Analyzing the Strength of Undergraduate Misconceptions about Software Engineering

Leigh Ann Sudol
Computer Science Department
Carnegie Mellon University
Pittsburgh, PA
leighann@cmu.edu

Ciera Jaspan
Institute for Software Research
Carnegie Mellon University
Pittsburgh, PA
ciera@cmu.edu

ABSTRACT

While many computer science students plan to pursue careers as software engineers, research shows that most traditional undergraduate CS programs fail to prepare students for the realities of programming in industry. Many misconceptions that are interfering with the transition to industry are belief-oriented, not skill-oriented, in nature, so traditional misconception assessments will not yield a deep understanding of them.

In this paper we present a novel methodology that shows interactions among the misconceptions based on a forced choice paradigm and reveals the relative strength of the misconceptions. By analyzing students' repeated responses and response times, we construct a model of participants' misconceptions. We used this methodology to assess CS undergraduates at Carnegie Mellon University and compared their results to those from industry practitioners at several highly regarded companies. The results show that the students have misconceptions about process and teamwork. Surprisingly, we found that several misconceptions are correlated with elective courses that we expected to weaken misconceptions about software engineering but instead appeared to strengthen them.

Categories and Subject Descriptors

K.3.2 [**Computers & Education**]: Computer and Information Sciences Education; D.2 [**Software**]: Software Engineering

General Terms

Human Factors

Keywords

Misconceptions, Software Engineering Education

ICER'10, Aarhus, Denmark

1. INTRODUCTION

While many computer science students enter software engineering jobs after graduation, prior research has shown that students are not prepared for industry positions. They lack the "soft skills" necessary for success in a team environment [1], they do not have the knowledge and skills for design and maintenance tasks [3], and new developers are often surprised to find that these skills are essential for their professional career [17]. Educators should be addressing this problem particularly because corporations are spending time and resources to remediate new developers with these skills [2, 3].

In this work, we show that students hold many misconceptions about industry practices and the expected responsibilities of software engineers. For example, based on a traditional CS education, a student may believe that software engineers spend most of their time developing new algorithms and data structures. A student might also believe, based on unproductive experiences with team projects in class, that teams are an ineffective way to approach a programming task. Such beliefs are dangerous to students because, in addition to being ill-prepared for industry, they may not actively seek out the skills and knowledge needed to succeed in a professional career.

In order to address such problematic misconceptions, educators must first be able to discover what misconceptions students hold and how strongly they hold them. This work contributes to this goal with:

- a forced-choice survey tool that facilitates educators' understanding of student misconceptions (Section 3) and confirms the expected misconceptions as compared to industry beliefs (Section 3.4),
- a macro-level analysis to correlate misconceptions to population demographics and to other misconceptions (Section 4.1), and
- a micro-level analysis to understand the strength of a student's (or sub-population's) misconceptions (Section 4.3).

We conducted our survey with 115 CS undergraduates at Carnegie Mellon University, and we validated the misconceptions with 45 industry practitioners taking the same survey. The analysis showed several interesting correlations, both positive and negative, between software engineering misconceptions and the age of the participant, whether the participant had an internship, and which CS courses the participant took.

2. MISCONCEPTIONS

The adage that students "know it or they don't" suggestions that either students "have" a particular piece of knowledge, or they have no conception of the topic. Research in the education sciences suggests that there is perhaps a third state, misconception [10, 13, 14]. Studies in a variety of domains such as general science [4, 18], mathematics, and physics [7] have shown that students often have complex, if incorrect, models of the world that they develop without formalized instruction.

2.1 Conceptual Change Theory

Conceptual change theories investigate the way in which naive knowledge is transformed into correct or more sophisticated knowledge in students [13]. Understanding misconceptions of novices is important if we want to promote efficient conceptual change through our preparatory programs [4, 6, 11, 14, 15]. Research has shown that in many cases, when presented with correct information, novices will maintain both the correct knowledge, in addition to the previously held misconceptions [5, 6, 9, 18]. The misconception must be activated as a part of the learning process in order to promote viable conceptual change [4, 5, 6, 14].

The classic example of students showing poor conceptual change after instruction is the motivation behind the physics force concept inventory [13]. Even though students were able to apply formulas and give correct definitions for concepts on exams, they often were unable to apply the same knowledge in similar situations. Only by understanding the misconceptions that students held, and addressing them explicitly are professors able to promote conceptual change of students' mental model of force.

2.2 Misconceptions in Software Engineering

Work in computer science education has shown us that the current model of software engineering education is not dispelling the misconceptions that students hold [1, 2, 12, 17]. These misconceptions have such a negative impact on the performance of software engineers that major corporations have had to implement their own in-house training programs in order to re-educate their employees. In 2002 Microsoft halted all operations on the Windows Operating System to train all 8,000 of their developers in software processes [3]. This type of intervention is not only time consuming but costly. Businesses are also calling for students to have a skill set that includes communication and process skills as well as traditional theory [17].

Although there is a significant body of research in software engineering education, few papers have taken the approach of understanding the misconceptions that students have prior to instruction. In particular, there is little work that describes the misconceptions undergraduate students have at varying points in their educational progression, and what experiences correlate with those misconceptions. In this work we look at undergraduates in varying years of their degree, as well as what courses they have completed, to see if any significant correlations occur.

We believed that we would see that students who have taken courses that involve a large software project would show less misconceptions than students who did not. The act of working in groups in project-based courses should help expose students to the knowledge needed to dispel any misconceptions they have. Also, courses such as software engineering explicitly teach the importance of these skills, so we would expect to see little to no misconceptions in the students who have completed these courses. Finally, we expected few misconceptions from students who had completed internships, since these students may have already resolved any misconceptions during the internship.

3. METHODOLOGY

In order to confirm our hypotheses about student misconceptions in software engineering, we constructed a list of misconceptions with the help of the faculty and graduate students in our institute, as shown in Table 1. We then used these misconceptions in a survey to assess how strongly they are held by students. In this section, we describe the survey methodology, our study population, how we selected the misconceptions for the survey, and our validation of the misconception statements on industry practitioners.

3.1 Instrumentation

Declarative knowledge represents a person's model of the world [6]. Although the presentation of refutational information is desired to promote a change in beliefs or understanding, it has been shown that just presenting the new information will not cause a change in belief [8]. The implications of this are that we need to understand what students' misconceptions are, and how strongly they hold them.

Previous measures of misconceptions focused on either the self report of the participants [12, 18] or on their ability to problem solve in an assessment situation [4, 9]. While these assessments could tell us what misconceptions students had, it would be difficult to evaluate the comparative strength of those misconceptions.

To measure the strength of participants' misconceptions, we used a forced-choice survey. This survey presents participants with a random misconception from Table 1 alongside the *valid inverse of a different misconception.* The survey then asks participants to select the statement which is "more true", and it records their selection and the response time. The survey shows each misconception three times alongside a different valid inverse statement, thus participants evaluate each misconception *six* times (three times as a misconception and three times as a valid inverse). Figure 1 shows a screen shot of one survey question. The survey randomly swaps whether the misconception is on the left or the right, however, all participants see the same pairs in the same order.

Pairing elements in this way and timing the responses allows us to analyze how strongly students held their beliefs and rank the misconceptions based upon the participant's selections. Responses with a short decision time indicated that the participant was able to make a quick choice between the two options, clearly confident in his answer. This indicates to us either a strongly held misconception or a confidence in the inverse. Longer decision times indicate that the participant needed to consider the two options before making a selection. This indicates the participant did not have a strong feeling about either statement, or that he felt strongly about both statements and took time to decide.

3.2 Subjects

The student participants were recruited through an email sent by the Associate Dean of the school to all CS undergraduates. As an incentive for the students to take the survey,

#	Misconception	Inverse
1	A defined software process is only important when you are working with people who are less skilled.	A defined software process is important for any team of programmers, regardless of skill.
2	A good software developer will often choose to work alone on a project in order to get it done faster.	A good software developer will often choose to work in a team in order to finish a project faster.
3	When you have a team of good programmers who work well together, a software process will usually get in the way.	When you have a team of good programmers who work well together, a software process can increase their ability to produce quality software.
4	My code should take advantage of the implementation details in other code.	My code should rely only on the specification of other code.
5	It is expected that clients will describe their requirements accurately before a team begins programming.	It is expected that clients will reevaluate and change requirements while a team is programming.
6	As a software developer, most of my time will be spent designing and implementing new algorithms and data structures.	As a software developer, most of my time will be spent determining how my code should interact with other people's code.
7	Most of the time when I start a new programming task in industry, I will be working on a new project.	Most of the time when I start a new programming task in industry, I will be modifying existing code.
8	Developers do not need to know the high-level context of the system; this allows them to concentrate on their task.	Developers need to know the high-level context of the system in addition to the details for their task.
9	A software project is successful only if it ships with very few known defects.	A successful software project can be shipped with many known defects.
10	Software engineering is about producing lots of documentation on the requirements and implementation of the project.	Software engineering is about how to produce software in an efficient and cost-effective manner.
11	Process, requirements, and team-management are important to business majors, not software developers.	Process, requirements, and team-management are just as important to software developers as programming is.
12	The majority of the cost of a successful software project will be the initial implementation effort.	The majority of the cost of a successful software project will occur after its initial deployment.

Table 1: List of Misconceptions and Inverse Statements

we held a raffle for two iPod shuffles for the participants. We had a 22% response rate; 115 undergraduate computer science students participated in the survey. The survey was taken by 34 first years, 36 sophomores, 31 juniors, 29 seniors, and 1 student who selected "Not applicable" in response to that question.

3.3 Construction of Misconception list

We studied 12 misconceptions in this work, as seen in Table 1. We selected these misconceptions from a longer list that was generated by the graduate students and faculty in the Institute for Software Research at CMU. The original list contained misconceptions that the people from our institute had either seen evidence of in students or had suspected existed in students. In some cases, the misconceptions were ones which people from our institute had held themselves while they were studying.

We shortened the original list by selecting misconceptions that were the least controversial and the most obvious misconceptions about industry. We aimed to select misconceptions that were the least likely to have exceptional cases; it should not be easy to find cases where the misconception is actually true in industry. We also chose to select misconceptions from a wide range of topics that all undergraduates would be familiar with.

This process has the risk that people from our institute may have their own misconceptions and pollute the list with truths, rather than misconceptions. To mitigate this risk, we evaluated the 12 misconceptions with industry professionals, as described in Section 3.4. Our evaluation shows that the industry professionals agree that the statements in Table 1 are indeed misconceptions.

3.4 Validation on Industry

To validate that the misconceptions used in the survey are indeed beliefs that are not held in industry, we also ran the survey on 45 industry professionals. The professionals were contacted through the authors' professional networks, so there does exist some bias towards the authors' beliefs. In particular, 78% of the respondents attended one of the same universities as the authors. However, the professionals are from several well known software companies; the top companies represented are Google and Microsoft with eight respondents each. Other companies with at least 2 respondents are listed in Table 2. They have an average of 7 years of experience, with a minimum of 1 year and a maximum of 21 years. 71% are currently software developers; Table 3 lists other positions (percentages do not add to 100% because the participants could select multiple positions). Given the years of experience and the type of companies represented, we believe that while this is not a random sampling of industry professionals, it is a fair sampling of *successful* industry professionals.

The industry professionals strongly agreed with the valid inverse statements listings in Table 1. On average, they disagreed with 1.84 of the 36 pairs that they saw. There were also three outliers in the group which disagreed with our "correct" response; two disagreed 8 times and one disagreed 9 times. Upon removing these outliers, the industry practitioners disagreed on an average of 1.38 of the 36 pairs. No

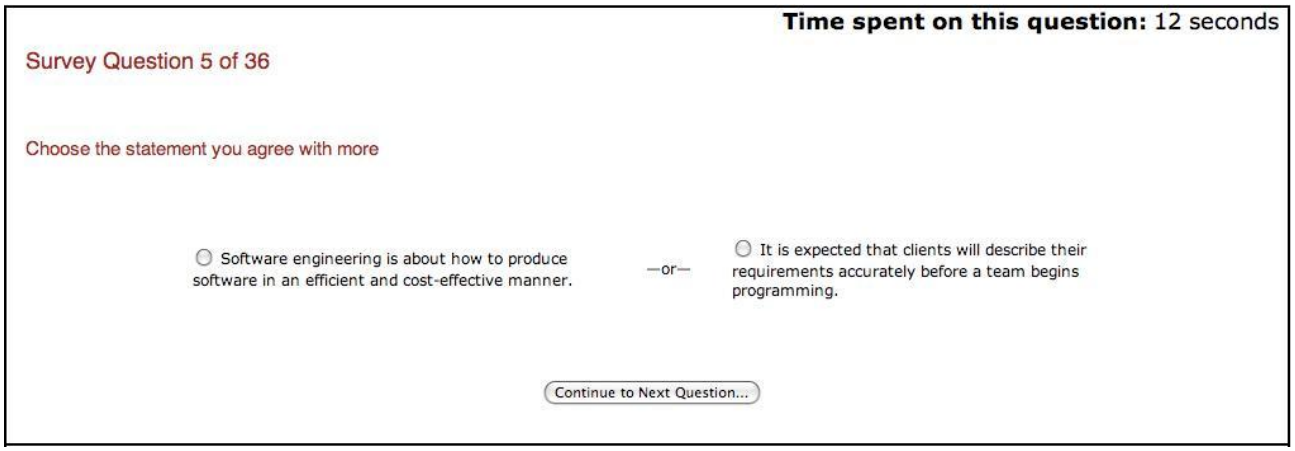

Figure 1: Screen Capture of Survey

single misconception stuck out as an invalid misconception; the disagreements were distributed across all statements. This low number and good distribution strongly validates that the misconception statements are truly misconceptions.

Amazon	Jambool
Expedia	Microsoft
Google	Rosetta Stone
Green Hills Software	SRI International

Table 2: Industry professional affiliations with two or more respondents

Position	% Respondents
Developer	71%
Tester	9%
Developer in Test	13%
Project/Program Manager	18%
Manager(of people)	16%
Tech Lead/Developer Lead	38%
Software Architect	13%
Reliability/Operations Engineer	7%
Other	11%

Table 3: Current position of industry respondents (multiple selections allowed)

4. STUDENT RESULTS

There are two particular layers of information that can be useful to educators and researchers when dealing with misconceptions. At the program, or macro level, it needs to be understood what general misconceptions emerge and what parts of the degree program either eliminate or encourage those misconceptions. At the student, or micro level, it is important to understand what individual misconceptions are held and how strong they are. This section presents analysis of our data at both the macro and micro levels.

4.1 Results

The responses to the assessment indicate that students do hold ideas about software engineering that are different from those held by professionals. Figure 2 compares these populations by the total number of responses that signal a misconception. While the industry population had an mean of 2.02 "incorrect" responses out of 36, students had significantly more misconceptions with an mean of 6.8 out of 36 ($t = -5.503$, $df = 154$, $p<.0001$).

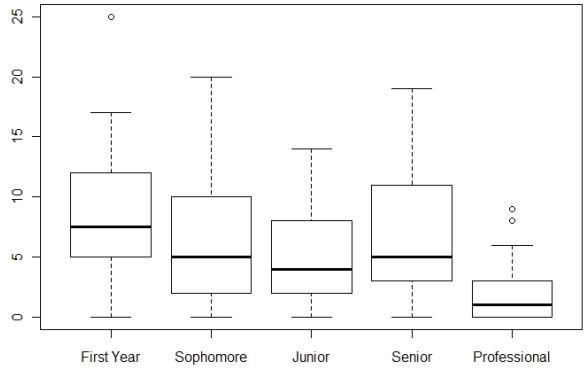

Figure 2: Professional vs. Undergrad Misconception Counts

We correlated the propensity of a student to hold a misconception with several demographics. The following demographics, shown in the first column of Table 4, showed interesting correlations to the number of incorrect responses.

Year. Students were asked to select their year in school, from the options of First Year, Sophomore, Junior, or Senior. We had an even distribution of students across years.

Age. Students were asked to provide their age.

Intern. Students were asked how many internships they had. For purposes of correlation model, we only considered whether or not the student had an internship.

Male. Students were asked their gender. For purposes of the correlation model, we considered male as a positive bit. Our proportion of male and female respondents is representative of the population.

Courses. Students were asked whether they had taken

any courses from a list of possible upper-division courses. Software Engineering, Operating Systems, and Compilers are all lecture-based introductions to their respective topics. Each of these courses also has a project component, and all three courses are optional but count towards a requirement. Operating Systems Practical, Software Engineering Practical, and Web Applications are optional, project-based electives.

Variable	N	Direction	p-value
Year	N/A	-	< 0.001
Age	N/A	-	< 0.001
Intern	67	-	< 0.001
Male	96	-	0.09
Software Engineering	5	-	< 0.001
Operating Systems	30	NS	0.28
Compilers	4	NS	0.32
Operating Systems Prac.	2	-	0.01
Software Engineering Prac.	2	-	0.001
Web Applications	13	+	0.004

Table 4: Correlation of demographics with misconception counts. N describes the number of students in the sub-group. Direction indicates (+) for correlation with a higher number of misconceptions, (-) for correlation with a fewer number of misconceptions, and (NS) when the correlation is not significant. The final column displays the p-value for the correlation.

As anticipated, students hold fewer misconceptions as their year and age increases. However, there was a slight up tick in the number of misconceptions held by seniors compared to their junior counterparts, as shown in Figure 2. The number of internships also displays this trend; misconceptions generally go down with number of internships, with the exception of students who reported 4 or more internships. This trend is clearly correlated, as only senior students could have so many internships. The cause of this trend is not clear; there have been no changes to the curriculum in many years. This trend will be discussed further in Section 5.

The correlations with courses held a few surprises. As expected, students that have taken the software engineering courses held fewer misconceptions, though very few of those students participated in the survey. We also found that compilers and operating systems had no significant correlation. However, we were surprised to find that web applications had a significantly positive correlation with the number of misconceptions. This course is heavily project intensive, the faculty have worked in industry, and students are required to work in group projects. As Web Applications is normally taken by juniors or seniors, this may again be correlating with the age of the participant, or it could be due to self-selection into the course. Possible hypotheses and future work to investigate this, and related trends, are discussed in Section 5.

4.2 Particular Misconceptions

In addition to analyzing individual students, we also analyzed individual misconceptions. We analyzed both the strength to which students held the misconception and the demographic variables that correlated with the strength of misconceptions.

For each misconception, we calculated a student's strength of belief in that misconception on a scale from 0-6. This number was based on the number of incorrect responses when seeing either the misconception or the valid inverse. For example, if a student never selected the misconception and always selected the valid inverse, the student has a strength of 0 for that misconception. However, if a student always selects the misconception and never selects the valid inverse, the student has a strength of 6. This allows us to distinguish between students who strongly hold a misconception, and those who chose it only once, which could indicate that the deciding factor in the selection was the opposing statement.

Figure 3 displays the percentage of students who held the misconception with varying strengths. For readability, we have divided students into four groups based upon misconception strength. Two misconceptions, 4 and 9, particularly stand out from the others. Misconception 9 makes sense based on most students experiences; they are likely not used to thinking about releasing software with known defects as most of their class assignments strive for perfection. However misconception 4 is very concerning; students are supposed to be familiar with the benefits of abstraction and interfaces to code, rather than using implementations directly.

#	Topic	Correlating Features
2	Choose to work alone	Year in School (-**) Operating Systems (+.) Compilers (+.)
4	Implementation details	Year in School (-*) Internship (-**) Gender (-**) Software Engineering (-*) Web Applications (+*)
5	Clients accurately describe requirements	Year in School (-*) Age (-.) Operating Systems (+*) Web Applications (+.)
8	High-level context	Year in School (-**) Web Applications (+*)
10	Producing lots of documentation	Year (-*) Internship (-**) Compilers (+*) Web Applications (+*)

Table 5: Interesting Misconceptions and Correlating Features. Correlation is shown as either positive (+) or negative (-). The last column also shows whether the feature is loosely correlated (. = p<0.1), moderately correlated (* = p<.05), or strongly correlated (= p<.01).**

Each misconception had one or more correlating factors associated with it. Table 5 contains several interesting misconceptions and the features that correlated significantly with those misconceptions. One particular misconception that can be considered to be dangerous is that a good software engineer will choose to work alone on a project in order to get it done faster. This misconception will directly affect the way that a future professional will interact with coworkers, and as shown by Begel and Simon[1] it can seriously hinder an engineer's productivity. While the correlation indicates that the longer students are in the program, the less

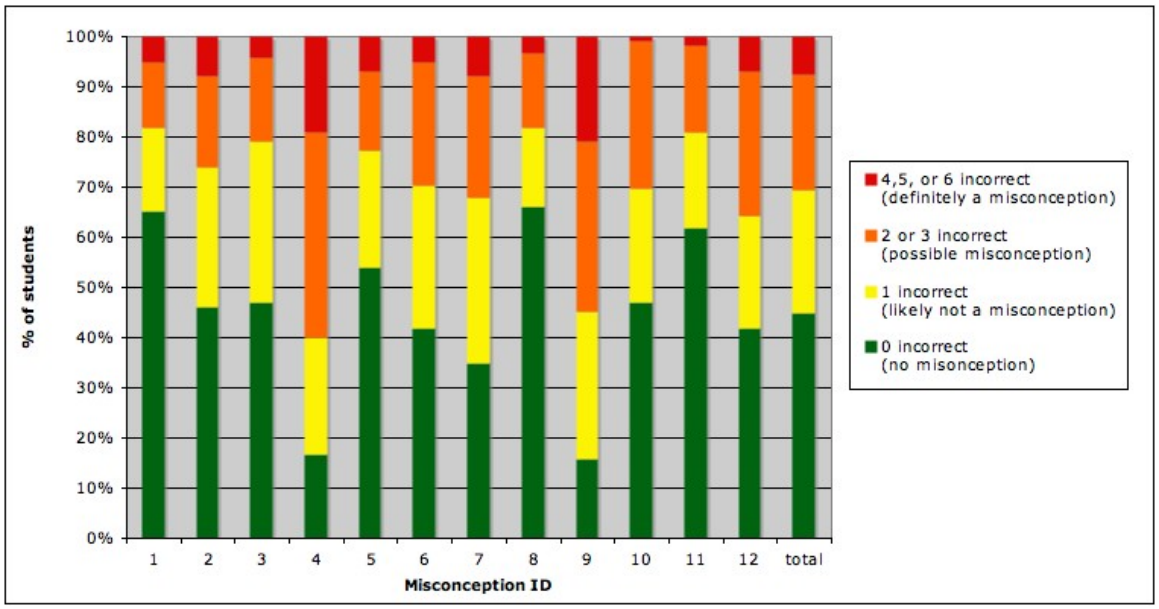

Figure 3: Strength of Misconceptions

likely they are to have this misconception, the positive correlation with two courses, and the absence of significance with the software engineering course (which is supposed to teach process) is interesting.

A complete listing of correlational features for all misconceptions can be found in the technical report for this work [16].

4.3 Individual Student Misconception Maps

The assessment used also presents us with opportunities to visualize the misconceptions of an individual student or subgroup in an interesting and informative way. The misconception map exposes information about the choices made as well as factors surrounding the choice. Although the map could be used for subsets of the participants of any size, we have chosen to highlight individual students here.

The misconception map plots the amount of time participants took to select an option in the forced-choice survey (see Figures 4 and 5). The horizontal axis represents time, and has a correct and incorrect region. This allows researchers to see correct choices on the right and incorrect choices on the left. In addition the character used to plot each choice is representative of whether statement was shown as a misconception or inverse. This visualization offers several cues to the strength of a misconception indicated by a student.

First, it is easy to see if the student was consistent in the choice of a misconception or its inverse. You can see from Figure 4 that the participant chose incorrectly every time they were presented with statement 6. This is an important cue because not only did the student choose statement 9 in its misconception form, he also rejected the inverse of the statement when presented with it. Statement 10 had the exact opposite result, with the student choosing the inverse and rejecting the misconception every time. This behavior would indicate strongly held beliefs about these two statements.

The second cue of strength offered from the misconception map is the time that it takes for a student to decide

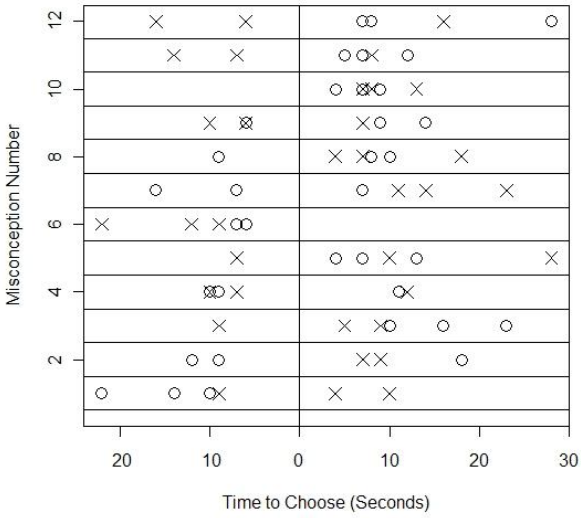

Figure 4: Student 96 Misconception Map

between the choices presented. In Figure 5, the responses for both the misconception and inverse of statement 4 are consistent and relatively quick. This would indicate that the student did not need to take time to weigh the two options presented, and instead responded mostly with their belief about statement 4. In contrast, The responses for statement 7 are spread in terms of both time and correctness. This would indicate that the student had no strong feelings about statement 7, or it was more important to think about how statement 7 compared to what it was shown against. Notice that there is a similar pattern for statement 10 as well. This additional information offers insight into the decisions of the students as well as the strength of their beliefs.

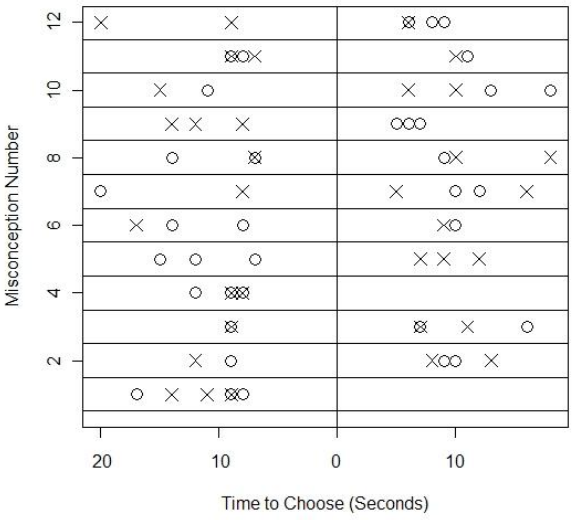

Figure 5: Student 159 Misconception Map

Inconsistency combined with long decision times would suggest a weak attachment to the misconception, while consistency and quick decisions point to strong misconceptions.

The third cue offered by the map is the character used to plot the decision. Each choice involved a misconception statement, and an inverse from another statement. Therefore each choice is represented by two points on this graph, one x and one o. The x's represent the misconception statement, and the o's represent the inverse statement. If an x is plotted on the left of the center line, it means that the student chose that misconception. If an o is plotted on the left, it means that the student rejected the inverse (by choosing another misconception). These are both incorrect actions.

The right side of the graph shows correct actions, and so an o on the right side indicates the selection of a correct inverse, while an x on the right indicates the rejection of a misconception. As we were analyzing the data we ran into an interesting phenomenon which can be seen in Figure 5 for statements 5 and 9. Notice that for statement 5 the student correctly rejected the misconception every time they saw it (x's on the right), however also rejected the inverse every time they saw it (o's on the left). In statement 9 we had the opposite behavior; the student correctly selected the inverse every time that they saw it (o's on the right), and did so fairly quickly. They also incorrectly selected the misconception every time they saw it (x's on the left). This behavior is very interesting, and could indicate several things. Possibly, it could show that the student is in a state of transition between their old beliefs and new beliefs being fostered.

In addition to visualizing this data at the student level, the visualization also assists with evaluating at higher levels. For example, we can graph a group of students with similar demographics on a single misconception map to see trends and patterns that are not visible from simple misconception counts. This information could be useful for assessing and reforming curriculums.

5. DISCUSSION

In this work we show that students at even a highly regarded, top ranked institution have misconceptions about software engineering. Along with the scientific results come analysis and hypothesis about some of the findings.

5.1 Impoverished Models

Conceptual change theories can be used to explain some of the phenomena presented in the results. The surprising positive correlations and inconsistent choices of individuals shown by the misconception maps indicate that students are not successfully integrating all the messages they are receiving in their academic studies.

In this work we have framed the misconceptions as declarative knowledge, and not as a connected model. It can be argued that the workings of a software engineering company and beliefs about appropriate work interactions are an integral part of the model of a successful business. However, in this work we show that students can both believe a misconception and its inverse at the same time. This should cause a conflict in their internal model of the system, and so we postulate that students are storing this knowledge as declarative statements that are isolated from a larger model. Cognitive scientists have shown that novices often store information as collection of unconnected facts about a domain [6, 9, 13, 14]. The misconceptions presented here align with this idea of individual pieces of knowledge that do not have an impact on a larger model or reasoning [6].

In an attempt to have students form complex models of the software engineering process, we often involve them in smaller scenarios designed to offer insights or experiences in projects where communication and effective processes will yield a more positive experience and, hopefully, a better project. Unfortunately, small groups working within the constraints of an academic setting offer a very impoverished model of industrial practice. Short time lines, lack of resources (in terms of manpower or support/management), and the individual differences between a team of all students (novices) and a team more diversified in experience can lead students to draw incorrect conclusions about the realities of industry. The impoverished situations that students work in combined with their novice perspective is contributing to inconsistent integration of software engineering.

5.2 Internships

The appearance of internships, overall, as a negative significant correlating factor is interesting, however it is more interesting where it does not appear in the individual misconceptions (see Table 5). When completing an internship students work within a company for a few months in order to both strengthen their skills and resume, but also to experience the culture of the company and the software engineering profession as a whole. The lack of a significant internship correlation with such process based misconceptions such as choosing to work alone and accurate description by clients suggest that students who are participating in internships may not be explicitly seeing defined process, especially in companies where the process is so well defined it may be mistaken for culture.

Although internships did correlate with a decrease in the number of misconceptions chosen, students who had completed internships still exhibited a large number of misconceptions at various levels. If we use the number of times that

students chose the misconception, or rejected the inverse as a measure, students who participated in internships had on average 5.8 misconceptions they selected at least once, 3.2 misconceptions that they selected 2 or more times, and .8 misconceptions they selected 4 or more times.

5.3 Course Correlations

One of the largest surprises from the analysis is that the web applications and operating systems courses correlated positively with specific misconceptions. Students normally take these courses their junior year, and participant age and number of internships did not statistically account for these correlations. These correlations are particularly interesting because the faculty in these courses promote good software engineering practices and have many years of industry experience. We hypothesis that this is not the problem of the faculty, but a problem that the students do not understand the differences between the challenges of a group project in school and a project in industry.

5.4 Future Work

Future work in this area includes two distinct approaches. In order to make the findings of the work more generalizable a larger, multi-institution implementation of the assessment would allow us to look for misconceptions that span a broader student population. In addition, including non-computer science students in the data could provide insight into what misconceptions are promoted by society and popular culture and how they align with misconceptions of computer science students.

The assessment tool itself could use further testing and validation. Informal feedback indicated that sometimes after seeing a statement more than once, choices were made without reading what the statement was compared to. Furthermore, a more qualitative study of the students with misconceptions could yield additional verification that the delay in time for decisions in the misconception map translates specifically to strength of the misconception.

in this area includes a larger, multi-institution implementation of the assessment to look for generalizable misconceptions. We also believe it would be interesting to survey non-computer science students to see what misconceptions are promoted by society and popular culture and how they align with misconceptions of computer science students. The assessment tool itself could use further testing and validation. Informal feedback indicated that sometimes after seeing a statement more than once, choices were made without reading what the statement was compared to. Also, a more qualitative study of the students with misconceptions in order to determine if the claim of strength indicated by the misconception maps hold up to a qualitative study.

Every misconception was chosen multiple times and by multiple students indicating that our list was appropriate for the student population. It may be informative to expand upon this list for future studies to determine the boundaries of student misconceptions. It may also be interesting to conduct qualitative analysis such as interviews to see if students misconceptions can be categorized into such divisions as process, job awareness, and goals.

6. CONCLUSIONS

The contributions of this work exist on two levels. For the local institution, this work highlights two important pieces of understanding. First, students at this institution have misconceptions about many facets of their future in engineering. Secondly, the structure of particular courses needs to be examined in order to determine why they are correlating with an increase in misconceptions by students.

For the larger community, we do not claim that our results are generalizable, but instead hold forth the methodology and analysis for implementation at other individual institutions. The combination of the macro level analysis of a program, and the student and subgroup analysis offered by the misconception maps are a powerful combination.

In addition to research in software engineering education, the forced-choice survey and data analysis with misconception maps could be used in other areas where student beliefs interfere with potential outcomes, or need to be measured and understood.

Our goals with this research were to develop an assessment to determine (1) if undergraduate students at Carnegie Mellon University did have misconceptions about software engineering as a discipline, and (2) if we could also measure and visualize the strength with which the misconceptions were held. The overall analysis did show that students held misconceptions from a list constructed by the software engineering research group, and verified with current professionals working in the field. The misconception maps offer a useful visualization for the data, and a potential way to view the strength of the misconceptions in addition to their existence.

Current undergraduate computer science programs focus on programming, and the creation of data structures and algorithms. It is our opinion that this focus as well as the common practice of having students build well-defined assignments from scratch help fuel the misconceptions about software engineering. One of the most frequently chosen misconception, that students should take advantage of implementation details in other code, could arise from students writing all the code themselves and therefore having an intimate knowledge of those details. Another misconception, that most of their time will be spent developing new algorithms and data structures, echoes the majority of coursework that students see in their undergraduate program. Overall, this work offers awareness of an issue in our undergraduate programs as well as an assessment that offers more information than a standard survey.

7. ACKNOWLEDGMENTS

The authors would like to thank our advisors, Mark Stehlik, Sharon Carver, and Johnathan Aldrich. This work was partially funded by the computer science department at CMU, as well as the Institute of Education Sciences, US Department of Education, through Grant R305B040063 to Carnegie Mellon University. The opinions expressed are those of the authors and do not represent the views of the Institute or the US Department of Education.

8. REFERENCES

[1] A. Begel and B. Simon. Novice software developers, all over again. In *ICER '08: Proceeding of the Fourth international Workshop on Computing Education Research*, pages 3–14, New York, NY, USA, 2008. ACM.

[2] A. Begel and B. Simon. Struggles of new college

graduates in their first software development job. volume 40, pages 226–230, New York, NY, USA, 2008. ACM.

[3] E. Brechner. Things they would not teach me of in college: what microsoft developers learn later. In *OOPSLA '03: Companion of the 18th annual ACM SIGPLAN conference on Object-oriented programming, systems, languages, and applications*, pages 134–136, New York, NY, USA, 2003. ACM.

[4] P. V. D. Broek and P. Kendeou. Cognitive processes in comprehension of science texts: The role of co-activation in confronting misconceptions. *Applied Cognitive Psychology*, 22:335–351, 2008.

[5] M. Chi. Commonsense conceptions of emergent processes: Why some misconceptions are robust. *The Journal of the Learning Sciences*, 14(2):161–199, 2005.

[6] M. Chi and S. Ohlsson. *Complex Declarative Learning*, chapter 16, pages 371–400. The Cambridge Handbook of Thinking and Learning. Cambridge University Press, 2005.

[7] M. Chi and K. VanLehn. The content of physics self-explanations. *Journal of the Learning Sciences*, 1:69–105, 1991.

[8] C. Chinn and W. Brewer. The role of anomalous data in knowledge acquisition: A theoretical framework and implications for science education. *Review of Educational Research*, 63(1):1–49, 1993.

[9] A. DiSessa. *Knowledge in Pieces*. Constructivism in the Computer Age. Erlbaum, 1988.

[10] D. Hammer. Misconceptions or p-prims: How may alternative perspectives on cognitive structure influence instructional perceptions and intentions. *Journal of the Learning Sciences*, 5(2):97–127, 1996.

[11] K. Hamza and P. Wickman. Describing and analyzing learning in action: An empirical study of the importance of misconceptions in learning. *Science Education*, 92(1):141–164, 2008.

[12] T. C. Lethbridge. A survey of the relevance of computer science and software engineering education. In *CSEET '98: Proceedings of the 11th Conference on Software Engineering Education and Training*, page 0056, Washington, DC, USA, 1998. IEEE Computer Society.

[13] National Research Council. *How People Learn: Brain, Mind, Experience and School*. National Academy Press, 2000.

[14] G. Ozdemir and D. Clark. An overview of conceptual change theories. *Eurasia Journal of Mathematics, Science & Technology Education*, 3:351–361, 2007.

[15] D. Perkins and R. Simmons. Patterns of misunderstanding: An integrative model for science, math, and programming. *Review of Educational Research*, 58(3):303–326, 1988.

[16] L. A. Sudol and C. Jaspan. Declarative misconceptions in software engineering. Technical report, Carnegie Mellon University, 2010.

[17] D. Taft. Programming grads meet a skills gap in the real world. *eweek.com*, 2007.

[18] W. Vosnaidou, Stella; Brewer. Mental models of the day/night cycle. *Cognitive Science*, 18:123–183, 1994.