarin framework to allow us to plug in an interface. Our logic end will be separated based on their group of nodes, map, schedules, storage, and finally a piece of logic that will tie them together, the State object.

IV.2.1. APP

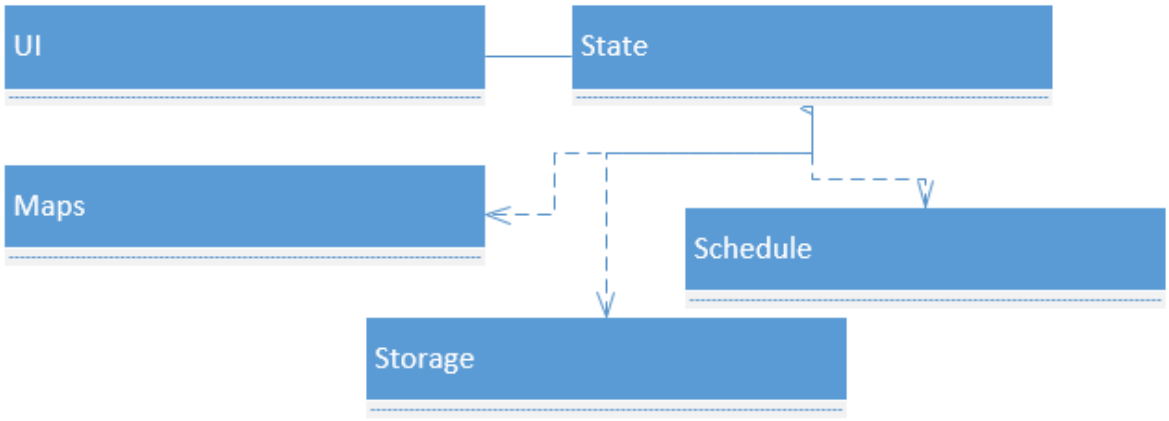


Figure IV.2.2 - App Subsystems

IV.2.1.1 UI

(i) The UI portion will need to be rewritten for each mobile device we are deploying on. C# uses callback events to handle the separation from logic and UI. We will separate the callbacks even further into a “context” or state class that holds reference to the rest of the app’s logic. This will be described below.

(ii) The UI will be event driven, meaning we just need to hook into various events to get input from the user, and to display to the user. The idea of separating our input/output and our logic is very Model-View-Controller esque. This approach makes for a flexible approach in changing the UI for each platform.

(iii) The UI will have various defined callbacks and hooks for registering form submitting, to displaying results and navigation tasks. The hardest part will be to display the map and directions to the use on three different platforms. This is why a Google Maps control will be used which will have the same interfaces and properties regardless of the platform used. Any event called will use the logic provided by the state object which is initialized on app startup and contains all the app’s current status.

Services Provided:

1.

Service provided to: State

Description: Various user inputs such as button clicks, what button was clicked, settings changed, etc.

values: UI makes using our app a more enjoyable experience.

Services Required:

.Net Framework and Xamarin.

IV.2.1.2 State

(i) The state subsystem will hold all together all the components internally of the app's logic. The APP's UI will call this object's public methods and properties to perform various tasks such as finding a route between two points, or pulling and saving the current schedule, even logging in to the app with the server.
(ii) The state object allows us to simply pass around the state object instead of relying on global namespaces and static objects which might take up more space or cause headaches when things might need resetting.
Services Provided:
1.
Service provided to: Schedule, Map, Storage, UI
Description: Central location of all application settings and logic.
values: Prevents need of global and scattered settings and logic. Bundled code is easier to maintain than scattered code. This method also allows us to pass the state around making testing easier to plug in our own "state" to configure.

Services Required:
UI, Schedule, Storage, Map

IV.2.1.3 Schedule
(i) The schedule subsystem will handle the current agenda for navigation. This subsystem will be an intermediary from our system and the data storage on the device or web server. The schedule is actually a very simple interface to handle the schedule, who owns this schedule, and where this schedule takes place. This can also be represented on the web server database side of things.
(ii) The schedule will have the ability to allow the app internally to access the current agenda, the network id used for pulling this schedule, and the campus this schedule is for (this interface is to allow possible expansions to other WSU campuses if we decide to do so).
(iii) This subsystem really is just a class representation of what is stored on our web server database and cached locally on the device of where the user wishes to navigate throughout the day. This class structure makes it easier to handle the data throughout the application.
Services Provided:
1.
Service provided to: State
Description: Place to represent the current user's agenda and information about their schedule.
values: Makes accessing their information simple and straightforward in code.

Services Required:
State, Storage, Map

IV.2.1.4 Map
(i) The Map side of things will most likely be the most advanced part of the entire program. This interface is responsible for combining all the separate data points and performing the algorithm search.

(ii) The Maps interface will handle most of the navigation items. The maps attributes allow us to plug in a GPS interface that allows us to plug in any kind of logic to tell the app the current location. The Maps interface will also have a subclass that's sole duty is to calculate a route between 2 node positions. This class will use a factory type creation so that we can plug in various algorithm types without huge messes. The most likely algorithm we will be using is the A* algorithm. This algorithm is essentially a DFS, or Depth First Search, algorithm using a heuristic. In our case, the heuristic consists of the traffic percent, change in height, distance, and the path being covered or not. A* is the better of the algorithms available in terms of large data set pathfinding and should work greatly for our usage.
(iii) The maps will have the interface for pathfinding, searching for buildings, and the various nodes available to the application (or the entire node database should the streaming model not be used).

Services Provided:

1.
Service provided to: State
Description: Performs node loading and searching
values: A very dynamic and pluggable system to handle node searching and loading.

Services Required:
Schedule, Storage, State

IV.2.1.5 Storage
(i) The storage utility will handle the persistence of the application data.
(ii) The storage class will translate classes between xml and binary to their in-memory representations. The idea of having the data stored in a query/relational database was thought about, however the costs of saving time for loading and storing the nodes into memory were offset by the costs of doing relational searches to determine the best path. The A* algorithm would not work nicely with a query database storage.
(iii) The storage utility is pretty straight forward since it will use most of the technology provided by the C#.net framework. The storage will also be responsible for pulling data from the Web API server.

Services Provided:

1.
Service provided to: Schedule, Map
Description: A class that can handle various data storing and retrieving.
values: Allows us to change the backend of data manipulation without changing the rest of the code.

Services Required:
State

IV.2.2 Server

IV.2.2.1 Web API

(i) The web API must be able to listen on a socket and speak HTTP with clients connecting. The API will take requests from the clients and submit responses based on their inputs. Valid requests must be authenticated first and then the server will keep the state of that login.
(ii) The web API must keep track of sessions based on a key generated upon a login. Each key should have an expiration date so as to prevent spoofing, and to protect users in case their phone was lost, or stolen.
(iii) The web API will basically act as a medium between our app and the database to make sure users are only accessing the data they need. The API will need to allow users to "login" then request their schedule, a number of pre-concluded paths that would be determined busy depending on time of day, and other account settings that needs to persist past the user's current session.

Services Provided:

1.
Service provided to: State, Storage
Description: Central location of all application settings and logic.
values: Prevents the app from needing to connect to our database itself, which allows us to keep the data secure from anyone who may snoop the connection from our app to our database.

Services Required:

SQL Server

IV.2.2.2 SQL Database

(i) The data stored by the Web API will be stored into an SQL database. The database doesn't need to store any nodes, simply use information, and pre-determined routes. The number of student schedules stored to determine a busy route means our database needs to be quickly accessed and queried. We will use a MySQL database to store our information.
(ii) We went over different methods of storing data, but MySQL is open source, fast, and reliable.
(iii) The database will only communicate with the Web API part of the server. The app should never be able to connect straight to the server because that could lead to vulnerabilities in our security of our data.

Services Provided:

1.
Service provided to: Web API Server
Description: Database to store our data on user schedules.
values: Efficient matter of storing and retrieving data.

Services Required:

SQL Server

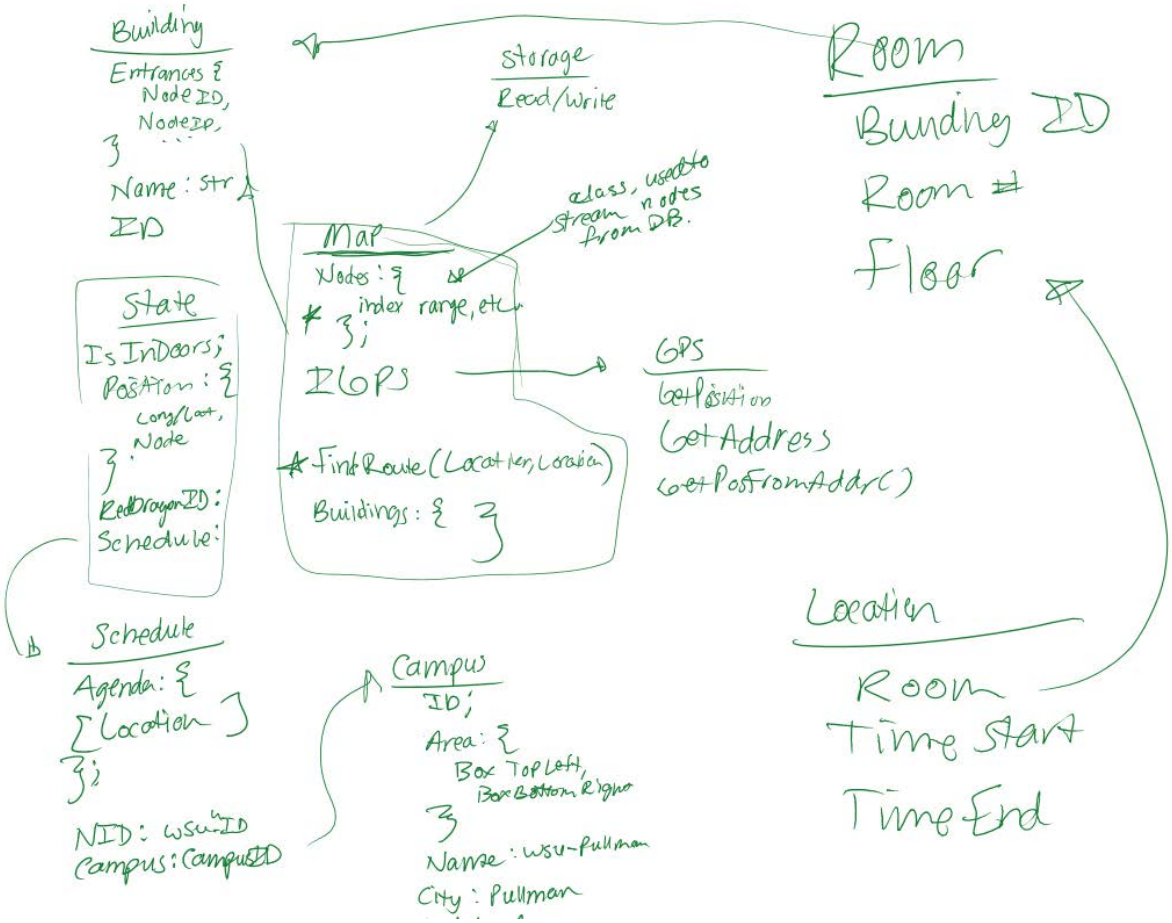


Figure IV.2.3 - Rough, Vague Class Diagram

IV.2. Data design

We decided upon a set of attributes for a node, being decimal:Longitude, decimal:Latitude, Neighbors (int:Up, int:down, int:left and int:right), decimal:Height, decimal:Traffic Percent, bool:Traversable, bool:Elevator, bool:Door, bool:Covered. A node will basically be a structure or a class with these attributes and Based on the size of each variable type in C#, 1 node will be 84 bytes. We will have about 237,980 nodes covering all of campus. This means our node database in plain form, will be about 19 megabytes. We have two potential approaches to storing and reading in our data for our application. We did the math and determined that our whole node network will be about 19mb in storage. We did research on the Windows, iOS, and Android phones available and found that the minimum available ram to a process across each platform was Android with 32-64mb of ram. This means that given the small amount of ram used by our actual application, we should be able to actually stream the entire grid into memory without any issue. The nodes will be stored into a text file as a binary representation of their values. When loaded, a new thread will be needed to stream-load in the nodes while the application does a short initial loading and then continue loading in the background so the user isn't waiting for the entire 19mb file to be read into memory. Our other option for loading in the

node database file, is to only load in what we need. So we would cover a 10x10 node grid and stream in the other surrounding nodes as we traverse further and further away from our origin point. This would be a better approach if devices turn out to not handle loading a 19mb file in well.

The other option is to use a sort of SQLite database where we use a relational database to store the data. This will prevent us from needing to load the whole thing into memory and speed up load times, it will however increase the time needed to search through the structure. So there are Pros and Cons to each. We will approach both solutions in our prototype to determine the best method to use for our final design.

IV.3. User Interface Design

The user interface, as with any other interface aims to keep a simple action and result display method. This will create a positive user experience on a mobile screen where the real estate is very limited. To create an atmosphere of simplicity, the use of large buttons and swipe animations is heavily relied upon.

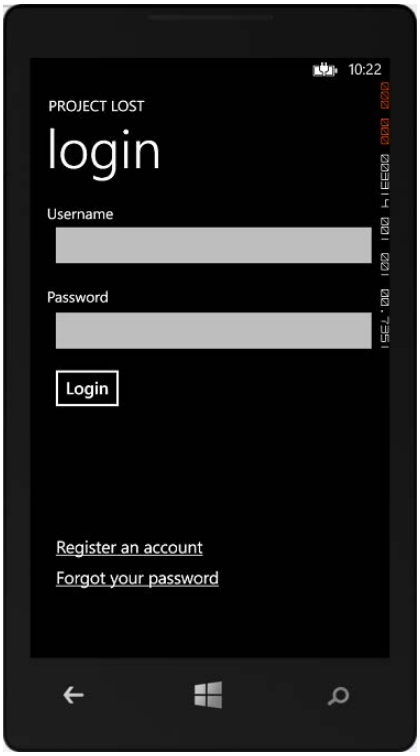


Figure IV.1 - Windows Phone Login Mockup

The app begins with a login form, prompting the user with a username and password field. Below the user will find a login button, a sign-up link, and forgotten password link. We expect the sign-up and forgotten password functionalities to be used less frequently, therefore, they are links instead of buttons. This is an acceptable design choice and is common practice in use of

many apps today (the Pandora app is an example of another app using this approach). If the user logs in with an incorrect username or password, the app will make an obvious display that they have entered their details improperly.

The login form on success will take them to their main form. The main form has the list of actions to perform such as navigating via their schedule, or set end point. Depending on the platform, Windows, iOS, or Android, the settings navigation button will be displayed. Windows phone will have it as a toolbar item on the bottom.

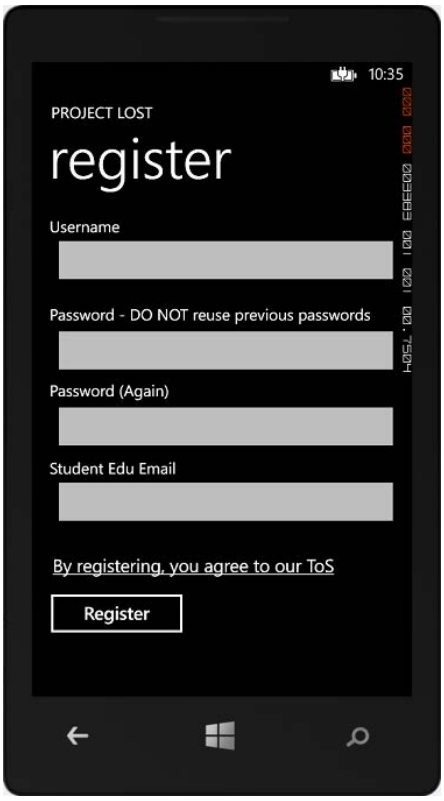


Figure IV.2 - Windows Phone Register Mockup

The registration link displays a form to prompt the user for an email and password. This form will clearly state that students are not encouraged to use the same password as any of their WSU passwords. Upon completion of this form, the server sends an email to the given email account verifying that account to be created belongs to the individual creating the account.

The forgotten password link will open a form to prompt the user for an email. This form sends notification to our server that sends will send a verification link to the corresponding email. Clicking the link will take the user to a webpage where they're password can be reset.

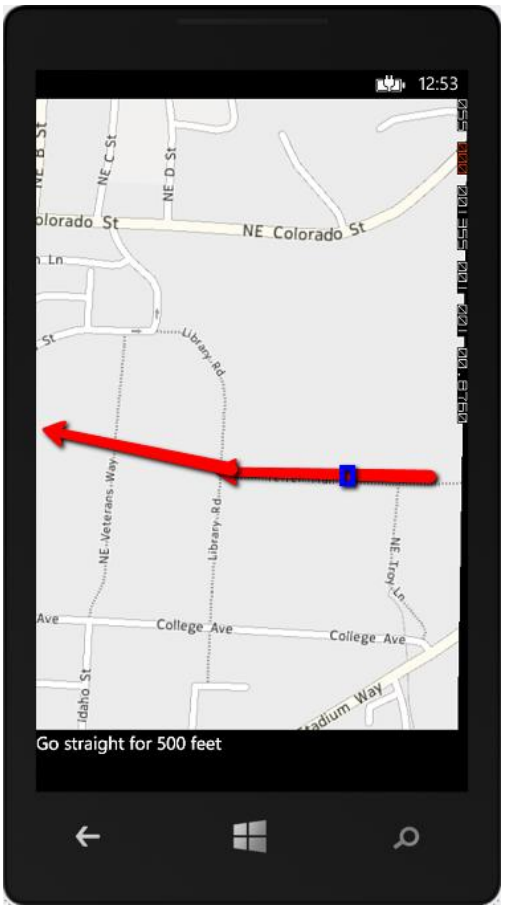


Figure IV.3 Windows Phone Map Mockup

The map interface will be in a form of its own. The map will have few, if any, navigational buttons. Key buttons here are the ‘return’ and ‘main’ button that allow the user to return to where they were previously or go straight back to the main form. This interface will also allow for scrolling as well as zooming in/out.

Setting destination manually simply opens a new form prompting the user to enter a building and room number. Completion of this form automatically takes the user to the navigation screen. From here the user can go back and make an adjustment or return directly to the main form.

The ‘settings’ button opens a form containing the settings and their adjustments. This portion of the user interface has not been set in stone yet. Potential settings are not yet concrete. The amount of settings implemented will determine the number of forms and layouts necessary to organize them.

V. Description of Final Prototype

The final prototype of all three of our phones handled the UI for information display and user manipulation. The figures below represent the UI for the Windows phone client. User interfaces for both iOS and Android follow this template almost exactly.

Upon execution of our application, the user would be greeted with the login screen displayed in figure 5.1. From here, the user could choose to log in via an already existing account, recover their password for an existing account, or create a new account.

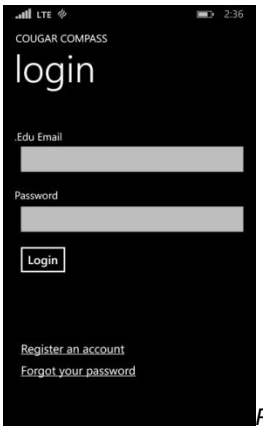


Figure V.1 Login Screen for the final prototype Windows Phone client

Assuming the user does not yet have an existing account, they will have to opt to create one in order to proceed. Clicking ‘Register an account’ on the login screen takes the user to the register screen seen in Figure 5.2

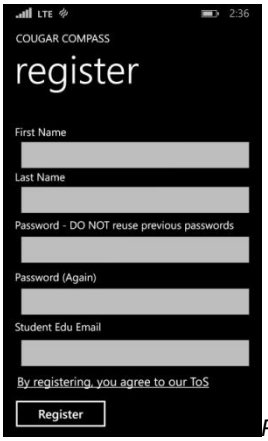


Figure V.2 Register Screen for the final prototype Windows Phone client

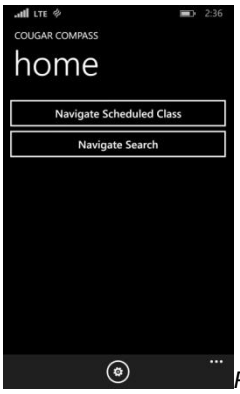


Figure V.3 Home screen for the final prototype Windows Phone client

Once the account is registered and has been verified via the given email account, the user can now successfully login. Upon login, the user will see a screen such as the one above in figure 5.3. As can be seen, the user can either navigate using their schedule of classes, or navigate manually by clicking ‘Navigate Search’.

Choosing navigate search will take the user to the below screen, figure 5.4, where they will be prompted for a start location. The prompt allows us to locate the start position even if the user happens to be indoors where GPS will not be as reliable as necessary to our applications success.

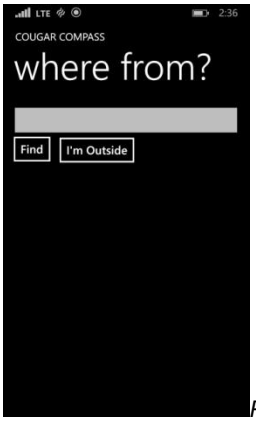


Figure V.4 Navigation Start Screen for the final prototype Windows Phone client

With the start location selected, the will have to select a destination to navigate to. This is done on the below screen, figure 5.5. In the example, the user has chosen ‘todd 220’ to be their destination. Pressing ‘Find’, searches for the building ‘todd’ in our building dictionary and displays the ‘official’ building string for the user to select.

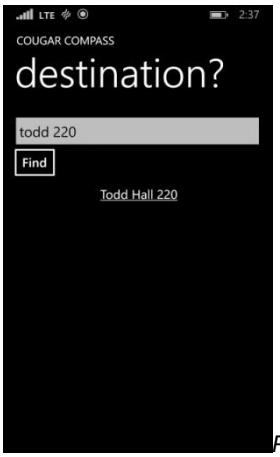


Figure V.5 Navigation end screen for the final prototype Windows Phone client

Having selected the building from the list of 'official' building dictionary strings, the application begins to load the building floor plan of their desired start location. Our modified A* algorithm then plots the line which successfully navigates the user from their start location through the floors to the appropriate exit or room. This is seen in figure 5.6

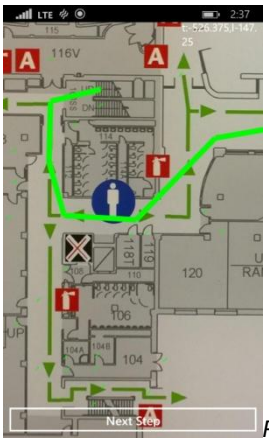


Figure V.6 Indoor Navigation for the final prototype Windows Phone client

Should the user's desired destination be located in another building, the application will load a satellite view of campus and plot the most appropriate path to the new building, upon the user's exit of the start location. The satellite view with a plotted path can be seen in figure 5.7.

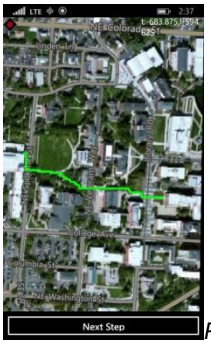


Figure V.7 Outdoor navigation for the final prototype Windows Phone client

If the desired destination is a specific room in the new building, the application will close the previously created satellite map and display the floor plans for navigating the new building as previously demonstrated in figure 5.6.

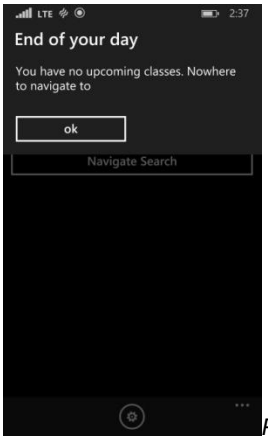


Figure V.8 End of day for the final prototype Windows Phone client

Automatically searching for rooms and buildings based on the user's schedule is similar to the manual search without the prompts for the user's input, making it more desirable if the user is not inclined to tediously input locations for source and destination. It also allows the user to be ignorant ahead of time about the locations of their classes. The only difference being that if the user is not responsible for attending any more classes for the day, the screen depicted in figure 5.8 will be shown.

VI. Results

VI.1. Prototype Test Results

We did not perform prototype tests excluding any live demo testing during our poster presentation.

VI.a. Unit Tests

Test Case Identifier: 1
Test Title/Name: API Call Login Success
Test Summary/Description: Testing the positive verification of login information on the server.
Assumptions: The user has already registered with the mobile application and verified email.
Input Specification: {“data”:{“email”:”demo@wsu.edu”,”pass”:”pass”}}
Output Specification:
    Expected output: {“result”:”success”,”api”:”Any 32 character alpha numeric key”}
    Actual Output: The expected output was the actual output.
Test Result (Pass/Fail): Pass
Pass/Fail Criteria: To pass and successfully verify the user’s login, the mobile application would receive an API key to further communicate with the server.

Test Case Identifier: 2
Test Title/Name: API Call Login Failure
Test Summary/Description: Testing the negative verification of login information on the server.
Assumptions: The user has not previously registered with the mobile application.
Input Specification: {“data”:{“email”:”demo@wsu.edu”,”pass”:”invalid pass”}}
Output Specification:
    Expected output: {“result”:”error”,”message”:”Invalid email or password”}
    Actual Output: The expected output was the actual output.
Test Result (Pass/Fail): Pass
Pass/Fail Criteria: To successfully return a negative verification the server would send the mobile application an error string indicating the error that occurred.

Test Case Identifier: 3
Test Title/Name: API Call Login While Activation Pending
Test Summary/Description: Upon registering, the user’s account should enter a state of pending activation. This should prevent the user from logging in without their credentials being verified via email.
Assumptions: The user has recently registered on the mobile application, but has failed to verify their account via email.
Input Specification: {“data”:{“email”:”demo@wsu.edu”,”pass”:”pass”}}

Output Specification:

Expected output: {“result”:”error”,”message”:”You must activate your account before signing in”}

Actual Output:
 PHP Fatal Error undefined offset 0 on variable $tok

Test Result (Pass/Fail): Fail

Pass/Fail Criteria: In order for the test to pass, the expected output string must be correctly returned from the server to the mobile application.

Test Case Identifier: 4

Test Title/Name: API Call Register Invalid Email

Test Summary/Description: During registration the user must use a valid WSU email address. The associated verification in the API must then result in a fail if anything but a WSU email is used to login.

Assumptions: This test case assumes that all other information sent to the server regarding this account is correct, and that the first validation test is the email validity test.

Input Specification: {“data”:{“email”:”invalid@email.notedu”, … other valid params}}

Output Specification:

Expected output: {“result”:”error”,”message”:”Invalid email specified.”}

Actual Output:{“result”:”error”,”message”:”Invalid email specified.”}

Test Result (Pass/Fail): Pass

Pass/Fail Criteria: The expected output should match the actual output

Test Case Identifier: 5

Test Title/Name: API Call Register Success

Test Summary/Description: Upon entering valid account credentials during registration, the API server should send an appropriate string to the mobile application, and a verification email to the user’s email.

Assumptions: The test case assumes that the email hasn’t been previously registered.

Input Specification: {“data”:{“email”:”valid@wsu.edu”,”pass”:”Passw0rd123!”, “first”:”David”,”last”:”Last”}}

Output Specification:

Expected output: {“result”:”success”,”message”:”You are now registered. Please check your email to verify your address”} and a verification email sent to the indicated email address.

Actual Output: {“result”:”success”,”message”:”You are now registered. Please check your email to verify your address”}, however no email was sent to the specified email address.

Test Result (Pass/Fail): Fail

Pass/Fail Criteria: Not only should a message be returned but an email must be sent for verification and activation of the account. Prior to the fix, no email would be sent from the server to the user’s email.

Test Case Identifier: 6

Test Title/Name: API Call empty parameters/arguments

Test Summary/Description: This test represents an API call with null parameters, and should not actually be possible via the mobile application.

Assumptions:
Input Specification: One or more null arguments passed into the API.
Output Specification:
Expected output: {“result”:”error”, “message”:”No valid data was received”}
Actual Output: {“result”:”error”, “message”:”No valid data was received”}
Test Result (Pass/Fail): Pass
Pass/Fail Criteria: For successful completion of the test, the API server must respond with an error preventing anyone with malicious intent from registering with

Test Case Identifier: 7
Test Title/Name: API Call blank
Test Summary/Description: No post data is sent to the API for any of the calls.
Assumptions: Any API command can be executed. Since all commands share the base string to object conversion code, a blank call should behave the same on every command.
Input Specification: “”
Output Specification:
Expected output: {“result”:”error”, “message”:”No valid data was received”}
Actual Output: {“result”:”error”, “message”:”No valid data was received”}
Test Result (Pass/Fail): Pass
Pass/Fail Criteria: The output should match the expected output.

Test Case Identifier: 8
Test Title/Name: API Call password reset valid email
Test Summary/Description: Resetting a password via email should reset their API key to be a reset API. An email is then sent.
Assumptions: An email and account exists, and the user has a valid API token.
Input Specification:
Output Specification:
Expected output: {“result”:”success”,”message”:”An email has been sent with a reset”}
Actual Output: {“result”:”success”,”message”:”An email has been sent with a reset”}
Test Result (Pass/Fail): Pass
Pass/Fail Criteria: Matching the expected output.

Test Case Identifier: 9
Test Title/Name: API Call Password reset invalid email
Test Summary/Description: When an email doesn’t exist for resetting a password.
Assumptions: No assumptions are required.
Input Specification: {“data”:{“email”:”email@notvalid.notedu”}}
Output Specification:
Expected output: {“result”;”error”,”message”:”Invalid email provided”}
Actual Output: {“result”;”error”,”message”:”Invalid email provided”}
Test Result (Pass/Fail): pass
Pass/Fail Criteria: The test case passes.

Test Case Identifier: 10
Test Title/Name: Levenhenstein Search Exact match
Test Summary/Description: When a search for a building is exactly in our dictionary.
Assumptions: the dictionary successfully loaded during bootup of the application.
Input Specification: building + room number
Output Specification: building + room number
Expected output: Sloan 151
Actual Output: Sloan 151
Test Result (Pass/Fail): pass
Pass/Fail Criteria: The test case passed.

Test Case Identifier: 11
Test Title/Name: Levenhenstein Search No Match
Test Summary/Description: Searching for something that has no match.
Assumptions: Database of words to search contains no words with z.
Input Specification: z
Output Specification:
Expected output: Results.Count == 0
Actual Output: Results.Count == 0
Test Result (Pass/Fail): Pass
Pass/Fail Criteria: When the result count is 0 like the expected output, the test passes.

Test Case Identifier: 12
Test Title/Name: Levenhenstein Search 1 character difference
Test Summary/Description: The user was searching for something, but the spelling was off by one character.
Assumptions: The dictionary has been loaded with the word to match against.
Input Specification: Slian 151
Output Specification:
Expected output: Sloan 151
Actual Output: Sloan 151
Test Result (Pass/Fail): Pass
Pass/Fail Criteria:

Test Case Identifier: 13
Test Title/Name: A* Search direct path
Test Summary/Description: Test whether or not it can navigate a straight line.
Assumptions: No assumptions are made.
Input Specification: To and From location, known result should be a straight line.
Output Specification:
Expected output: results show a straight line as the path found
Actual Output: results show a straight line as the path found.
Test Result (Pass/Fail): Pass

Pass/Fail Criteria: This test results in a line of nodes being connected.

Test Case Identifier: 14
Test Title/Name: A* Search boxed path
Test Summary/Description: Searching for a path from A to B, but A is in a boxed in area of non-traversable nodes.
Assumptions: none
Input Specification: To and From location, to being inside a known box, and from location being outside of starting box.
Output Specification:
Expected output: results with a count == 0
Actual Output: results returned null
Test Result (Pass/Fail): fail
Pass/Fail Criteria: This test case failed.

Test Case Identifier: 15
Test Title/Name: A* Search multiple paths with 1 path the only option
Test Summary/Description: When presented with multiple paths to take to get closer to the destination, the A* algorithm should pick the one with the closest route. that successfully chooses.
Assumptions: We assume that 3 paths are available and all 3 lead to the destination. Where 1 of the 3 might originally seem to go further out, it actually takes us closer than the other 2 routes.
Input Specification: <Grid of nodes>
Output Specification:
Expected output: Results.Count == 0
Actual Output: null
Test Result (Pass/Fail): fail
Pass/Fail Criteria: As in Unit Test 14, a null was returned when no result was returned.

Test Case Identifier: 16
Test Title/Name: A* Search Invalid node structure (No neighbors)
Test Summary/Description: When no neighbors are found for a node, the A* should return a results of 0.
Assumptions: We assume the node has no neighbors and could possibly happen.
Input Specification: Single node with no neighbors
Output Specification:
Expected output: Results.Count == 0
Actual Output: null
Test Result (Pass/Fail): Fail
Pass/Fail Criteria: Same issue as with unit test 15 and 14.

VI.b. Integration Tests

Test Case Identifier: 1
Test Title/Name: Manual Building Search
Test Summary/Description: Manually input string for building to be searched for in the building dictionary.
Assumptions: Test case assumes building exists in the building dictionary.
Input Specification: The name of a building on campus is entered. An example of such a query is “Sloan”.
Output Specification:
Expected output: The mobile application returns the “official” building string for the user’s confirmation.
Actual Output: The expected output was the actual output.
Test Result (Pass/Fail): Pass
Pass/Fail Criteria:

Test Case Identifier: 2
Test Title/Name: Manual Search floor # found
Test Summary/Description: When the Levenshtein formula matches the building string to a building structure yet no floor exists provided with the search.
Assumptions: The floor doesn’t exist.
Input Specification: “sloan 3”
Output Specification:
Expected output: “Sloan 3”
Actual Output: “Sloan G3”
Test Result (Pass/Fail): Fail
Pass/Fail Criteria: The algorithm detected the floor as a room number 3 instead of the floor.

Test Case Identifier: 3
Test Title/Name: Class search floor # not found
Test Summary/Description: When the Levenshtein formula matches the building string to a building structure yet no floor exists provided with the search.
Assumptions: The floor doesn’t exist.
Input Specification: “sloan 535”
Output Specification:
Expected output: “Sloan”
Actual Output: “Sloan”
Test Result (Pass/Fail): Pass
Pass/Fail Criteria: The proper building was selected with no floor since we don’t know what floor the user wants.

Test Case Identifier: 4
Test Title/Name: Class search room number not found

Test Summary/Description: Search for a room number that was not found.
Assumptions: We assume room 399 doesn't exist in sloan, we do assume the search is a valid floor.
Input Specification: Sloan 399
Output Specification:
Expected output: Sloan 3
Actual Output: Sloan 3
Test Result (Pass/Fail): pass
Pass/Fail Criteria: The expected output should match the actual output.

Test Case Identifier: 5
Test Title/Name: Class search room number found
Test Summary/Description: When a search for a room number is in our dictionary.
Assumptions: The dictionary is loaded and the room number is within it.
Input Specification: Sloan 153
Output Specification:
Expected output: Sloan 153
Actual Output: Sloan 153
Test Result (Pass/Fail): pass
Pass/Fail Criteria: The actual output should match the room number that the user typed in.

Test Case Identifier: 6
Test Title/Name: Class search building not found
Test Summary/Description: Search for a building that was not found within our database.
Assumptions: The building doesn't exist.
Input Specification: X
Output Specification:
Expected output: Not found, so zero results
Actual Output: Zero results
Test Result (Pass/Fail): pass
Pass/Fail Criteria: The search should return zero results, or a close building name found.

VI.c. User Acceptance Tests

Test Case Identifier: 1
Test Title/Name: Logging in
Test Summary/Description: Using the user interface of the iOS, Android, and Windows phone, log into the API.
Assumptions: We assume the inputs are valid and there are no UI Thread conditions and race conditions
Input Specification: The user logs in with their valid details.
Output Specification:
Expected output: App navigates to the next page without crashing.

Actual Output: App navigates to the next page without crashing.
Test Result (Pass/Fail): pass
Pass/Fail Criteria: The user must not wait too long and the page must navigate to the next page without issue.

Test Case Identifier: 2
Test Title/Name: Manual Building Search
Test Summary/Description: The user tests the ability to manually search for a building.
Assumptions:
Input Specification: The user enters a building and room number such as “Sloan 7”.
Output Specification:
Expected output: The mobile application returns the “official” building string as well as the room number entered.
Actual Output: The expected output was the actual output.
Test Result (Pass/Fail): Pass
Pass/Fail Criteria:

Test Case Identifier: 3
Test Title/Name: Navigate using the steps
Test Summary/Description: Can the user use the steps provided to accurately navigate the route displayed by the app.
Assumptions: We assume the user is testing this in the location they searched for.
Input Specification: From: Sloan 353; To: Sloan 151
Output Specification:
Expected output: A valid route is displayed and the user is able to trace the steps accurately.
Actual Output: The algorithm skips the 2nd floor.
Test Result (Pass/Fail): Fail.
Pass/Fail Criteria: The algorithm skipped the second floor thus causing slight confusion. This test case was repaired and passed upon another test.

Test Case Identifier: 4
Test Title/Name: Verify personal schedule to results from API
Test Summary/Description: Verify the results of the schedule the API returns with a known set of classes.
Assumptions: None.
Input Specification: beau_schwieso (network ID)
Output Specification:
Expected output: List of classes for the Spring 2014 semester.
Actual Output: List of classes for the Spring 2014 semester.
Test Result (Pass/Fail): pass
Pass/Fail Criteria: If the API returns the correct amount and list of classes for the given network ID.

VI.d. Performance Tests

Test Case Identifier: 1
Test Title/Name: Web API
Test Summary/Description: Accessing the web API once proves it works but accessing thousands of time in a row shows scalability.
Assumptions: We assume there are no bandwidth latency issues.
Input Specification: List of 1,000 usernames and passwords that are all valid with the login post request to the api.
Output Specification:
Expected output: API should return with minimal delay
Actual Output: API returns in a somewhat timely manner.
Test Result (Pass/Fail): pass
Pass/Fail Criteria: The API should return 1,000 results within 5 minutes. It took 2.5 minutes.

Test Case Identifier: 2
Test Title/Name: A* Search
Test Summary/Description: A search on the campus shouldn't take more than 10 seconds on a phone device.
Assumptions: We assume we have 2 points to travel between.
Input Specification: Node Id: 122451, 345012
Output Specification:
Expected output: A path is found between points. The algorithm executes in less than 10 seconds.
Actual Output: A path is found between points. The algorithm executes in less than 10 seconds (1.5 seconds)
Test Result (Pass/Fail): pass
Pass/Fail Criteria: The algorithm completes a path under the maximum time allotted.

VI.e. Test Cases Omitted:

The test cases that we omitted for this project involved testing navigation on the Android phone, testing logging in using the Android phone, performing a stress test on our API server to determine a range of users that could be logged in at once, and determining any latency issues that would result from multiple users accessing the server at once.

We did not test logging into the Android phone, because the programming was the same for both Windows Phone and iPhone. We believe that this would've been redundant to test, and therefore didn't duplicate our testing on the Android phone if the code hadn't changed. The same reasoning was used for all Android development, with the exception of loading images (our inside map), and calling the separate screens (the graphical user interface).

The decision to omit both testing the range of users on the server and the latency issues involved with that was due to the lack of available resources to properly stress test our server. The lack of resources was due to the lack of available personnel to use our application, because the application wasn't ready for public use until the week of the poster session, and the lack of available devices to use, because of licensing restrictions making it required to release to the app store to release to more than five phones for the iPhone.

VI.2. Additional Results

There were no additional tests that we performed on our final prototype.

VII. Limitations and Recommendations:

Include a brief description of the limitations of your current prototype that you are aware of, and discuss possible solution approaches to overcome those limitations.
Please keep the discussion brief. 1 page text should be sufficient for this section.
Note that there might be overlap between the Limitations and Recommendations section and the Future Work section. "Future Work" should discuss the possible ways to extend the project, whereas "Limitations and Recommendations" should be more detailed and should discuss the problems and missing features in the final prototype and should describe the possible solution approaches.

VIII. Conclusions and Future Work

Overall, Team Red Dragon was able to accomplish the majority of our project goals, and to complete the application as we intended. There are enhancements we would make with more time, and with a sponsorship from the university for blueprints and clearer information from the school we would be able to produce a better user interface. However, the application navigates with clearly identifiable information that is familiar to students, and we heard a lot of buzz about how useful the product is especially to those just getting used to the campus. We are proud of the application that we produced, and for the potential the application has with further development.

With the second semester coming to a close, our project reflection focuses on not only what we have done well, but where the project could have gone as well. There are some secondary and tertiary goals that we would have liked to complete, given more time and resources to work on this project. These updates highlight the true potential of a campus navigation app for WSU and other campuses across the nation.

Given more time, we would like to expand the data entry to include all buildings and entrances on campus. Currently, the app is not ready to be listed for purchase or use, but with more data entry complete, we would be able to deploy this to the Windows, Apple, and Android app stores. The remaining data entry includes entering nodes for building floors associating each door with an outside node to tie the two grids of nodes together. This inclusion of new data would allow for a complete application and would allow for more buildings to be searched

Another feature we would implement in the future is to give more than one path for a given destination. For example, we could allow users to chose the fasted versus slowest route, most or least populated route, or even the path that stays indoors for longest. This would increase the usability of our application as well as give the user the option to avoid a given path based on that day's events (i.e. giving them a route to use that would include using a path that is blocked due to an unknown event, or the fastest if they are running short on time).

When designing our application, we kept it in mind to make it very easy to transfer this technology to another campus or location. To deploy at a separate campus, the entry nodes would need to be updated to reflect eh new layout, but the patching and new development would be minimal. It was our goal to expand this to include not only other WSU campuses, but to custom make this application to fit other University campuses as well. Keeping the code clear and logical makes the transfer easy.

Additional time on this project, along with the support of providing us actual blueprints of the buildings on the Washington State campus would allow us to increase the graphical user interface by allowing us to show a 3-Dimensional result for the indoor path. This update is important because it changes the user interface to a view consistent with other routing software that smart phone users are used to. Although it may seem trivial, familiarity increases the brand

support and backing of our users, making the application easier to understand, and more successful.

We would like to expand our current path results interface to dynamically change to the next result using the current location of the user and to dynamically detour the path if the user deviates from the proposed route. This would expand on the goal to be able to find the user's location for both indoor and outdoor without forcing the user to give us the location. Again, this is a feature supported by other routing applications, and is a feature our users would likely expect.

Our application has a high potential to be bought out from another company or by the University, because of the real world need it has. We had a significant percentage of people ask us when this would be available to the public based on showing them a live demonstration.

Finally, we would go back and change our data structure to include a separate class for marking indoor and outdoor nodes. The current implementation has an integer for marking a door, zero representing a non-door. If the field is non-zero, we have to then parse that integer to figure out whether or not it points to an outdoor node (indoor to outdoor navigation) or if it points to another building (indoor to indoor navigation). With the changes making into another class, it would reduce processing power requirements severely.

IX. Acknowledgements

Thank you, Andy O'Fallon for all your help and guidance for this project.

X. References

This document does not have any external references.

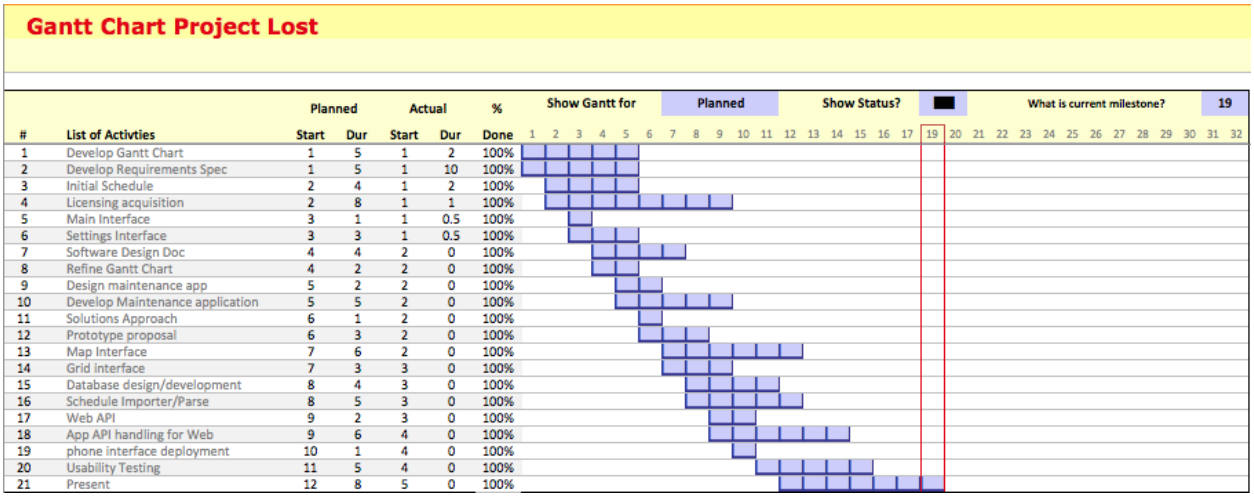
XI. Appendix A – Team Information

Team Members: Beau Schwieso, David Barajas, Joey Smith

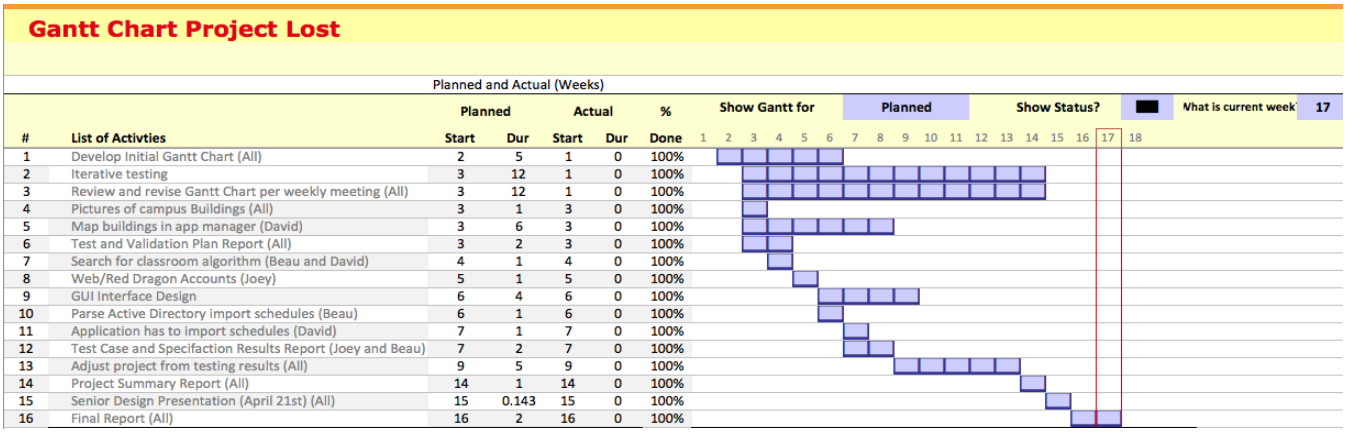
Team Picture:

XII. Appendix B - Project Management

Each week, our team met with the instructor to give her a progress of the previous week's tasks. We also would compare the tasks and goals listed on our current Gantt chart. Outside of our weekly meeting with our professor, we would meet bi-weekly as a team to discuss any progress that would require any implementation changes or answer any questions to give clarification to a specific task and it's requirements. The bi-weekly meetings as a team were very beneficial to our team and kept us on task for each milestone to be completed on time. It was also very beneficial because we were able to have a feedback loop on each task at a rapid pace.



Gantt Chart for semester 1. Produced at week 1 of semester 1.



Gantt Chart for semester 2. Produced at week 1 of semester 2.