STUDENT CONCEPTIONS ABOUT THE FIELD OF COMPUTER SCIENCE

A Dissertation
Presented to
The Academic Faculty

by

Michael Hewner

In Partial Fulfillment
of the Requirements for the Degree
Doctor of Philosophy in
Human–Centered Computing

School of Interactive Computing
Georgia Institute of Technology
December 2012

STUDENT CONCEPTIONS ABOUT THE FIELD OF COMPUTER SCIENCE

Approved by:

Dr. Mark Guzdial, Advisor
School of Interactive Computing
Georgia Institute of Technology

Dr. Amy Bruckman
School of Interactive Computing
Georgia Institute of Technology

Dr. Keith Edwards
School of Interactive Computing
Georgia Institute of Technology

Dr. Ellen Zegura
School of Computer Science
Georgia Institute of Technology

Dr. Yasmin Kafai
School of Graduate Education
University of Pennsylvania

Date Approved: November 5, 2012

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

SUMMARY

Computer Science is a complex field, and even experts do not always agree how the field should be defined. Though a moderate amount is known about how precollege students think about the field of CS, less is known about how CS majors conceptions of the field develop during the undergraduate curriculum. Given the difficulty of understanding CS, how do students make educational decisions like what electives or specializations to pursue?

This work presents a theory of student conceptions of CS, based on 37 interviews with students and student advisers and analyzed with a grounded theory approach. Students tend to have one of three main views about CS: CS as an academic discipline focused on the mathematical study of algorithms, CS as mostly about programming but also incorporating supporting subfields, and CS as a broad discipline with many different (programming and non-programming) subfields. I have also developed and piloted a survey instrument to determine how prevalent each kind of conception in the undergraduate population.

I also present a theory of student educational decisions in CS. Students do not usually have specific educational goals in CS and instead take an exploratory approach to their classes. Particularly enjoyable or unenjoyable classes cause them to narrow their educational focus. As a result, students do not reason very deeply about the CS content of their classes when they make educational decisions.

This work makes three main contributions: the theory of student conceptions, the theory of student educational decisions, and the preliminary survey instrument for evaluating student conceptions. This work has applications in CS curriculum design as well as for future research in the CS education community.

CHAPTER I

INTRODUCTION

"When I was younger, I called it programming, 'cause that's what I thought it was. When I first came to Tech, my idea of what I might do was along the lines of: I'd make video games . . . But the actual term 'Computer Science', I had never heard until I declared my major as Computer Science, 'cause that was the closest thing to a programming word. And since then, it's only been the fact that this college calls it 'Computer Science' that makes me think of it. It seems like kind of an awkward term, honestly. And I think it's the right term; it's just — it just sounds weird for some reason."

—Quote from student interview (P27)

Students come to CS from a variety of backgrounds and with a variety of preconceptions. Some (as in the quote above) select CS with a very vague idea of the field they are majoring in. Even if a student researches CS before they select a major, however, it is certainly reasonable to expect that their understanding of the field will develop as they progress through the undergraduate curriculum.

Most CS educators have experience with students who come to CS with misconceptions about what the field of Computer Science is. Some examples that educators frequently mention anecdotally are:

- Students may come to CS expecting to learn "advanced features" of existing application software.
- Students may come to CS expecting to learn how to do IT work like assembling

computers from parts and configuring routers.

- Students may come to CS thinking just about programming and not understand why learning theory or architecture is valuable.

In departmental materials, textbooks, conversations with students, and occasionally in class, we attempt to educate students about what CS really is. Educators hope that students in time come to an accurate view of what the field of Computer Science is, but understanding the field of CS is usually not a curricular goal or something we assess. Most importantly, the computer science education community does not know much about how this understanding of the field of Computer Science changes over time.

Of course, not all views of Computer Science that differ from experts are the result of confusion. Students may understand the "expert" point of view and feel that view is wrong:

- Students may feel the credential of a CS degree is important, but feel that the actual material covered is not useful to them in the "real world".
- Students may feel that understanding CS is important, but only as part of a larger different goal that is not purely CS (e.g. manager, computational artist).
- Students may have a position in the ongoing debates on what CS "ought to be". They could feel that CS needs to be more human–centered, interdisciplinary, or theoretical, etc.

Some educators might wish to encourage at least some of these viewpoints; certainly there is value in a spirited debate about the direction of CS going forward. Regardless, similar to basic confusion, students who have different conceptions of CS may want different things from their CS classes.

When educators design their classes, obviously they take into account the fact that student goals and expectations may be different. But because little is known about how CS students conceive of the field, educators must rely on their intuition and ad–hoc conversations with students. By knowing how students view the field of CS and how they make educational decisions, CS educators and curriculum designers can better teach to the variety of student perspectives that exist.

In this document, I describe my research into student conceptions of the field of Computer Science. My approach was cognitive: I studied what students think CS is and how students made decisions about their courses and curriculum. Based on 37 qualitative interviews with students and student advisors, I extracted and described three main conceptions about CS found in undergraduate CS majors. Based on these viewpoints, I developed a preliminary survey instrument that I used to estimate the prevalence of these conceptions in a class of CS students.

In this chapter, I present an overview of my approach, the arguments that understanding conceptions of CS is important, and a brief summary of my two studies and their results. In chapter 2, I provide discussion of related work in the general areas of science education and the computer science education community in particular. In chapter 3, I provide detailed information about the design of my studies and my qualitative analysis process. In chapters 4-6, I present the detailed results of those studies. Chapter 7 presents the educational implications of my work and some thoughts for future research.

1.1 Conceptions of CS and Educational Decisions

There are many ways we could examine student perspectives on CS. Other research has considered a variety of factors: if students think CS is fun or a good match for them [41], if students are unduly affected by stereotypes [45], if the culture of CS is welcoming to new participants [57, 64], and many other aspects. I focus on the cognitive aspect of CS major's perspective of the discipline; I am interested in what majors think CS is and what they expect to learn by studying CS.

Much of my approach comes from the large literature of alternative conceptions in science [79]. In science education literature, an 'alternative conception' is simply an existing student understanding of science concepts that usually differs from the 'expert–like' understanding of their teachers. Alternative conceptions are usually about science content, not conceptions of science fields (e.g. alternative conception researchers generally study if students understand evolution, not if students understand what biologists do). Nonetheless, the alternative conception literature has established a variety of robust findings about students learning difficult concepts that have been applied to many areas of education and many groups of students.

My work also draws on the literature on student epistemologies of science. This is an area of science research that explores how students think about the process of scientific discovery and argumentation. One thing that differentiates epistemology of science research from scientific alternative conceptions is that there is no single correct view of the scientific process [68]. Similarly, experts in CS have different views of the field [52] so it is reasonable to think students might have different but valid views as well. From an educational perspective, establishing a single universal definition of CS is not important. Students simply need a clear enough understanding of CS to choose good classes, attend to the right material in class, and make other ordinary educational decisions. Therefore, my definition of CS conceptions focuses on student understanding of the curriculum and educational choices.

I define a student's *conception* of CS to be what skills and concepts a student expects to learn in their CS curriculum and how they expect to use those skills and concepts after graduation. This includes an understanding of the reasons behind the curriculum: why particular concepts are covered and others are left out. Although there is expert disagreement about a formal definition of CS [52], the curriculum itself is fairly standardized [70]. Even if not all CS experts agree with the curriculum, it is reasonable to expect that students who do not understand the CS curriculum may make poor educational decisions.

I define an *educational decision* as any choice a student makes about their own education. For example, selecting a particular major or choosing a particular elective course are both educational decisions. Educational decisions can also be as small as identifying the material likely to be on a particular test. This research focuses on how students make educational decisions and especially how reasoning about conceptions enters into this process.

This research also classifies student conceptions into two categories: *productive conceptions* and *potentially problematic conceptions*. Productive conceptions are those that show a good overall understanding of how the parts of CS fit into a coherent whole and have a logical vision for how those concepts and skills will help after graduation. A student may have a productive conception even if they are not certain how some specific concepts (e.g. finite automata) fit into the overall picture. Overall, a productive conception is one that lets a student reason about CS and make informed educational decisions.

Potentially problematic conceptions are conceptions that in some way significantly deviate from the curriculum. A potentially problematic conception might be an outright misconception about CS (e.g. CS is training in using advanced features of applications). A potentially problematic conception could also be simply overfocusing on some real aspect of CS (e.g. CS is programming). Of course, it is possible that a student with a potentially problematic conception of CS may never have a problem; the conception may be corrected before the student makes any significant educational decisions. A potentially problematic conception is simply a conception that has the potential to cause poor educational decisions.

Research on precollege students CS suggests that, at least initially, many students may have potentially problematic conceptions. Similar to science concepts, we have evidence that few high–school students in the general population understand much about CS. Carter [9] reports that of students in a high school calculus and precalculus class, 80% of students reported that they did not know what Computer Science is. Qualitative research suggests similar results for a variety of age groups [56, 81]. However, previous work with college students near graduation suggests that at least near the end of their careers, student views of CS differ significantly from incoming students [43, 6, 59].

1.2 Why Study Conceptions of CS?

My own previous research and the literature CS education provides some evidence that many students come to CS with potentially problematic conceptions. What is not clear is if these potentially problematic conceptions actually cause poor educational decisions or have implications for CS educators. I argue that conceptions are likely to persist longer than we expect, that have implications for CS education, and that the types of potentially problematic conceptions are likely to be different in CS than they are in other fields.

1.2.1 Conceptions are Persistent

Taking a page from the large literature of science alternative conception research, we should be skeptical about assuming that students quickly overcome their initial potentially problematic conceptions about Computer Science. One of the most commonly observed aspects of alternative conceptions in the science literature is that teachers habitually underestimate how difficult it will be to change student conceptions [79]. Often, even after alternative conceptions are explicitly discredited in class, students will either misinterpret this new information as supporting the conception they still have or adopt parallel concepts: one for answering teacher questions, and one for reasoning in everyday life [38].

Even if students do change their conceptions after participating in CS classes, there is no reason to assume that the new concept of Computer Science is more expert–like than the old. Stevens [75] discusses how students in engineering often view "what makes a good engineer" in the light of their current coursework. In the beginning, student viewpoints in engineering are dominated by the view of an engineer as someone who works on highly constrained mathematical problems with known answers. When the coursework finally begins to be more open-ended and deal with real world problems, students confident in the "math problem" approach now have significant stress. There is evidence to suggest that this tendency to narrowly define the field by introductory coursework occurs in CS too. McGuffee [59] reports that while initially students' definitions of CS are too broad, after one CS class students narrowly define CS as just programming.

My research was not longitudinal, so it is difficult to know with certainty if student conceptions persist. However, it was clear interviewing students that their views of CS often incorporated incomplete aspects of topics they had heard from class. This is what might be expected as new concepts of the CS field clash with existing student ideas. For example, many students definitely mentioned that CS was not simply programming (and that this was something they had heard from instructors). However, when I asked students for some examples of CS topics that were not programming–related, they often had great difficulty. Although students obviously had heard something about what CS was, they had not yet reconciled that with the programming–oriented classes they were taking. Changing student conceptions of CS does seem to be challenging.

1.2.2 Conceptions Have Educational Implications

Even if students have expert–like views of CS, there are still several conflicting opinions about what CS is. If some students expect a mathematical theoretical approach to CS and other students expect focus on the mechanics of programming, building a class that appeals to both is difficult. By understanding what conceptions students are likely to have, instructors are better able to address student expectations.

Beyond the issue of individual classes, understanding how students make educational decisions has curricular implications. In my interviews, I found students often relied heavily on the curriculum to ensure that they would learn any essential CS content. That puts different constrains on curriculums than students who have strong opinions about CS and want greater freedom to specialize. Knowing to what extent students use conceptions of CS in educational decisions suggests how important it is develop a accurate view of the CS field early on.

Finally, there is a possibility that more accurate conceptions of CS make it easier to learn CS content. Obviously, a accurate view of the field of CS is not required to learn specific CS content. Even if a student thinks Computer Science is simply training in how to use applications, they can still learn CS content like variables and loops. But we know from general psychology research that what information an observer is looking for greatly affects what they notice [34]. Similar to research with experts and novices [16], understanding what features of information are important is an large part of the skills that differentiate experts and novices. If a student expects to learn about using applications, it is reasonable to suspect they might focus on how to use the IDE more than they focus on how to construct good algorithms. My research did not focus on establishing a connection between conceptions of CS and academic success in CS (though I address the possibilities in more detail in Chapter 2). However, the relationship between conceptions and content leaning is still an interesting avenue for future work.

1.2.3 Conceptions Vary Between Fields

Every field probably has issues with majors not initially understanding the field. But although the fact that students have potentially problematic conceptions is similar, the ways in which each field develops student understanding is deeply involved with the individual concepts of the field. Because educational practices differ between fields, it is reasonable to expect that student's conceptions change in different ways.

Studies in various disciplines definitely provide evidence that student development is different across disciplines. In Nespor's studies [60], physics students and management students experienced very different trajectories as they advanced in their majors, and these are different than the stages of development in engineering disciplines that Stevens notes [75]. Each of these disciplines has to deal with some similar challenges: students in each field feel that the actual professional practice of the discipline is distinct from the problems of classroom. In engineering, Stevens [75] notes that, as the curriculum progressed, professors moved more in line with "real-world" problems and constraints, as well as increasing training in laboratory skills. This caused significant stress as students dealt with having to apply more varied skills to less constrained problems. In Nespor's physics students, by contrast, the essential problems remained similar but became more abstract: students might work a problem they had worked in previous semester but this time from more basic principles. Because physics is not a design-oriented discipline, physics students focused exclusively on the difficulty of theory and problem solving, and not vague requirements or managing large scale group projects.

There is also reason to think that Computer Science is a particularly interesting field to study conceptions. CS is a new field and even among experts there are a variety of opinions about what is and is not CS [52]. Probably even within individual departments, many potential viewpoints on what CS is are represented. In research in CS conceptions, it is especially important to keep in mind that while some student may have conceptions of CS that hurt them academically (and this is problematic), there are many potential conceptions of CS that may be different but perfectly productive for our purposes.

1.2.4 Summary

In the previous section I reviewed several arguments about why learning student conceptions in important. From other research into alternative conceptions, we find evidence that alternative conceptions are resistant to education and persist longer than instructors suspect. From general research in psychology and studies of other educational settings, we find evidence that student conceptions are likely to affect learning and behavior. From educational research, we find evidence that although conceptions exist in every field, they manifest in different ways and therefore it worthwhile to look at CS specifically.

1.3 My Previous Work

This document focuses on two main studies specifically designed to elicit conceptions of the field of CS in undergraduate students. However, this is not the first time I have looked at student views of the field. Two studies served as inspiration for the work presented here.

The first study was designed to identify whether student's college experiences, including interest–targeted introductory CS courses, had an affect on student attitudes about computing four years later. This project compared essays from college seniors in different majors about computing. The difference between CS majors and non–CS majors in these essays were striking: although both groups enjoyed computers the CS majors emphasized the breadth and surprising nature of computing while non–majors focused mainly on enjoying technology. This suggested that the CS undergraduate experience really does seem to change student conceptions.

The second study was explicitly designed to elicit conceptions of CS, but in pre–college students. I interviewed high school students who had had some non–traditional CS instruction. Students in this group often had potentially problematic view of CS, especially that CS was application use. Even after explicit instruction in the field of CS, student potentially problematic conceptions persisted. This work is what originally made me consider whether student conceptions of CS might cause educational problems at the college level.

Both these studies suggested that student conceptions of CS were likely to be complex. Both of them also used qualitative methods to evaluate student views (although each had some problems). These studies are discussed in detail in Chapter 2. The results of these studies provided motivation to explore the conceptions of undergraduate students in more depth.

1.4 Studying CS Conceptions

In the previous sections, I argued that:

1. There is good reason to think that students have different conceptions about Computer Science across their undergraduate career.

2. These conceptions are likely to affect their education in a variety of ways

3. My previous work suggested the conception of CS undergraduates would be a fruitful topic to study

In this section I describe two studies to elicit student conceptions of Computer Science. See Table 1 for a summary of research questions and the studies that address them.

1.4.1 Thesis Statement

By examining the evidence of student concepts of Computer Science in undergraduate students, I believe it will be possible to:

Table 1: Summary of research questions and studies

RQ1: What types of CS field conceptions exist in CS undergraduate students?	H1. CS majors will exhibit a changing understanding of CS, initially potentially problematic but becoming more productive.	Grounded theory analysis based on interviews with students, student surveys, advisors	Study 1 n = 37
	H2. Multiple productive conceptions will be found in graduating undergraduate students.	Grounded theory analysis based on interviews with students, student surveys	Study 1 n = 37
RQ2: Do potentially problematic CS conceptions affect student educational decisions?	H3. Potentially problematic conceptions of CS will affect educational decisions.	Grounded theory analysis based on interviews with students	Study 1 n = 37
RQ3: What is the prevalence of different kinds of conceptions among the CS major population?	H4. Students conceptions will vary across the student population, with potentially problematic conceptions persisting after introductory coursework.	Survey (instrument based on grounded theory)	Study 2 n = 99

1. Characterize changes in student understanding of the CS field

2. Develop a preliminary instrument that can be used to elicit student understanding of the CS field

1.4.2 Study 1: Design

Study 1 is a qualitative study that focused on eliciting students' conceptions of CS. Study 1 addressed two main research questions:

RQ1: What types of CS field conceptions exist in CS undergraduate students?

RQ2: Do potentially problematic CS conceptions affect student educational decisions?

To answer these research questions, I designed Study 1; a qualitative study based on several data sources:

1. *Interviews with CS counselors.* Those who advise CS students have a large opportunity to examine a variety of student conceptions about CS, and especially how these conceptions factor in to students' educational decisions.

2. *Interviews with students.* I interviewed CS majors at three different schools about how they view CS and how they feel that view has changed over time.

3. *Survey Instruments.* As I worked with students through interviews, I planned to refine an open–ended survey instrument with questions that elicit student conceptions of CS.

Students were drawn from various stages in their major, and varying levels of academic success. To select which students to study, I engaged in theoretical sampling [18]: selecting which types of students would be most likely to provide new insights based on the analysis of interviews thus far. I used grounded theory [11] to analyze all the data sources; the goal of the analysis will be to extract a general theory of the various student conceptions of CS at different points in the curriculum. In accordance with grounded theory practices, I interviewed until the theoretical categories achieved saturation (i.e. new interviews do not generate new theoretical constructs). Each interview took about 90 minutes in total. In the end, I interviewed 37 participants (33 students, 4 Advisors).

1.4.3 Study 1: Results

My grounded theory analysis of student interviews of CS identified three main conceptions of CS:

1. *Theory–View: CS as Mathematical Study of Algorithms.* Students who held this view thought of CS as a primarily theoretical and mathematical discipline.

The design of conceptually difficult algorithms was most central to CS, as were other mathematical ideas like Big O and NP–Completeness. Programming was viewed as useful but more peripheral to CS, and students often emphasized that CS could exist without any physical computer.

2. *Programming–View: CS as Programming–Centric but Including Supporting Subfields.* Students who held this view considered CS to be mainly about programming, but emphasized that other subfields were also necessary to do good programming. Writing programs to solve large and technically challenging problems was the central activity. Students with this view varied on how important non–programming subfields of CS were.

3. *Broad–View: CS as Having Many Different Subfields.* Students who held this view thought of CS as mix of many different computer–related subfields. Theory, Robotics, Programming, and (often many) others are all equally important parts of a broad CS 'umbrella'. In this view, comparatively little knowledge was considered 'essential' to a Computer Scientist; students emphasized the differences between subfields and the freedom to pursue different paths.

All three of these viewpoints were a high–level view of CS. Students did not know many details about subdisciplines of CS or the content of future courses. Details of the three main conception types are covered in Chapter 4.

Beyond simply cataloging conceptions of CS, I also probed how students made educational decisions. My interviews suggested students make educational decisions in a way that initially seemed arbitrary. Students in CS generally did not have specific career goals or skills they were hoping to learn in Computer Science. As a result, they did not worry about which courses or specializations would best help them achieve their goals. Instead, they were mostly concerned about finding an area of CS that they would be well–suited for. They measured how well–suited they were for a particular area by how enjoyable they found classes in that area.

Students would explore different classes, relying on the curriculum to ensure to ensure that they did not make any educational mistakes. As a result, they did not use their conception of CS to reason about class and specialization choices. More details about the way students made educational decisions and the role of conceptions and enjoyment is covered in Chapter 5.

1.4.4 Study 2: Survey of Students in a CS Class

RQ3: What is the prevalence of different kinds of conceptions among the CS major population?

From the qualitative work in the previous study, I devised a preliminary survey instrument that can be used to assess student alternative conception prevalence on a larger scale. The survey included both closed–form and open–ended questions. I tested and revised the survey using a thinkaloud protocol. Finally, I used the survey to evaluate the prevalence of CS conceptions in one class. I also attempted to build a programmatic classifier to classify student conceptions based on the closed–form responses alone.

1.4.5 Study 2: Results

All three of the major conceptions were recognized in students. The programming–view was the most common (41%) followed by the broad–view (27%) followed by the theory–view (8%). I also performed post–hoc statistical analysis on the results. Membership in underrepresented groups seems likely to affect conception, although this is not a strong statistical claim. The programmatic classifier was 74% accurate, but would still need to be improved for larger scale research work. Chapter 6 presents the results of Study 2 in detail.

1.5 Summary

This chapter has presented an overview of the argument that CS conceptions affect student thinking and are worth studying. The next chapter provides a more detailed version of this argument, with a more complete analysis of previous work in student conceptions. This chapter has also provided an overview of my research and results. The main thing to keep in mind from this chapter is that my work has two main goals. First, I want to understand what conceptions of the field of CS exist in undergradute CS majors (and how prevalent they are). Second, I want to determine how undergraduates make educational decisions and if problematic conceptions of CS lead them to make ill–advised decisions,

CHAPTER II

PREVIOUS WORK

This chapter covers five main topic areas:

1. *Student Conceptions.* Studies on students' views of science, student conceptions in general, and how conceptions might affect content learning. This topic includes examples of qualitative approaches to understand student views.

2. *Differing Expert Definitions of a Field.* My work is made more complicated by the fact that there is no single "right" concept of Computer Science. This section discusses the controversy and how decisions of definition have real consequences for students.

3. *Studies of Enjoyment and Decision Making.* There are a great number of psychological theories about how decisions are made. I look at theories of how enjoyment motivates decision making and at the Eccles model, which has been used to study educational decisions similar to the ones reviewed in this study.

4. *Studies of Student Field Conceptions.* Though my focus on the cognitive understanding of the field of CS has not been examined specifically before, many studies have interesting results concerning how students conceptualize their field. This section also includes a survey based study that I did which was an early inspiration for this work.

5. *Conceptions of CS in High School Students.* The one of the main inspirations for this proposed work is a study I did with high school students. This section discusses the method I used on that project and what I observed of high school student conceptions.

These sections draw on two different bodies of literature: educational research from other fields and CS–specific education research. Disciplines like science and engineering education research are often older and have established methods. CS–specific research give more information about the specific attitudes and problems that CS students have.

2.1 Student Conceptions

My focus on students' understanding of their field draws heavily on existing literature in education. This section covers three related areas of study:

1. *Epistemology of Science.* One area of science education research that is very close to studying field conceptions is research into student beliefs about science (e.g. role of experiments in science, differences between laws and theories, etc.).

2. *Alternative Conceptions.* Alternative conceptions is the idea the students may have their own concepts for topics instructors wish to teach, and that these existing student conceptions are resistant to change.

3. *Relationship Between Field Conceptions and Learning Content.* There are several different studies that suggest that having a greater understanding of a particular field makes it easier to learn content in that field.

2.1.1 Epistemology of Science

Though teaching science facts and theories is important to educators, it is not usually the only goal. Educators often wish students to learn truths about the nature of scientific endeavor. It is important for students to learn the relationship of scientific theories to evidence, how various scientific methods work, the differences between a hypothesis law and theory, and how scientific knowledge is tentative [68]. These "big ideas" of science are often referred to as teaching the nature of science (NOS) or as the epistemology of science.

Although teaching the epistemology of science is important to many science educators, teaching the epistemology of science is difficult. In general, research in this area shows that precollege students often have simplistic views of science and this result has been confirmed by many different studies [51]. For example students often [68]:

- Characterize scientific truth as coming directly from experiments rather than understanding the role of analysis and interpretation
- Fail to distinguish correctly between theories and evidence for those theories
- Describe hypotheses theories and laws as basically the same thing but with different levels of evidence

Problems with student epistemology of science has been shown to be present in students throughout middle school and high school and even with science teachers [51].

This section will talk about two aspects of epistemology of science research that is relevant to student conceptions of CS. First, I will review how student epistemologies have been elicited. Second, I will examine the relationship between student epistemologies of science and pedagogy.

2.1.1.1 Eliciting Student Epistemologies of Science

Eliciting student epistemologies of science is actually a very similar problem to eliciting student conceptions of CS. Similar to CS, although there is general agreement about what the endeavor of science is among experts, there is also some debate. Epistemologies held by students bear little resemblance to the epistemologies of experts and even expert educators may not really know what epistemologies students have. Finally, there are difficulties of terminology: even if students do have "expert–like" epistemologies of science, they are not likely to use the same words that experts would use.

There is robust debate on the correct way to elicit student epistemologies of science [68]. Most of the early work focused on quantitative measures of science, but criticism that researcher–designed quantitative instruments were not truly measuring students' viewpoints has motivated more qualitative approaches [51]. Aikenhead and Ryan, for example, argue that attempting to measure student epistemologies of science without understanding student views and interpretation makes it impossible to determine what quantitative survey results actually mean [2]. In their work designing an instrument for assessing student epistemologies of science, they found that semi–structured interviews provided the most unambiguous measure of student views, but that doing interviews was very time intensive. They analyzed students' written arguments to design multiple choice questions that could be deployed on a much larger scale [2, 67].

One criticism of this research which is particularly relevant to conceptions of Computer Science is that it is not reasonable to assume that student conceptions are coherent [68]. For both epistemologies of science and conceptions of the field of CS, experts are much more likely to have thought deeply about the issue and developed a single universal view. Students' views on the other hand, may be "inconsistent, fragmented and possibly unstable" [68]: in that sense, asking abstract questions like "how would you define Computer Science" is not likely to yield accurate results. Sandoval argues that student reasoning can be mostly clearly seen when it is connected to their practice of developing specific scientific artifacts [68]. Similarly, I found student reasoning clearest when discussing educational decisions they had actually made and their reasons for them. However, this sort of contextualized reasoning is difficult to do in a short survey like the one developed for Study 2, so this issue of assuming a single coherent view remains a concern both for Study 2 and more broadly.

2.1.1.2 Teaching Epistemology of Science

One of the things that differentiates student epistemology of science and conceptions of the field of CS is that teaching epistemology of science has been an explicit curricular goal as early as 1920 [51]. Because of this, a considerable amount of effort has been devoted to attempting to teach epistemology of science to students. In particular, researchers have studied whether exposure to inquiry based science courses (some of which draw explicit connections to epistemological topics, and some of which do not) can improve students' epistemologies of science.

The research has found some educational approaches that improve students' scores on epistemology of science assessments, but finding a good approach is by no means a solved problem. Science teachers themselves often have poor epistemologies of science [51]. Educators often have to tradeoff focusing on learning science content with learning epistemology of science [51]. Activities designed to simulate scientific inquiry turn into replicating some existing experiment; implicitly conveying incorrect ideas about the purpose of experimentation [68]. Students can do authentic activities correctly (e.g. design experiments) but still have difficulty abstracting ideas into a general theory of science [68].

This area of research suggests is that even when students are exposed to authentic CS activities in classes, they can still have difficulty abstracting that idea into an overall view of CS. Even when CS curriculum is designed to teach about the field of CS (e.g a breadth–first introductory CS course [25]), it might still fail to affect students' abstract reasoning. Overall, the epistemology of science literature suggests that helping students understand the nature of a field is at least as difficult as any other challenging educational goal.

2.1.2 Alternative Conceptions

Another area of research that heavily influenced this work is alternative conceptions research. In contrast with the epistemology of science work above, alternative conceptions work usually focuses on specific science conceptions (e.g. force). Alternative conceptions provide a good perspective on how student views can develop in ways that can surprise educators.

2.1.2.1 Alternative Conceptions in Science Education

Based on constructivism and the learning experiments of Piaget, alternative conceptions research begins with the idea that students often develop cognitive models of science that differ from experts. The "alternative conceptions" that students have are accurate enough for everyday life, and maybe even accurate enough to pass simple testing, but cause students to reason incorrectly in key ways. For example, a physics–based alternative conception is Aristotelian motion; in which objects in motion naturally stop unless force is applied [62]. This can explain most ordinary phenomena students come into contact with, but is very different from the way physicists view motion. By understanding how alternative conceptions affect student thinking, science educators believe that better educational approaches can be developed.

Science researchers have found students to have alternative conceptions in every area of science and these alternative conceptions cut across normal social boundaries like race, age, and cultural background [79]. But what is most interesting about alternative conceptions is that they are very often resistant to instruction; Wandersee et al. call this resistance "the most reported finding in the field" ([79, p.190]). Gilbert et al. [39] have classified the diverse ways students with an alternative conceptions may respond to instruction that contradicts their beliefs:

- Students may simply leave their original conception unchanged.

- Students may construct two views: one for answering instructor questions and one for explaining real world situations.
- Students may misinterpret the new information as confirming their existing belief.
- Students may construct a hybrid idea with elements of the contradictory viewpoints.

These sort of responses underscore the need for careful analysis to determine if students have alternative conceptions. Poorly constructed instruments can mistake "hybrid ideas" or "two views" responses for an expert–like understanding. Given that, a large challenge in alternative conception research is designing reliable tools to determine what student alternative conceptions are.

2.1.2.2 How Alternative Conceptions are Elicited

The process of determining student alternative conceptions generally begins with a combination of instructor intuition and open–ended questions. One example is an early study by Osborne and Gilbert [62] in which students are asked general questions based on simple drawings. For example, students are shown a drawing of a man pushing a motionless car. The interviewer might ask if there is a force acting on the car. This single question can elicit a variety of alternative conceptions. A student may argue that there is no force because there is no motion (force implies motion). A student may argue that there is a force, because the person is "forcing" the car (confusion with everyday word "force"). A student may argue that there is a force, but that the car "wants" to stand still so it has no effect (ascribing desires to objects). All these are potential alternative conceptions that Osborne and Gilbert identify.

The main difficulty of initial alternative conceptions research is choosing good questions that elicit alternative conceptions. The man pushing car scenario is a good question because it highlights the differences between the students' everyday conception and that of experts. As physics educators, Osborne and Gilbert have a good intuition about the sorts of questions that make good starting points. Similarly, in other areas ([14] and [33]) educators use known problematic situations to form the basis for open ended questioning. Because science educators have a good intuitive understanding of what situations students will find difficult, initial research can start with a focused set of questions that students will have problems with.

When I began my research, there was no clear intuition about what questions will elicit alternative conceptions about the field of CS. For that reason, semi–structured interviewing focusing on students' views and educational experiences worked better than trying to ask 'tricky' questions. Once I was able to develop a theory of student conceptions of CS (in essence, developing some instructor–intuition), it did become possible to devise questions for the survey based Study 2.

2.1.2.3 How Alternative Conceptions Drive Pedagogy

The purpose of identifying alternative conceptions is not simply to understand what "errors" students are making. Even if a instructor explicitly says an alternative conception is wrong in the classroom, students will often maintain their old viewpoints [73]. Many creative classroom techniques have been applied, some have proven successful in some cases, but no generally reliable method exists for encouraging students to change alternative conceptions to expert–like conceptions [79].

The benefit to understanding alternative conceptions is not simply to explain why students are wrong, but also to understand the ways in which students are right. Smith, DiSessa, and Roschelle [73] argue that alternative conceptions are established because they work well for the practitioner and have many of the aspects of expert–like reasoning. The goal of appropriate constructivist pedagogy is to use students' existing understanding to construct a better point of view: existing alternative conceptions provide the resources necessary to do that [73].

One consequence of the need to work within the framework of existing alternative conceptions is that not every expert–like conception is equally valuable to learners. Lynn and Muilenburg [54] describe the development of heat curriculum that omitted the expert idea of heat being caused by atomic motion. Lynn and Muilenburg argue that the atomic motion model does not relate well to student everyday experiences and that instead discussing "heat flow" provides a clearer basis for students to see the benefits of scientific conceptions of heat. Though the idea of students adopting expert conceptions is appealing, the goal is really to allow students to reason accurately and provide a good basis for learning in the future.

The research on the relationship between alternative conceptions and pedagogy has several implications for my research. Not every non–expert conception is necessary bad. If a student conception is sufficient for the reasoning a student needs to do, it may be good enough (especially considering that changing it will likely be difficult). Also, this work suggests that CS educators need to build on the alternative conceptions students have—not simply attempt to replace them with the expert version.

2.1.2.4 Alternative Conceptions in CS Education

The idea of alternative conceptions has also been used before in understanding student conceptions of CS content. Even when not explicitly mentioning alternative conceptions, a great deal of CS education research focuses on student conceptions of programming, and how that understanding differs from experts (see [12] for an overview). Recently, work is being done that explicitly references the science education alternative conception community directly (e.g. [78, 42, 47]). This work has had success in documenting alternative conceptions and designing instruments that can test for specific conceptions.

Although alternative conceptions have been used to understand problems students have with CS *content*, no work in CS has been done that explicitly uses the same approach to understand problems student have with the *field* of CS. Researchers studying student views of the field (discussed in detail later in this chapter) have focused either on student affect or on the social environment of CS. The alternative conception research I propose allows the CS community to look at views of the field of CS from a cognitive perspective, and understand how different conceptions of CS can affect student educational choices.

2.1.3 Relationship Between Field Conceptions and Learning Content

In my research, field conceptions of CS did not have a large effect on educational decisions like which classes to select, at least at the undergraduate level. What is less clear is if potentially problematic conceptions of CS can influence learning directly. Is a student with a potentially problematic conception of CS less able to learn ordinary CS content like algorithms? In my research, students commented about having a lack of motivation to learn material that seemed 'useless' at the time (see Chapter 5). There is some research that suggests that having a problematic conception of the broader field can directly hurt students' ability to learn content, maybe in ways students might not notice.

2.1.3.1 Field Conceptions and Transfer

One simple connection between an understanding of the overall field and learning content is the issue of transfer. Because students' have pretty undefined career goals, they have to take a variety of courses that on the surface seem abstract and disconnected. In my interviews, students had difficulty identifying how what they were learning might be useful to them in some later career. One of the robust findings of the literature of learning transfer is that students often fail to connect abstract ideas with their applied context unless they see the connection when these ideas are taught

[7]. This is one of the factors that makes teaching for transfer frequently unsuccessful [3]. If students do not have a conception of CS that explains why CS concepts are important, it is possible that they can learn the concepts in their original classroom context and then be unable to transfer the ideas to later courses. In interviews, students did sometimes incorrectly identify the purpose of specific content in their classes. However, it is difficult to know if this made learning other aspects of CS more difficult.

2.1.3.2 Field Conceptions as Representing Disciplinary Approaches

Another connection between understanding of the overall field and learning content is the idea that there are implicit ideas at the "field level" that underlie approaches to content. Donald argues that each discipline has its own essential methods of approaching problems that are distinctive and difficult for students to internalize [26]. For Donald, a significant part of education within disciplines is conveying these ways of approaching problems. A student expecting to use everyday (or even rigorously scientific) approaches in a discipline where these approaches are not valued is sure to meet with failure.

Some examples of this idea do have experimental support in the epistemology of science literature. Having a strong understanding of the epistemology of science helps skills like designing experiments. For example Deanna Kuhn asked students and adults to reason about social problems (e.g. why prisoners commit crimes after release) as well as concrete experiments (e.g. determining what factors made simulated cars go faster) [49]. In both cases, with both children and adults, Kuhn found her subjects unable to differentiate between evidence and a plausible story to justify assertions. Even factors like expertise in the issue under consideration did not improve the quality of argumentation. However scientists were able to correctly design experiments [48] without trouble. Kuhn argues this suggests that scientific reasoning is not a simple extension of common sense. Although the design and analysis of experiments may never be explicitly taught, reasoning correctly about experiments represents a conceptual approach that people do not naturally develop without instruction.

2.1.3.3 Field Conceptions as Guides for Learning

A third connection between understanding of the overall field and learning content is the idea that an accurate view of the field allows learners to view themselves as more capable. This idea is analogous to Dweck's work on intelligence [27], who showed that learners who viewed intelligence as something that came from their own effort were more academically successful. Songer et. al [74] attempted to establish a similar connection for epistemology of science: that students who viewed science as facts to be memorized would do more poorly than students who viewed science as integrated knowledge that could be understood through reasoning. In the study, students who viewed science as facts to be learned rather than as explainations that could be changed had greater difficulties learning thermodynamics. Similar results were found in several other studies ([68, p.646]). Attitudes about how a field like science ought to be learned has affected some learning outcomes.

2.1.4 Summary

This section has reviewed three different areas of education literature. From the epistemology of science and alternative conception literature, the main point is that student conceptions are likely to difficult to anticipate. For the reason, a qualitative approach was selected to elicit student conceptions in an open ended way. Similar to others' approaches, in Study 1 I elicited student conceptions and used the resultant theory to build the survey instrument in Study 2.

Changing student conceptions is also difficult. In both the epistemology of science and alternative conception literature, attempts to improve student conceptions with pedagogical met with mixed success. Before CS educators can improve student conceptions of the field, it is necessary to both understand students' existing conceptions and if those conceptions are likely to cause educational problems.

There is no simple well supported connection between field conceptions and success in learning content. However, several different bodies of literature suggest that student conceptions may affect student learning outcomes in ways students do not expect. From interviews with students, students were not usually able to recall many situations where a view of CS hampered their ability to learn (see Chapter 5 for more details). Overall, the question as to whether problematic conceptions of CS affect students' ability to learn CS content remains unanswered.

2.2 Differing Expert Definitions of a Field

In the previous chapter, I introduced the idea of students having 'potentially problematic' conceptions of CS but did not describe what I consider a 'correct' definition of CS. Defining a field like CS is difficult. Both science and CS have a history of differing expert definitions, and these differing definitions do have educational implications. In this section, I discuss the various definitions and describe how I will evaluate student responses despite the reality of differing expert viewpoints on CS.

2.2.1 Differing Expert Definitions of Science

Educators often agree that science education is important, but disagree about why. In *Rising Above The Gathering Storm*, the Committee on Science, Engineering, and Public Policy argues that science education is essential because innovation in science and technology are necessary for improved national well–being and competitiveness [17]. Other advocates argue that science is important because voting citizens must weigh in on scientific debates or because science teaches important habits of mind [71]. Each of these viewpoints on the goal of science education suggests different scientific curricula.

There are also those who argue that standard educational approaches to science education force students to adopt a westernized culture of science [1]. While this might not seem like a bad thing to all educators, practically speaking, students who are unwilling or unable to adopt the rules of science culture can be alienated and not learn [1]. Moreover, if educators are serious in desiring a diversity of opinions in the science community, then the goal is definitely not to repress real objections that science is not moving in the right direction. It is a careful balancing act to teach students science content while at the same time not implying the scientists naturally know best or that certain types of people are not well–suited for science [5].

My research directly touches on these issues in regards to what is an appropriate definition for CS and what is not. As CS educators, we wish to encourage thoughtful debate on what CS ought to be. On the other hand, when students have a misconception that has the potential to harm them educationally, we hope educators can guide them to a better definition of CS. The controversy about science makes it clear that deciding whether a definition of is 'good enough' is not a value free choice: the goal is to allow a variety of valid conceptions of CS while understanding how some conceptions can cause educational problems.

2.2.2 Differing Expert Definitions of Computer Science

Debates about CS education parallel those of science education in many ways. There are those that argue CS teaches good habits of thought [63, 80], those that argue understanding CS is essential to good participation in modern society [66], and those who argue that CS is training for jobs that are in high demand [24]. Even some of the most famous practitioners of CS cannot agree if CS should be considered science, mathematics, engineering, or art [22]. Some of this disagreement is a matter of emphasis [52], but there are also arguments about whether subfields should be considered part of CS (e.g. [32, 58]). Even when formal definitions for CS have been attempted, they are not of a form that beginners could reasonably be expected to articulate or reason with (see [23] for one such definition). As CS mingles with other disciplines, the distinctions become even more complex; Rosenbloom [65] presents an algebraic notation to describe the various ways CS can be combined with other sciences. This problem has resurfaced again with the interest in "computational thinking" : even among experts, a definition that everyone can agree with is elusive [15].

This controversy makes it clear that my research cannot compare student conceptions to a single expert viewpoint on Computer Science. But not all the distinctions that matter to experts have educational consequences for undergraduates: students can think of CS as primarily artistic or primarily mathematical, and still have clear views of what classes and content will help them achieve their goals. Because this research focuses on conceptions of CS that have educational consequences, what is most important is how students conceptualize the content of the undergraduate curriculum.

For all the discussion of 'what CS is,' the curriculum of CS undergraduate degrees has a standardized set of topics to cover [10]. This standardization is what I will use to try and assess student understanding. If a student has an understanding of the topics covered in CS and why they are considered important, I will consider that a productive conception of CS. If a student expects to learn things in CS that are not related to the curriculum (or, in retrospect, believes instructors intended him or her to learn something outside the curriculum), my research will consider that a potentially problematic conception of CS.

Of course, it is also possible for students to understand the reasons they are taught something, yet feel that some other content is actually more useful to them (e.g. understanding that data structures was the point of the course, but feeling that the C++ programming language is actually more useful). Although the reasons behind the student's choice are definitely worth pursuing in that case, by the definition of this research, the student conception of CS is accurate. Given the wide ranges of goals and viewpoints in CS, no one can second–guess what educational choices are correct for an individual student. Students must be arbiters of what they wish to learn; but it is important they make decisions with a clear understanding of CS.

2.3 Studies of Enjoyment and Decision Making

Students in our study used their enjoyment of classes as a key factor in making educational decisions. There are many psychological models of decision making, including those that focus on issues of enjoyment [29]. In this section, we will look at a few models of enjoyment in particular, and a few models that integrate enjoyment into a larger decision making process.

2.3.1 Enjoyment

When students talked about enjoying or having fun in classes, they referred to a subjective feeling in class or doing assignments. From a student's perspective, enjoyment is a emotional response. Research has identified a few factors that help cause the feeling of enjoyment, however.

Csikszentmihalyi studied the subjective experience of experts in various fields when fully engaged, a state he called "flow" [19]. While in the flow state, individuals felt immersed in their activity and in control of their environment. To active this, the challenge of the activity needed to match the skill level of the individual: it had to be difficult enough to pose a challenge but easy enough that the individual felt it was within their capability.

Deci and Ryan identify three main needs that drive intrinsic motivation: competence (feeling skillful at a particular activity), relatedness (feeling connected to others), and autonomy (feeling in–control and consistent with one's sense of self) [20]. Deci and Ryan argue that when these needs are satisfied, individuals enjoy the activity and are motivated to pursue it in the future. For Deci and Ryan, the motivation to pursue the activity is more important than the subjective feeling while doing it. Deci and Ryan note that when individuals are required to do a particular activity for some other extrinsic reward (e.g. money) it tends to lower intrinsic motivation [21]. But under certain conditions, extrinsic controls can become internalized and motivate in a way similar to intrinsic controls. Thus, feelings of intrinsic enjoyment are affected by factors that an individual is not aware of.

Although there are other theories of enjoyment [29], these two highlight a few main points. First, people find certain activities enjoyable purely for the subjective experience and separate from any external reward (e.g. status). For that to happen certain conditions must be met. For example, the activity has to provide the right level of challenge. Second, other factors do affect the feeling including larger social forces like self identity [20]. Students can often identify some factors that helped increase their enjoyment (e.g. related to existing interests, doing well compared to other students), but there are also other reasons for their feelings they might not be aware of.

2.3.2 Eccles's Model of Achievement–Related Choices

Students are obviously motivated by more than simply how much they enjoy particular activities. Several models attempt to integrate ideas of intrinsic enjoyment, self efficacy, social factors, etc. into a single unified model [29]. One model that is particularly well suited to student educational decisions is Eccles's model of achievement–related choices, which has been applied to a variety of educational choices including selection of major and courses [28].

The Eccles model is an expectancy–value model: students made decisions based on both their estimated expectation of success and what they expect to gain (called the subjective task value). For educational decisions, the subjective task value has several aspects [30]:

- *Attainment value.* Attainment value is the perceived benefit placed on a particular choice by a valued social identity. For example, if a student considers their parents' views important and their parents value majors with good career prospects, CS might be a valued choice. If one's gender is normally associated with 'helping others', then choosing a specialization involved with helping others reinforces a valued identity. Each student values different social identities (race, gender, their particular peer group) but this is also an area where stereotypes and other aspects of culture influence student behavior.

- *Utility.* Utility is the usefulness of a particular choice, leaving aside issues of identity. Students might need certain courses to graduate, for example, or might believe that a particular skill will help them solve a particular concrete problem they expect to have.

- *Interest-enjoyment value.* Interest–enjoyment is how the subjective experience of enjoyment is incorporated into the Eccles's model.

- *Cost.* The Eccles model emphasizes that achievement–related choices often come with a cost [28]. A decision to select one course makes it impossible to take another. Eccles argues that this is what explains most variation in educational decisions: students might be interested in a particular course, but choosing to pursue it comes at the expense of another interest.

Students' expectation of success is the second half of Eccles's expectancy–value model. What 'success' means in any particular context varies by student, but students do consider their aptitude for specific courses and majors. This is in addition to the fact that a lack of confidence can decrease enjoyment in models such as Deci and Ryan's [21]. In the Eccles model, expectations of success are determined by a student's previous educational experiences and social and cultural factors.

Studies have shown that the factors outlined in Eccles model do affect student educational choices [30]. What is difficult to infer is how greatly each individual factor enters into a particular educational decision. In my research, enjoyment was frequently mentioned and attainment value type choices were mentioned less. However, it is difficult to know if other factors (in particular student expectations of success) might have affected student decisions in ways that they were uncomfortable talking about in an interview. For a discussion of Eccles model and its relationships to the results of Study 1, see Chapter 5.

2.4 Studies of Student Field Conceptions

In the previous sections, I have reviewed research related to methods of eliciting student conceptions and the effects of conceptions on learning. We have also looked at student enjoyment and how it affects educational decisions. In this section, we will look at studies that have elicited student field conceptions and what they found.

2.4.1 Studies of College Student Field Conceptions Outside of CS

Several studies have attempted to understand how students relate to their college major. Certainly students do not always make a well researched choice of major, and students' major choice are influenced by factors beyond how much interest and aptitude they think they have [46]. Students often change major in college, and the major change is often to a related field [4] which may suggest that they refine the partly uninformed choice they made when they arrived.

Several studies have focused on engineering students. Engineering has high student attrition [31] like CS. Edward's study [31] attempts to look at the problem both qualitatively and quantitatively. Edward claims that students initially envision engineering as hands–on with a mathematical component, but are unprepared for the level of abstraction in the introductory curriculum. As their career in engineering continues, students find the later courses more practical. In the end, 66% of the students felt that their classes prepared them well and looked forward to their career while "the remainder were dreading what they saw as a continuation of slogging through mathematical calculations" [31]. Edward recommends that the curricula explicitly connect between the theory and everyday engineer practice in order to prevent attrition and disenchantment with the major.

In a study that reports similar findings, Stevens [75] analyzes the results of longitudinal interviews with students in engineering programs. Stevens emphasizes that processes which seem homogeneous (student recruitment, development, and eventual success or failure in the major) result in idiosyncratic student experiences. Students attempt to display what Stevens calls "accountable disciplinary knowledge" (ADK): things that, in the view of experts, position them on the path to engineering. Students reason about the nature of the field, using the ADK they are expected to display as a guide, but what counts as ADK varies depending on the point in the curriculum and other contextual factors. As a result of this, when the ADK students are expected to display changes they experience stress. Students who initially struggle with the ADK of introductory courses can distinguish themselves later. Failure to display the expected ADK causes students to be shut out of the major or makes the students question their appropriateness for engineering. Stevens argues that deep qualitative analysis is necessary to discover the way in which the structures of the curriculum influence different kinds of development.

ADK is one way of thinking about the effect of a distant, disembodied field on the education of individual students. The field (which includes both industrial and academic experts) shapes what counts as ADK. ADK then structures the experience of students in the major. But simple departmental requirements are only part of the picture: students communicate with each other and develop a intuitive understanding of what the requirements mean [75]. Similar to the social groups discussed in Lave and Wenger [50], students form a community of practice that transmits an understanding of overall field structure. But unlike the groups studied by Lave and Wenger, the community of practice of engineering students at particular school is not really independent. If students are to come to an accurate view of their field, then their community of practice must be somehow connected to larger distributed community of practice of engineers.

How does a local community of practice relate to this larger discipline? This idea of a distant community of practice is examined in Nespor's [60] study of physics and management students. For both physics and management students, the community of other students transmitted a very distinct conception of their field.

Nespor's [60] observation of the practices of physics students suggest that the students and teachers use objects like textbooks to replicate a consistent community of practice. Nespor argues that physics's highly abstract notation and traditional problems makes this replication possible. Physics content shaped the community of students; it encouraged them to devote a huge amount of effort to constantly solve textbook problems. Because the act of solving these traditional problems takes so much effort, physics students need to work together in groups that make a community of practice locally at the school. Because the work has as its focus textbooks, lectures, and the actions of a few teachers, the community is consistent with physics student communities across many schools. The consistent use of problems and notations to generate a similar community of practice is what makes physics a discipline that can replicate itself across long distances (though, as Nespor points out, social and structural factors are in place to make this replication possible).

Nespor's example of management majors provides an interesting contrast. In this case, the curriculum is disjointed and did not build to generalized abstractions. Students generally did not have a respect for what they were learning in class; they viewed their instructors as out of touch with the true practice of management. But the majors nonetheless adopted a set of practices based around appearance and interpersonal interview–type skills. In this sense, the student–generated curriculum management majors adopted mimicked some of the characteristics of the physics curriculum (consistent, requiring practice, involving personal investment and elaboration outside of class). Nespor argues that this might even have been correct, that perhaps management majors were right in their view that the true management community of practice was based around the interpersonal skills they practiced. It is definitely true that the students reproduced the practices of a real community beyond the individual college. In both management and physics, student effort and the content of the field allowed students to feel they were a part of a community that they were not directly connected to.

This research suggests several things for my results. Firstly, because CS, like engineering, is not easily understandable by introductory students, students may struggle to understand what counts as ADK in CS. As in Edward's research [31], programming which took a central in introductory CS courses was a large part of students' vision of CS. For some students, programming was the entirety of CS — for others it was a large component but not everything. Although my study was not longitudinal like Edward's was, it is reasonable to think that student views of CS expands for many students as later classes include course work that includes different types of ADK.

Nespor's research on how communities of practice are replicated suggests a deep relationship between the actual content of the discipline and the ways students come to learn it. Certain structures within CS may encourage students to focus on the academic curriculum, as in physics, others may encourage students to construct their view of disciplinary knowledge in contrast to their instructors, as in management. Whatever the result, it seems clear that careful qualitative analysis of student concepts of CS is likely to be able to draw a connection between students understanding of CS and the way they make educational choices.

2.4.2 Studies of Student Field Conceptions of CS

2.4.2.1 Research on Conceptions of Precollege Students

Much of the research in conceptions of the field of CS has focused on the perspective of precollege students, generally middle school and high school students. It is difficult to characterize the results simply: students are often positive about technology, but simply enjoying doing “CS type” activities does not translate into feelings that CS is a good career [55]. Students obviously use technology regularly, but often have concerns about a computing career being “sitting in front of a computer all day” [81]. Developing accurate instruments is difficult [41] because small changes can significantly alter how students respond. The large–scale WGBH study [36] of students age 13-17 indicates that careers in Computing interest students across ethnic groups, although young men like computing more than young women.

It seems that students do not have a ready definition for Computer Science in general. Greening [40] asked high school students to complete the sentence “Computer Science is mostly about…”: the majority said they didn’t know or provided trivial answers like “computers”. Only a small percentage mentioned using applications, something that others have identified as a student misconception. Similarly, in a student of high school calculus students, Carter [9] found that 80% of students left blank the question “What is your impression of what Computer Science Majors learn? (leave blank if you have no idea)”. Of course, the general population may not be a good guide for the attitudes of those who major in CS: obviously everyone who chooses CS as a major as at least some preconceptions for what it will be about.

2.4.2.2 Research on the Postsecondary Community of CS

At the postsecondary level, several qualitative studies have been done to characterize the student experience in the CS major. The most well–known of these is Margolis and Fisher's study of women in the CS major at Carnegie Mellon university [57]. Margolis and Fisher focus on barriers to women's full participation in CS, and their work has many interesting reflections on how female students contrast with their male counterparts. But the work also has several tantalizing hints that reflect both the diversity of student conceptions entering CS and their evolution over time. Students entering CMU view CS as a natural fit with their role as computer wizards, an opportunity to pursue larger social goals, or as part of a opportunity to achieve financial success. As time progresses for students who stay in the program, a growing connection to CS both on a personal and academic level develops ([57] pg. 103–107). Margolis and Fisher look at this problem through a social lens: We hear about students' feelings of greater integration and success, but less about cognitive changes. Does the diversity of initial views of CS eventually converge, remain fixed, or change to another set of diverse viewpoints? Margolis and Fisher provide an excellent illustration of the deep description a qualitative study can provide, but there are many things that remain to be understood about the student experience in the CS major.

Another suggestive study is Rasmussen and Håpnes's analysis of three social groups within the CS major: computer hackers, dedicated students, and "normal" students. Based on interviews with students, the authors argue that the three groups view both the social environment of the major and CS in a different way. Most interesting, Rasmussen and Håpnes argue that the perspective of the majority of students (the group the authors term as "normal" students) is actually defined in contrast to that of the professors. To the "normal" students, the professors were disconnected from the practical reality of computing jobs and too similar to the hacker subgroup that the normals did not wish to associate with. In terms of the nature of CS, "normal" students were interested in user interfaces and other parts of CS that were considered practical. How these views were reconciled into an overall concept of CS is again outside of the scope of the primarily social analysis that Rasmussen and

Håpnes focus on.

The qualitative research that has been done highlights the stressful, occasionally unsuccessful, ways in which students are integrated in CS. Many of these barriers are social, but both of the studies above also highlight the fact that students vary in their reasons for pursing CS and that these reasons change as the students learn. Obviously, it is important that departments ease cultural problems that make students feel like they do not belong. But both these studies suggest that CS educators need to understand the various perspectives on the field, and expect that these alternative conceptions of CS will have real consequences in what students expect in the curricula.

2.4.2.3 Research on How CS Student Conceptions Change

While more is known about student conceptions at the beginning of the CS major, there is evidence to suggest that the student conceptions do change. McGuffee [59] describes student responses to the question "What is Computer Science?" He reports that at the beginning of CS1, student conceptions are too broad, while at the beginning of CS2 students definitions are too narrowly focused on programming. This is consistent with Steven's research [75] that students overfocus on what they are taught in introductory classes.

Biggers et al. [6] compares conceptions of CS in seniors: some of whom left the CS major and some of whom stayed in the major to completion. One of the main differences between the stayers and the leavers is that stayers are more likely to define CS broadly while leavers were more likely to define CS as simply programming. Because the majority of CS students left earlier in their careers, there are two possible interpretations of this result. One is that all students initially think of CS as programming, but then that conception changes as they are exposed to more courses. The other is that students with broader conceptions are likely to persist in the major, while those who think of CS as just programming are likely to leave. Either way, the study suggests important relationships between changing conceptions and student retention.

A third study on conceptions focuses on student conceptions about software engineering. Sudol and Jaspan measured student agreement with statements about software engineering that were tested on experts to ensure correctness [76]. They found that students had misconceptions compared to experts, and in general these misconceptions decreased over time. However, project courses in operating systems and web applications seemed to increase misconceptions, despite the fact that these courses are taught by faculty with real development experience and focus on software engineering concepts. The authors hypothesize that the group work in these classes is not realistic enough and therefore causes students to endorse bad practices. Clearly, even when students have been exposed to expert viewpoints, they readily develop potentially problematic conceptions based on their own observations and experiences.

2.4.3 Attitudes About Computing in Postsecondary Graduates

A previous study of mine also provides evidence of CS conception change for CS majors near graduation. The goal of the study was to identify whether student's college experiences, including interest–targeted introductory CS courses, had an affect on student attitudes about computing four years later. This project compared essays from students in different majors about computing; one of the interesting results of this study was how different CS majors' essays about computing differed from the other majors.

One of the difficult parts of this project was eliciting student experiences in a way that was not leading. Previous work had shown that students were positive about their CS courses [77]; the goal of the study was in part of to see if students would bring those courses up unprompted as significant computing related experiences. The method we chose was based on the techniques of Schulte and Knobelsdorf [69] who asked beginning CS majors and 3rd year psychology students to write “computing autobiographies”. These autobiographies revealed interesting differences in attitudes, while giving participants a great deal of control to discuss whatever they felt was personally significant.

In my study, we collected data from 4th year students at Georgia Tech [43]. The autobiographies of CS majors were clearly different from students in other majors. CS majors concentrated their autobiographies on the breadth on the discipline. Other majors were often extremely enthusiastic, but focused on technology itself as a fun thing to play with. If the student population can be considered similar to those interviewed by Schulte and Knobelsdorf [69], it is reasonable to suspect that this focus on the broad possibilities of Computer Science represents a change in conception from CS freshmen. It is also consistent with the results of Biggers [6] and Yardi [81] which similarly find CS majors who focus on the breadth of the discipline. From this, I suspect it is likely that this breadth–focused CS conception may be at least one of the conceptions my proposed work is able to elaborate. Given the limited data in the autobiographies, it is difficult to know if this breadth–focused conception leads to good educational choices.

Although this study provided some hints about the ultimate result of my research, the autobiography format had several disadvantages. First, though it allowed many students to respond, because they were fully in control of what was written about, there was no way to probe understanding. The focus of the analysis therefore, had to be on what students chose to mention rather than the cognitive aspects of their conceptions. Second, the ability to analyze a large number of autobiographies is less useful in qualitative work: it was more useful at that stage to analyze deeply and sample students who’s attributes contributed most to the researcher’s tentative understanding [18]. Still, as one of the few studies to probe CS conceptions for students later in the major, it was an important foundation for my hypothesis that students conceptions change throughout a student's time in a major.

2.4.4 Summary

Previous research into student field conceptions had several interesting results and affected my expectations for my research. From the studies in engineering and other fields, I expected students to have a evolving conception of their field which would be different from what their professors might envision. In my interviews, students did describe their views changing views but were very trusting of the CS curriculum (in constrast to Nespor's management students). From the work in pre–college conceptions of CS, I expected students to enter the CS major with at least some confusion about the field. Students did talk about their early views changing. From the work in later college conceptions of CS, I expected to see a broad view of CS in at least some students. In my interviews, I saw a broad view in some but not all students. Overall, this work provided direction for my initial interview approach by giving me some idea what to expect.

2.5 Conceptions of CS in High School Students

Prior to the research outlined in the document, I conducted a study of high school students' conceptions of CS. This study provided clear evidence that students before formal training in CS have large misconceptions and these misconceptions can be difficult to dislodge. The high school group that was studied was unusual: all the students had experienced a great variety of interesting CS interventions: fun approaches designed to focus both on programming and the innovative potential of CS [8], but most of the students had never taken a 'formal' CS class.

The study method consisted of four parts:

1. *Concept Map Instruction.* We gave the students a one hour introduction to concept maps, based on the instructions in [61]. Finally, after the hour long

introduction, the students were given twenty minutes to build a concept map about Computer Science.

2. *Pre–Interviews.* After the students build the concept maps, individual students were called into interview. The interview focused initially on the student's concept map and asked them to explain their reasoning, with particular attention to areas that suggested misconceptions. Then the interviewer presented a few example activities and asked if these activities are part of Computer Science and if so where they fit on the student's concept maps. Finally, the interviewer asked some questions about how the student might interest a friend in Computer Science and what attributes the participant considered essential for success in Computer Science.

3. *Class.* A week after the interview, the students attended two 4-hour class sessions (1 week apart) that attempted to further elaborate their concepts of Computer Science.

4. *Post–Interviews.* A week after the last class session, the students were asked to build a second concept map about Computer Science. They were given a copy of their original concept map to use as a reference if they wished. After building the map, they were interviewed following a similar script to the pre–interviews. As part of the post–interview, the students were asked to compare and contrast their concept map of four weeks ago with their current one.

It was clear from both from the concept maps and interviews that students had significant potentially problematic conceptions about CS. For example, students frequently believed that CS could train students for non–CS jobs that simply involved computers. For example:

Interviewer: [What would you say if someone asked] "If I were to get a degree in Computer Science, could it be my job to use photoshop professionally?"

> Student: I'm sure. I mean Pixar and all the Disney companies they are using digital art media now. All their movies are digital pretty much. Marketing too there's a lot of digital applications to design marketing advertising sorts of things.

As in the alternative conception literature [79], student conceptions colored their perceptions. Students frequently remarked about a presentation they had received about how CS could be used in medicine. However in the students' interpretation, CS people were responsible for using computers in places like hospitals.

Futhermore, there seemed to be ways in which student conceptions of CS influenced their educational choices. One of the students we interviewed was considering enrolling in Computer Science at Georgia Tech as part of the Digital Media specialization of CS. Contrasting this to a traditional art degree, he/she said:

> Student: You could probably get more into graphics and creating art with the computer and animating things. Where as in the art [program] you might be more dealing with the pure painting, sketching, sculpting.

Though certainly one could use the understanding of computer graphics and similar topics in the CS degree to create innovative art, this student seems to be envisioning something more similar to a digital media program at an art school.

What made this situation even more concerning was that students did not, in general, discard their potentially conceptions after they were addressed in class. One problem was that there are few resources for understanding the real internal structure of the field of CS in a way beginning students can understand, outside of an expert explaining. Contrasting before and after concept maps (see Figure 1) the student's conception of CS has more recognizable categories, but within the categories there is

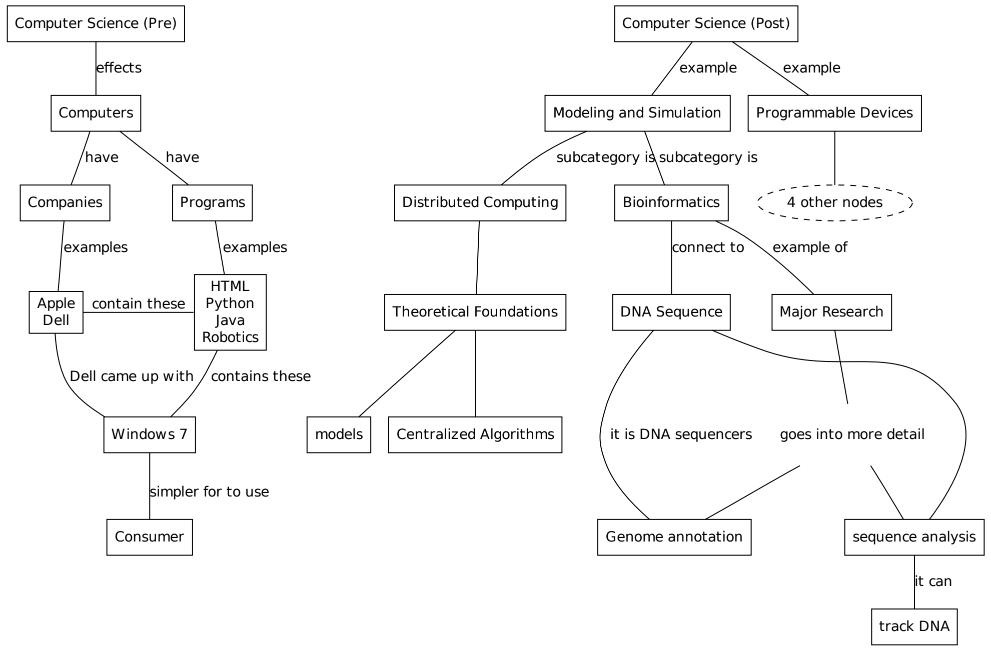


Figure 1: Pre and post concept maps about Computer Science (digitized from the handwritten form for clarity). You can see both the student has a more recognizable categories within CS, but that within these categories there is still confusion.

still confusion. Even for concepts students can readily understand conceptually, like distributed algorithms, the resources available outside of the classroom are so focused on experts that students can't understand what they discover on their own:

> Interviewer: So let's look at your modeling and simulation concept categories. First off, you have under there distributed computing. What is it?
>
> Student: It's kind of like um when I researched it it was kind of like finding stuff kind of I looked on a couple I think I looked on one website and learned a little about it I thought it was like technical part of computers kinda...
>
> Interviewer: So you said theoretical foundations and under that you put centralized algorithms and models. So explain what's a theoretical foundation, and why are centralized algorithms and models beneath that?
>
> Student: Well I didn't really know that much about theoretical foundations. So when I kinda went under that these were the categories that were underneath it. I really don't know what a centralized algorithm is but I wanted to take some more time to learn that... it looked interesting.

It is from this study that I took my some of my key hypotheses:

- Students may have potentially problematic conceptions about CS.
- These potentially problematic conceptions influence their educational choices.
- These potentially problematic conceptions are resistant to change.

There were obviously significant differences between the students in this preliminary study and the groups I studied in Study 1 and Study 2. These students (in general) did not intend to study CS professionally. They were high–school students and therefore are in a very different environment that CS majors are. Based on the results of Study 1 and Study 2, the environment and courses in CS program do seem change student conceptions from those that I encountered in high school students.

2.6 Summary

This chapter reviewed related work in five main areas. Here is a summary of the most important points to take away:

1. *Student Conceptions*

   This section reviewed three bodies of literature: epistemology of science research, alternative conceptions research, and research into the relationship between student field conceptions and learning content. All three discussed the issue that students conceptions are likely to be slow to change and designing educational interventions is difficult. Each area had unique insights however.

   Some of the methodology of the epistemology of science research is similar to my own. Epistemology of science is about helping students understand the scientific process. Similar to conceptions of the field of CS, views of the scientific process are difficult to elicit because students and experts think very differently. Similar to Study 1 and Study 2, qualitative approaches are often used to elicit student views in an open–ended way and then these open–ended views are used to develop survey instruments with constrained choices [2, 67]. In that sense, the methods of the epistemology of science literature is close to my own approach.

   Alternative conception research was the original basis for my work (and why I call student views of the field of CS ‘conceptions’. The basic premise is that students come to the classroom with existing ideas about how the world works called alternative conceptions. These conceptions are resistant to change, especially to classroom instruction techniques that simply present the ‘expert’ viewpoint [79]. Understanding alternative conceptions is important not just because they must be addressed, but because these existing conceptions provide the resources from which students necessarily construct new knowledge [73].

There are several lines of research that link field conceptions and learning. The educational literature of transfer suggests that understanding the reasoning for learning concepts is important in applying them beyond the classroom [3]. Donald suggests that individual disciplines have implicit approaches that students need to understand [26]. Songer and Linn connect field conceptions to individual learning strategies [74]. Each of these studies link specific conceptions to specific academic performances, but specific instances of field conceptions causing learning problems were rare in my interviews. A qualitative approach might not be sufficient to determine if problems with conceptions can affect learning in ways students' are unaware of.

2. *Differing Expert Definitions of a Field.*

   Both science and CS must deal with the fact that there is no single expert definition of the field that everyone can agree on [71, 22]. In both science and CS, there are concerns that certain definitions make it more difficult for some students to participate [5, 57]. This creates a difficult situation: educators wish to encourage a diversity of opinions while at the same time correcting conceptions that are problematical.

   The controversy makes it clear that this research cannot simply compare student conceptions to a single 'expert' view. The goal of this work is to focus on conceptions that cause students to make poor educational choices: therefore if students understand the basic curriculum of CS [70] and why it is considered important then their conceptions are likely accurate enough to prevent educational mistakes.

3. *Studies of Student Decision Making.*

   In my research students often used how enjoyable their class experiences were to make educational decisions. Based on psychological research, the subjective experience of enjoyment is based on several factors. The difficulty much be challenging, but allow the student to feel confident [21, 19]. The student must feel in control and connected to others [21]. Student enjoyment is also related to internalized societal forces that a student might not be aware of.

   Beyond the question of enjoyment, the Eccles model of achievement related choices suggests that students use enjoyment as only one of several factors influencing decisions like what course or major to select. Students consider what their likelihood of academic success and the values of different social groups [30]. Students also consider that each choice has a cost in terms of time and other opportunity. Although all these issues did come up at various times in our interviews, enjoyment of classes played a more dominant role than the Eccles model might suggest.

4. *Studies of Student Conceptions.*

   In engineering and science, qualitative studies have shown that students often perceive their majors in unexpected ways. Stevens's work [75] suggests that context strongly influences what skills students associate with the major, and therefore how successful they perceive themselves to be. Nespor argues that the structure of the disciplinary knowledge itself creates different social structures that change student experience. Both authors show the benefits of detailed qualitative analysis to uncover how academic structures affect student learning in unexpected ways.

   In CS education research, research on precollege students suggests that students do not (in general) have a good definition for what CS is. Qualitative studies on the community of CS majors suggests that students do have different CS conception, and these conceptions cause them to make different educational choices [57, 64]. Survey–based studies provide evidence that student conceptions do change [59, 6]. But although all these studies provide interesting hints, no study has attempted to understand how student conceptions in CS change across all four years of their undergraduate career.

5. *Conceptions of CS in High School Students.*

   Similar to previous work, my study of high school students provided evidence that high school students have poor conceptions of CS. Even when problematic aspects of students' were address directly in class, students maintained problematic conceptions of CS. At least in some cases these conceptions seemed to influence student educational decisions. Though this study was not explicitly designed with the idea of field–level conceptions, it provided a starting point for my proposed work.

In this chapter, I have reviewed research related to conceptions, both in education in general as well as within CS specifically. This research is what underlies my argument that a cognitive view of field–level conceptions is a fruitful perspective. In the next chapter, I will focus on method: both the research that underlies the method I chose and design of two studies to elicit student conceptions of CS.

CHAPTER III

STUDY DESIGN

In the previous chapters, I have argued that CS educators have a limited idea of what conceptions students have about the field of CS. My research attempts to advance this understanding by answering three research questions in two studies:

RQ1: What types of CS field conceptions exist in CS undergraduate students?[Study 1]

RQ2: Do potentially problematic CS conceptions affect student educational decisions?[Study 1]

RQ3: What is the prevalence of different kinds of conceptions among the CS major population?[Study 2]

Both studies focus on the understanding of undergraduate CS majors. Undergraduate CS majors were chosen because significant research has already been conducted on the views of high school students (e.g. [81, 36]). By looking at undergraduate students, I was able to find educational implications for the retention and success of students that already have some interest and familiarity with Computer Science.

The first study was an open–ended qualitative study designed to understand what conceptions exist and how they affect student educational decisions. The primary data for this study came from interviews with undergraduate CS majors. The student interviews were supplemented with written sources of information about CS as well as interviews with student advisors. The data sources were analyzed using grounded theory methods with the goal of producing an accurate understanding of the different CS conceptions students have.

As the theory of student CS conceptions was developed, I began to try assessing student conceptions on a larger scale using something like a survey. As part of the first study, I started to develop a open–ended survey instrument. As the study progressed, I changed the survey with the developing theory and tested how various survey methods elicited responses.

The second study is a larger study of students in one undergraduate CS class. I finished developing the survey instrument from Study 1 to assess conceptions across a large groups of students. The results were evaluated based on the theory built from the interviews. The goal of this study was to determine how common different conceptions of students are, for several groups of students.

3.1 Study 1: Method

This section focuses on the procedures for Study 1, which used interviews to study student conceptions of Computer Science. As part of this study, thirty three students and four counselors were interviewed and surveyed about their views of Computer Science. The various sources of data are discussed, including details of participants and recruitment. The section also includes details of the interview process, both at the beginning and end of the study. Appendix A contains the initial interview guide and initial survey (both of were changed as the study progressed).

The study was designed to follow the grounded theory qualitative methodology, but this section focuses on the procedure and not the reasons why. In the next section, the justification for grounded theory is discussed in detail. Details of the analysis are also presented in that section.

3.1.1 Data Sources

In accordance with grounded theory [18], this study analyzed several different data sources in its analysis of student conceptions of CS.

3.1.1.1 Written Materials.

I looked at several written sources as possible influences on student conceptions of CS. I did not explicitly analyze these sources using grounded theory analysis. I familiarized myself with them in order to understand student discussion:

1. Departmental materials: This category includes the departmental website, promotional materials, and the description of the major in the course catalog.

2. Major courses and requirements: The relationship between courses and other curriculum features and CS is both explicit and implied. The course catalog has explicit descriptions of courses and why they are important. But students are also likely to infer what is important in the field by looking at curriculum requirements including non–major courses or departmental application processes [75].

The websites and course catalogs I examined were the ones from the three schools where I conducted interviews: Georgia Tech, Duke University, and Spelman College.

3.1.1.2 Interviews with CS Counselors.

People in the CS department who advise students have some understanding of common CS conceptions students have and how those conceptions influence student educational decisions. At Georgia Tech, students must speak to a counselor before switching majors from CS, so the counselors were likely to have experience with conceptions that cause educational problems. I interviewed four Georgia Tech counselors to understand what conceptions counselors notice in students and what sort of questions students ask.

In my interviews, counselors did not have their own theories about how student conceptions of CS change. They did not have formal CS training and were hesitant to make generalizations about CS. However, they did have experience with students' educational decision making process. I found my discussion with counselors to be very valuable in terms of anticipating the interesting student decision making behavior that I found in the interviews.

3.1.1.3 Interviews with Students

I interviewed thirty three students about their conceptions of CS and how those conceptions have changed over time (see Table 2 for an overview). I interviewed students at three schools: Georgia Tech, Duke University, and Spelman College. Spelman is a historically black college for women, and as such is demographically different from Georgia Tech on a variety of dimensions. Duke is demographically similar to Geogia Tech, but is not an engineering focused school. All schools are four year programs with curricula in alignment with the standard [10].

Recruitment was done through presentations in CS classes. Students were asked to volunteer and offered a gift certificate to compensate them for participating. The students' contact information would be recorded and demographic information was collected. I also obtained permission from students to get course grade information (only for Georgia Tech students, due the difficulty of getting review board approval for grade information).

3.1.1.4 Sampling

The grounded theory approach includes the process of theoretical sampling [11]. Theoretical sampling rests on the idea that a researcher in a qualitative study does not initially know what factors will turn out to be important. Therefore the researcher cannot plan in advance how the population ought to be sampled. The researcher selects a small initial starting population to interview and plans to select further candidates as the theory is developed. Thus later interviews are chosen on the basis of the theory (which is why it is called 'theoretical' sampling). I used theoretical sampling in Study 1 as much as possible, although some factors had to be planned for in

Table 2: Summary of interviews. Id numbers are used to identify individual quotes.

Id	Year	Conception	School
P4	after-undergrad	theory	Georgia Tech
P5	counselor		Georgia Tech
P6	counselor		Georgia Tech
P7	counselor		Georgia Tech
P8	after-undergrad	broad	Georgia Tech
P9	senior	theory (maybe)	Georgia Tech
P10	after-undergrad	theory	Georgia Tech
P11	sophomore	programming-centric	Georgia Tech
P12	sophomore	theory	Georgia Tech
P13	sophomore	broad	Georgia Tech
P14	sophomore	programming-centric	Georgia Tech
P15	counselor		Georgia Tech
P16	sophomore	programming-centric	Georgia Tech
P17	senior	programming-centric	Georgia Tech
P18	junior	theory	Georgia Tech
P19	senior	broad	Georgia Tech
P20	junior	programming-centric	Georgia Tech
P21	sophomore	just-programming	Georgia Tech
P22	junior	broad	Georgia Tech
P23	sophomore	programming-centric	Georgia Tech
P24	junior	theory	Georgia Tech
P25	junior	unknown	Georgia Tech
P26	senior	broad	Georgia Tech
P27	freshman	programming-centric	Georgia Tech
P28	freshman	programming-centric	Duke
P29	freshman	broad	Spelman
P30	senior	broad	Spelman
P31	freshman	broad	Spelman
P32	sophomore	broad	Spelman
P33	freshman	programming-centric	Spelman
P34	senior	programming-centric	Spelman
P35	sophomore	programming-centric	Spelman
P36	junior	theory	Duke
P37	junior	programming-centric	Duke
P39	junior	theory	Duke
P40	junior	broad	Duke
P41	junior	theory	Duke

advance. I discuss the factors that turned out to be useful in selecting candidates in Section 3.2.1.2.

There were three categories of variation I had to plan for in advance because of logistical difficulties and IRB requirements:

1. *Time in Major.* One of the main hypotheses of this research is that conceptions change over time. I interviewed students at all four years of their CS degree.

2. *School.* Because the populations schools draw from is different, and because curricula does vary between schools, I interviewed students from three different schools: Georgia Tech, Duke University, Spelman College.

3. *Level of Academic Success.* Because one of the goals of this study was to connect conceptions with academic success, and because there is reason to think that level of academic success might affect conceptions, I interviewed to look at students with varying levels of success in their classes. I used instructor–evaluated student performance in class as a proxy for overall academic success because of the difficultly of working with long–term student data like transcripts. I divided students into very approximate top-3rd, middle-3rd, and bottom-3rd, and then attempted to select students for interviews with different groups represented. Grade information was only provided for Georgia Tech students.

We also recruited students with a questionnaire that asked about race, gender, and other data we felt might be useful in selecting participants based on the emerging theory.

I made a special effort to recruit students from traditionally underrepresented groups in CS. Selecting students from underrepresented groups was difficult because, in general, these students were less likely to volunteer after class presentations. Still, by interviewing almost all underrepresented students who volunteered and selecting

Spelman College as one of the schools in the study, students from traditionally underrepresented groups made up more than a third of my student interviews.

3.1.1.5 Preliminary Survey Instrument

Students were asked, as part of the interview process, to fill out a preliminary version of the survey instrument being developed for the second study. The survey provided an opportunity to validate that a conception elicited during interviews was consistent with the responses to the survey. It also provided an opportunity to test potential survey questions for Study 2 and see responses.

An important question is whether the survey should be administered before or after the interviews with students. When I begin interviewing students, the survey was to be administered after the interview process: this provides the least risk of specific questions on the survey guiding student thinking and corrupting the open–ended interview responses. As I felt confident that the survey was not biasing student responses, I began administering the survey before the interviews. That let me test that the survey questions are understandable on their own without the interviewer. In both cases, I had the participants think aloud as they fill out the survey to ensure that the questions were being interpreted as intended.

3.1.2 Interview Method

At a high level, the goals of the student interviews were:

1. Determine a students conception of CS, how the student came to the conception, and if the conception is potentially problematic or productive.

2. Determine if a student feels their conception has changed, and if so how and why.

3. Determine how a student conception is influencing educational choices, and (if the student's conception has changed) how previous conceptions influenced

educational decisions.

Interviews generally took between forty–five minutes and one hour. The initial interview guide for the interviews can be seen in the appendix. Initially, the interviews tended to follow a question and answer format similar to those given in the guide. With greater experience, the interviews tended to follow more naturally by asking the student questions about their time in the CS major. I would ask the student how they ended up initially majoring in CS and ask them to make a timeline of the courses they had taken. At that point, the student would tend to volunteer information about the courses, which would tend to lead into a discussion of good verses bad courses and educational decisions the student made. The student would usually bring up long term goals and I would pursue that line of questioning.

If the field of CS did not come up during that process, I would bring it up explicitly and briefly ask about definitions (asking students to explicitly define CS, it turned out, was not usually productive). To test the boundaries of a students understanding, I might ask the student:

1. Whether they considered a particular course they mentioned was completely CS, or a mix of CS and some other field

2. For an example of a “really Computer Sciencey” job

3. If they considered someone doing a particular job to be “doing Computer Science”

To test for potentially problematic conceptions, I would ask students to talk about the content of courses they were looking forward to taking or had taken. I would also ask about how students selected particular courses, or what they were looking forward to learning before graduation. For students with concreate ideas about future courses, this could naturally move into a discussion of the subdisciplines of CS. I would also ask them if their view of CS had changed and if they recalled any problems with their previous understanding (that line of questions, in contrast to asking about definitions, could prove quite fruitful).

Overall, I found that allowing students to tell their own stories about their experiences in CS could allow me to elicit a good understanding of their conceptions without leading questioning.

Interviews for CS counselors tended to focus more on their experiences couneling and questions students usually ask (beyond those about graduation requirements). Beyond that, the interview focused on eliciting counselors' theories on students' conceptions: what conceptions counselors think are common in students, how counselors think student conceptions develop, and how counselors think conceptions affect student education decisions.

3.1.2.1 Checks to Ensure Validity

When attempting to understand student conceptions, there is a risk of misinterpretation and bias. This is a common problem in qualitative research; even when participants and researchers act in good faith, it is difficult to understand when backgrounds and assumptions are different. There are a variety of techniques to mitigate this risk [53]. I used two: triangulation from multiple data sources and member checking.

The survey instrument allowed me to use triangulation: comparing data from two similar sources to verify that interpretations of one source are consistent with the other. Given that the eventual final interview process involved fairly open ended questions, the risk of leading is reduced. The survey further reduced the risk by asking similar questions to the interview process, but avoiding accidental leading that may occur with expressions and other accidental social cues. But unfortunately, most of the disparities between surveys and interview responses were caused by vague student answers that couldn't be followed up in the survey.

With three students, I also used member checking: providing the student with my analysis of their conception, and asking for feedback. In one case, I contacted research participant after the initial interview and reinterviewed them with the my interpretation of their viewpoint. In two others, I presented my analysis after the regular interview process concluded. With member checking, one needs to be aware that participants may be likely to agree with researchers, and therefore not simply take bland agreement at face value ([11], pg. 111).

Both types of checking helped, but there were problems. The survey data sometimes confirmed the analysis but often there was little in the results that could be used. For much of the interview process, the survey lagged behind the theory because it is much easier to elicit conceptions in interviews than via a survey. For the member checked students, they all confirmed my analysis but I found it difficult to distinguish "bland agreement" from genuine agreement, especially about a topic like CS where students do not have well thought out opinions.

3.1.3 Summary

This section has introduced the procedures for Study 1. It was an interview–based study of 33 CS students from 3 different CS programs. The interview was semi–structured with a focus on understanding students conceptions of computer science, how students make educational decisions, and how student conceptions have changed. In the next section, I discuss the justification for the design and the process of analysis.

3.2 Study 1: Study Design and Analysis

This section describes how grounded theory informed the design for Study 1 and how it was used in the analysis of data. I highlight two main differences between a grounded theory approach and other interview based qualitative approaches and describe how they played out in the study. Then I describe the process of grounded theory analysis and how I used it to develop my theory.

3.2.1 Grounded Theory in Study Design

I argue in Chapter 2 that because CS educators do not have experience eliciting student conceptions, a qualitative approach is necessary. Many different qualitative approaches have been used in education [72]. Some qualitative approaches are not suitable for interviewing a variety of students about CS (e.g. ethnographic observation or case studies). I believe interviews are the easiest and best ways to understand how students conceptualize CS. However within the framework of student interviews, there are still several potential approaches.

Grounded theory had two main advantages for this study. First, grounded theory emphasizes developing a theory grounded in the participants and not testing a preconceived structure. Second, the process of theoretical sampling in which new participants are selected based on the emerging theory. I discuss both in the sections below.

3.2.1.1 Emphasis on Developing a Theory Grounded in the Participants

One of differences between grounded theory and other approaches is its requirement that the theory emerge from the participants. The researcher is encouraged to come to the interviews (to the extent possible) as if they had no preconceptions about the topic and focus on how the participants reason. Though (as discussed in Chapter 2) the CS education community does have some ideas about likely conceptions, there was also a high likelihood that the way students thinking about CS would be different from what was expected. Grounded theory gives the researcher the freedom to pursue what concepts naturally arise; to understand students conceptions of CS we need to be prepared to approach the issue on the students' terms.

In practice, the flexibility to base the emerging theory on the participants turned out to be very important. There were terminology differences between students and what experts might say. For example, 'theory' as used by students tended to refer to any mathematical aspect of any CS course, not a discrete subdiscipline within CS. There was also focus differences – a large part of the theory developed to describe student educational decisions came from student discussions of classes being fun on enjoyable: things that did not form a large part of our framework as we begun.

Grounded theory is also focused on developing a 'theory' from the participants. As you read the later sections, it should be clear that the results are intended to be more than simply a description of what students know and do. The theory attempts to explain why students act the way they do (e.g. why they choose their courses in a seemingly arbitrary way). Like any scientific theory, there is a possibility that it might not be true (or that it might not be true at different schools, or for different students). The goal of the process is to ensure that the theory is as likely as possible by 'grounding' it in the qualitative data. The fact that grounded theory encourages the research to attempt to develop these theories and explanations makes it an excellent process for understanding student conceptions.

3.2.1.2 Theoretical Sampling

Theoretical sampling is the process of selecting later study participants based on the evolving theory, and it is an important part of the grounded theory method. It was useful to have the ability to select participants as the theory developed. The purpose of sampling was not to interview a statistically representative group of students, but to interview students from a wide variety of backgrounds and develop the theory. In accordance with theoretical sampling, I did not initially plan exactly how many students I would interview or what groups each individual person will be drawn from. The interview process continued until new interviews did not add new concepts (see the discussion of saturation below).

Selecting which courses to recruit in very much followed from the developing theory. I selected an introductory course for my initial interviews because it occurred at a point when students had to make decisions about upcoming courses. From there, I selected courses based on the theory. For example, because students intending to pursue the People thread often had different conceptions of CS, I selected a course in the People thread. Not all instructors were willing to spend class time for study recruitment, but I was usually able to find a course that met my needs.

A variable that turned out to be interesting was a question I asked about long–term career goals. Students did not usually have a very concrete idea, but even their vague answers turned out to be varied and interesting. Students who selected unusual career goals (e.g. becoming a CS teacher, working for the Navy) generally had interesting forces behind those directions. The ability to theoretically sample allowed me to select students based on responses like this, rather than selecting a particular quota from a particular class for example.

Another set of variables that turned out to be valuable was underrepresented groups — both women and underrepresented minorities. Students from underrepresented groups often had less pre–college experience with CS and had more concerns about CS and their place within it. Although this study cannot compare underrepresented groups because small sample sizes makes generalizations about groups impossible, the interviews with these students shed a great deal of light on the student decision making process.

Finally, theoretically sampling allowed me to avoid a group that might have seemed initially interesting – freshman. Even sophomores were very vague about CS and the educational decisions they might make. It quickly became clear when interviewing freshman that they had not thought very much about CS beyond their initial CS course. Sophomores and juniors were at the stage that they had to make interesting decisions about their goals in CS. I would have preferred to get more seniors – because I generally recruited in larger classes to get greater choice in my participants, seniors (who are normally taking the final, smaller, courses in their specializations) were rarer.

3.2.1.3 Reflections on the Study Design

Overall, I am definitely glad that I chose a qualitative approach for this study. Qualitative approaches are best when you want to pursue unanticipated questions. When this study began, I did not have a clear idea of how categorize student conceptions of Computer Science or how students used those conceptions in their reasoning. If I had attempted to use a formal survey or explicit experiment without that understanding, I would have almost certainly have asked the wrong questions. Even with a qualitative approach, it took many interviews until the theory began to approach saturation and my questions began to focus on issues that really mattered to students.

As far as grounded theory in particular, I was pleased with how grounded theory encouraged me to approach the interviews without a particular conception in mind. The explicit goal of grounded theory, to develop a theory, also encouraged me to analyze deeper. Even when doing something as simple as selecting the next interview candidate, the process makes clear the expectation that a theory should be developing.

3.2.2 Analysis

For the more structured aspects of the interviews, analysis was straightforward. Determining if a student could correctly reflect on the content of their courses or anticipate the content of future courses was simply a matter of asking the right questions and pursing ambiguous answers. Although I often had to approach the same question from several different directions, usually by the end I was confidently about a students ability to reason about the curriculum.

The goal of the grounded theory analysis was to understand the common threads that underlie different student conceptions. Each conception arises from a student's unique experience. The goal of the analysis is to develop a theory, grounded in each student's individual experiences, that describes how conceptions develop, change, and influence educational choices.

A grounded theory is based off careful line–by–line analysis of data sources that are methodically abstracted into categories and theories. In this case, the primary sources were transcripts of interviews. First the researcher develops initial codes that describe what is being expressed in each line of the data [11]. Second, the researcher goes back through the body of research accumulated and selects 'focused' codes that explain larger segments of the data. Third, the focused codes are abstracted into categories in a tentative theory that is then checked against other parts of the data to test its explanatory power. There are several techniques to help the researcher attempt to develop the categories in this larger theory including axial coding [18], theoretical coding [11], and situational maps [13]. Tentative theories and ideas are written in memos. The interview/analysis process continues until "saturation": when additional interviews do not further elaborate the theory.

There are several different variations of the grounded theory process [11]. My processes was heavily influenced by Charmaz's approach [11], as opposed to Corbin and Strauss [18]. The approaches are similar in the initial stages, but Charmaz suggests a variety of alternatives for the later analysis process. In Corbin and Strauss, the final result of a theory is always a single category that subsumes all others that represents the main theme of the research [18, p. 266]. In Charmaz's approach [11, p. 115-121], a single category is not the final goal — instead the researcher attempts to integrate the categories into a cohesive theory but a single category is not necessarily the only result.

To illustrate the analysis process, the following sections will provide a few examples of the process.

3.2.2.1 Turning Transcripts Into Codes and Focused Codes

> "Software engineering, it looked like it was more offered by lower tier colleges…I figured, even though I don't really like theory, there's probably some stuff in it that's useful and probably would make me a better programmer overall. So I figured I'll stick with Computer Science but try to take more practical side of classes."
>
> —P12

One of the things I coded about this quote was the student's decision to rely on the reputation of the CS curriculum, despite negative experiences with CS theory in high school. The initial coding was abstracted into the focused code "trust in the curriculum," which included several other students who specifically mentioned they chose particular specializations because the specializations were considered "traditional" CS. When comparing student responses, I saw similar but different responses: students who argued that specializations were unimportant because they knew the curriculum would cover any really essential CS topics. I created a superordinate code about how students assume the CS curriculum will teach them everything they need to know, even when they often don't know what they really want from CS. Eventually, this code became called "abdicating responsibility to the curriculum."

3.2.2.2 Revising the Theory

Throughout the grounded theory process, there are tentative theories. These theories are being put to the test in later interviews, and during analysis processes like situational analysis (see the next section). Usually, initial generalizations turn out to not to universally true. Contradictions triggered me to go back to the source data and to become more nuanced which moves the grounded theory forward.

For example, at one point in the analysis, the idea that enjoying classes was the main determinant for student educational decisions was a major part of the tentative theory. There were a variety of codes having to do with student enjoyment like "frustration causing reconsideration", "enjoying classes involved in educational decisions", and "just choosing what sounds 'interesting' ". But, by looking at the counts of each code, other codes like "parental involvement" were almost as common. That seemed wrong insofar as enjoyment seemed to figure greatly into student decisions, but parental involvement definitely seemed more peripheral. It was clear that something about student enjoyment was being missed, so I went back through the codes and attempted to understand the role of enjoyment more clearly.

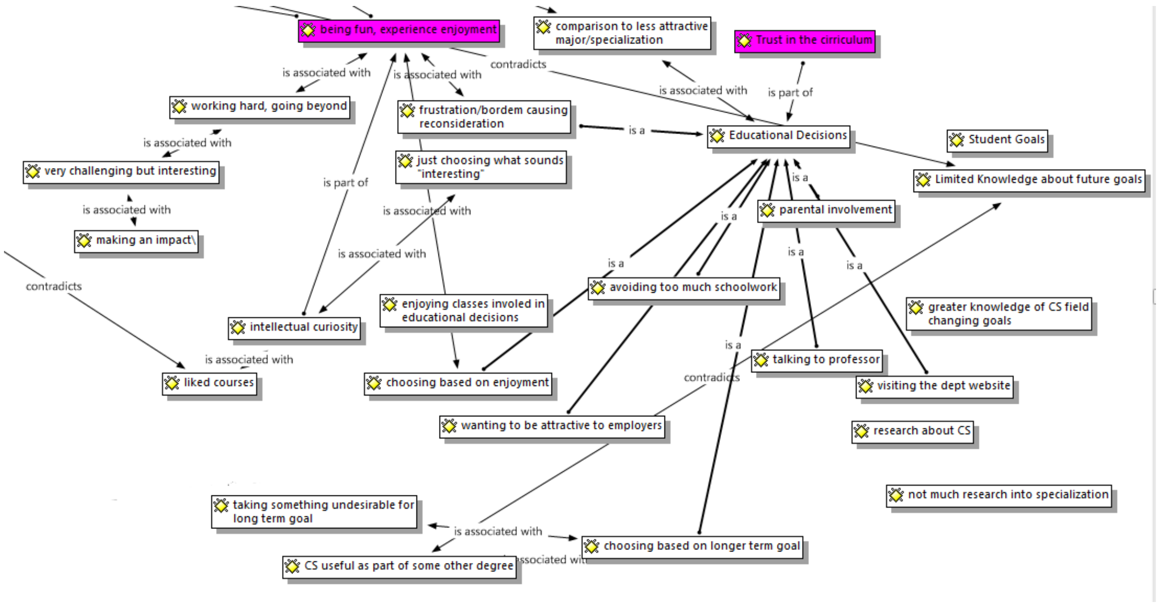
> "I got [to my architecture course] and I was like, 'I don't understand any of this. I don't really like it.' So I switched to [the people specialization] which I like a lot more. I have a lot of interest in psychology. I'm actually getting a certificate in social and personality psychology . So I switched.
>
> And I was kind of hesitant at first when I talked to my — the advisor in the CS department, because I was like, 'This - that really isn't as good for a career in video game animation and special effects or whatever I decided to go into.' She was like, 'It's not.' "
>
> —P19

Quotes like the one above made me realize that there were different kinds of enjoyment experiences. Weaker positive experiences encourage students to explore. But when a student has a very negative experience in a course, it often triggered them to make an educational decision. Then when they're making that decision, they solicit advice from parents or advisers (as in the quote above). But the experience triggering the sudden reorinetation is the emotional experience of enjoyment, which is why enjoyment seemed abstractly to be more important than, for example, parental advice. This idea eventually was revised even further into the overall idea of student educational decisions that is discussed in Chapter 5.

3.2.2.3 Situational Maps

One technique I made use of to develop the grounded theory was situational maps as described by Clarke [13]. The process of coding produces a huge number of codes: it is difficult at times even to keep track of them. In the mapping process, I would have several diagrams containing the main elements of the evolving theory, and place the main codes on the map and try to begin relating the various parts of the map. Related codes are often organized near each other. Codes that have important relationships in the evolving theory are linked by lines and arrows.

The map itself tends to encourage 'cleaning' of the codes: although the maps are large (often bigger than a single computer screen) there is a limit to how many codes can be displayed. Similar codes are combined, often making better more general codes. Parts of the emerging theory that seem to be contradictory are made more obvious (e.g. students taking courses they expect to dislike, in contrast to most of their peers).



being fun, experience enjoyment
comparison to less attractive major/specialization
Trust in the cirriculum
is associated with
is associated with
contradicts
is associated with
is part of
working hard, going beyond
frustration/bordem causing reconsideration
is a
Educational Decisions
Student Goals
is associated with
very challenging but interesting
just choosing what sounds "interesting"
is part of
is a
Limited Knowledge about future goals
is associated with
parental involvement
is a
making an impact\
is associated with
is a
is a
is a
contradicts
avoiding too much schoolwork
enjoying classes involed in educational decisions
greater knowledge of CS field changing goals
intellectual curiosity
is a
talking to professor
is associated with
liked courses
choosing based on enjoyment
contradicts
visiting the dept website
wanting to be attractive to employers
research about CS
not much research into specialization
taking something undesirable for long term goal
is associated with
choosing based on longer term goal
CS useful as part of some other degree
ssociated wi

Figure 2: A sample situational map. Not every code represented on this map was equally common or important — only some are discussed in the final theory.

You can see an example part of the situational map that focused on student educational decisions in Figure 2. I used this map to refine my own theory of student educational decisions and organize my codes — it is not a diagram intended to be useful to others. Some of the codes represented here did not become a key part of the final theory (for example, ‘avoiding too much schoolwork’ which is an obvious driver of student educational decisions but was only occasionally important in student reasoning).

3.2.2.4 Analysis of the Preliminary Survey Instrument

The primary goal of the preliminary survey instrument was to test ideas for a survey to elicit conceptions that can be used at a larger scale than individual interviews. As the grounded theory develops and common conceptions of CS were identified, the survey was modified to incorporate questions that work well to elicit key differences in student conceptions. Over time, the survey moved from attempting to identify specific problematic conceptions about future or past courses towards determining which of the 3 conceptions discussed in Chapter 4 students had.

One of the components of the initial survey instrument was the use of a Computer Science concept map. This is based on the research by Novak and Gowin [61], that identifies a concept map as an excellent way to both allows students freedom in expressing complicated conceptions but also produces an artifact that is easy to evaluate for conceptual flaws. Concept maps worked well for me in my interviews with high school students (described in Chapter 2).

In my college level interviews, the concept map occasionally worked very well but also had its problems. Very often students near the beginning of CS could not make a concept map at all. Students at all levels did not like filling out the concept map, which suggested it might not work for a large scale survey. It is one thing when an interviewer is right there watching, but students might be tempted to avoid the difficult mental work required to build a concept map on a large scale survey. Students also interpreted the concept map instructions slightly differently, so comparison could not always be made between maps. The concept map was an interesting exercise that occasionally produced very interesting results, but it was not a good candidate for the final survey.

Another question type that had mixed utility was open ended 1-3 sentence response questions. Questions of this sort included “How would you define Computer Science” or “Can you think of a concrete thing you have learned (or will learn) in your Computer Architecture course that you think will be useful even if you don’t ever design a piece of hardware?” In all of these questions, students would occasionally respond in a detailed way that provided a good view of their conception of CS. But many times, students would answer in a way that was impossible to interpret.

By the end of Study 1, very few questions had been identified that would reliably elicit student conceptions. Part of the problem may have been that, towards the end of the study as the overall theory approached saturation, not enough focus was placed on the revising the survey in innovative ways. Therefore, for Study 2 it was decided to take a new approach to the survey and add on a thinkaloud portion to validate the approach.

3.2.2.5 Reflections on Analysis

The basic process of my grounded theory analysis was:

1. Go through the written interview transcript, coding line–by–line and developing new codes as necessary

2. When similarities are seen across interviews, combine similar codes to broader focused codes

3. Based on the codes, the interviews themselves, or techniques like the situational maps, refine the codes into a tentative theory about what is happening

4. Review the data based on the tentative theory (and do more interviews, chosen with theoretical sampling) as contradictions and nuances are found revise the tentative theory
5. When new interviews do not produce changes to the theory, it is considered saturated and can be presented

Overall the grounded theory process was excellent in producing a theory of student conceptions of CS and educational decisions. What problems occurred tended to be from modifications to the process (e.g. the thinkaloud survey which was less successful) or logistical constraints (e.g. sometimes it was necessary to schedule student interviews so closely it was not feasible to analyze them before the next interview). Although the resultant theories developed were different than I anticipated, I believe they are well grounded in students' reasoning processes.

3.2.3 Differences From Proposal

There are two main differences between the Study 1 presented here and the one I proposed:

1. *Less Focus on Textual Data.* The original proposed study planned to analyze introductory textbooks and college websites using a grounded theory approach. Although CS department curricula were examined to supplement interviews, there was no formal grounded theory analysis done. Given students' discussions I suspect that introductory textbooks would not have been a useful source for theory. Students did, however, use department websites and curricula extensively. A further study that analyzed these sources in detail and also allowed students to reason about CS with the departmental resources available to them would be an interesting supplement to the work presented here.

2. *Significantly Expanded Interviews.* The original proposed study estimated approximately 25 interviews at two schools. Study 1 consisted of 37 interviews (33 students, 4 counselors) at three schools. Duke University (the school that was added beyond the proposal) is interesting because although it is similar to Georgia Tech in terms of admissions requirements, in terms of curriculum it offers students many fewer options. Overall, I believe the extra interviews helped to get a better theory that is applicable to a broader group of CS students.

3.3 Study 2: Assessing Prevalence of CS Conceptions

In Study 2, the survey instrument, based on the theory developed in Study 1, was given to students. The goal of this study is to determine if it is possible to assess student conceptions with a simple survey and how common the conceptions elicited in the first study are. The survey instrument used in this study was tested using a thinkaloud protocol to ensure that the questions accurately elicit student conceptions. This study estimates how frequently each of the Study 1 conceptions occur and therefore how much instructors can expect students with different conceptions to be in their classes.

3.3.1 Design of the Survey Instrument

The main portion of the survey instrument was based on questions that proved fruitful in interviews: “Would you call a person who does [some activity] a Computer Scientist?” Students who viewed CS in a theoretical mathematical way would find researchers who proved characteristics of algorithms to be very good examples of Computer Scientists. Students who viewed CS as primarily about programming would see such researchers as on the border between CS and some other field. In the survey, students are given a list of activities and asked to rank how Computer Scientist–like they are on a scale from “Not Computer Science At All” to “A great example of someone who does Computer Science”. The main page of the survey can be see in

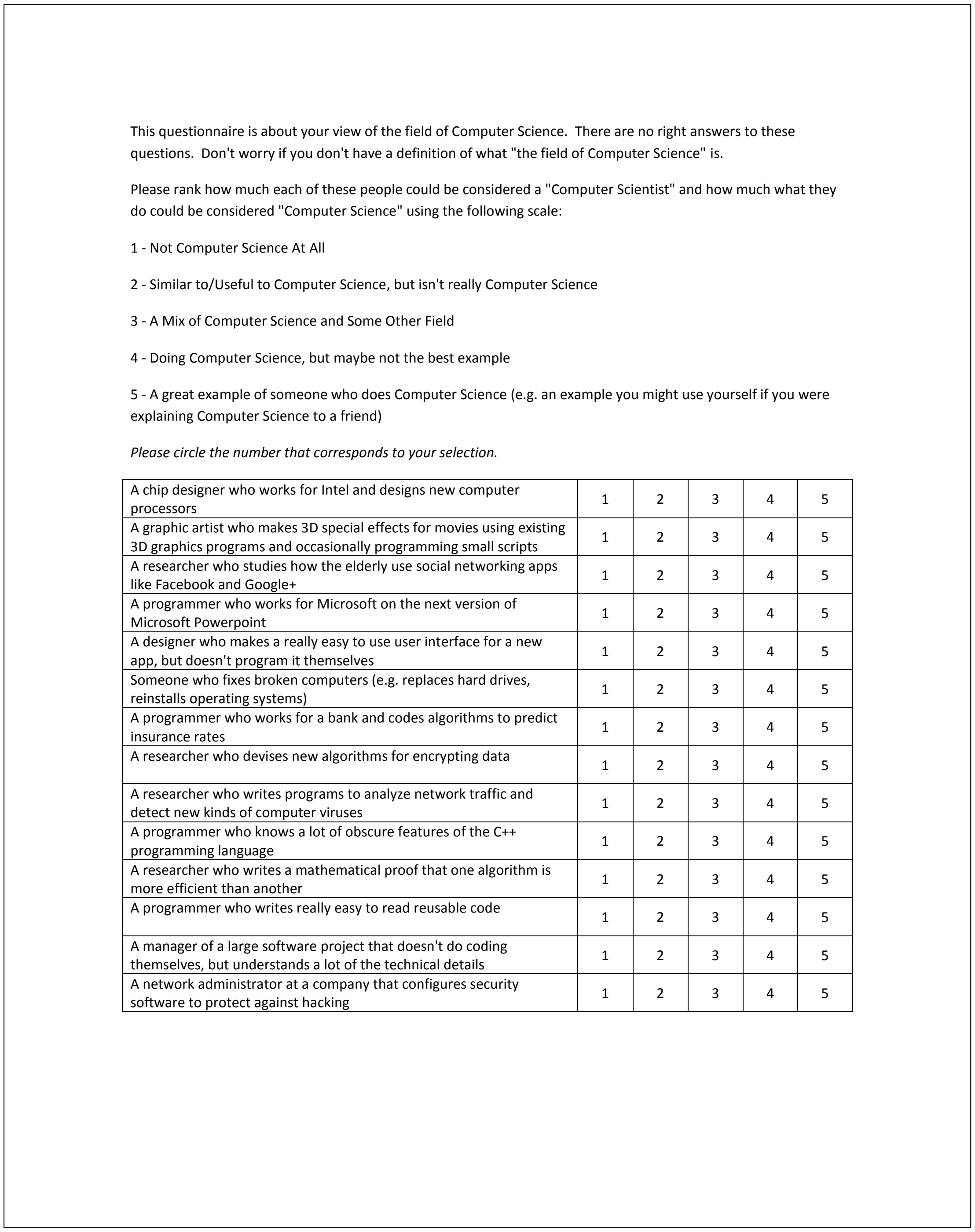
Figure 3.

The survey also asks some questions that can be answered with a few sentences. The survey also explicitly asks students to rate their agreement with descriptions of the three main conceptions identified in Chapter 4. You can see the complete (post thinkaloud) survey instrument (including demographic questions) in the appendix. Although open–ended questions have sometimes worked poorly in the past, they do provide an opportunity for students to provide more information about their conceptions which I can hopefully use to understand their conception if the first section produces inconsistent results.

3.3.2 Thinkaloud

To test the survey instrument, I recruited six Computer Science students through presentations in two different CS class. The students filled out the survey and thought aloud while I took notes on questions they had problems with or interpreted differently. Then, after the survey, I did shorter 30–minute interview on their views about Computer Science. After I determined my interpretation of the students view, I also told them my analysis and asked them to elaborate on anything that was wrong or that I left out.

For students who had a programming–centric conception of CS, the survey accurately measured their CS views. For students who had a more broad conception of CS, there were some problems. The broad view is the most difficult to assess because although most individuals with a broad view assert there are non–programming and non–theory areas of CS, each person tends to be a little different about what those areas are. I revised the survey to try and more accurately elicit the broad view. None of my thinkaloud participants had the theory view, but an informal test with a graduate student did assess the theory view correctly.



This questionnaire is about your view of the field of Computer Science. There are no right answers to these questions. Don't worry if you don't have a definition of what "the field of Computer Science" is.

Please rank how much each of these people could be considered a "Computer Scientist" and how much what they do could be considered "Computer Science" using the following scale:

1 - Not Computer Science At All

2 - Similar to/Useful to Computer Science, but isn't really Computer Science

3 - A Mix of Computer Science and Some Other Field

4 - Doing Computer Science, but maybe not the best example

5 - A great example of someone who does Computer Science (e.g. an example you might use yourself if you were explaining Computer Science to a friend)

Please circle the number that corresponds to your selection.

A chip designer who works for Intel and designs new computer processors	1	2	3	4	5
A graphic artist who makes 3D special effects for movies using existing 3D graphics programs and occasionally programming small scripts	1	2	3	4	5
A researcher who studies how the elderly use social networking apps like Facebook and Google+	1	2	3	4	5
A programmer who works for Microsoft on the next version of Microsoft Powerpoint	1	2	3	4	5
A designer who makes a really easy to use user interface for a new app, but doesn't program it themselves	1	2	3	4	5
Someone who fixes broken computers (e.g. replaces hard drives, reinstalls operating systems)	1	2	3	4	5
A programmer who works for a bank and codes algorithms to predict insurance rates	1	2	3	4	5
A researcher who devises new algorithms for encrypting data	1	2	3	4	5
A researcher who writes programs to analyze network traffic and detect new kinds of computer viruses	1	2	3	4	5
A programmer who knows a lot of obscure features of the C++ programming language	1	2	3	4	5
A researcher who writes a mathematical proof that one algorithm is more efficient than another	1	2	3	4	5
A programmer who writes really easy to read reusable code	1	2	3	4	5
A manager of a large software project that doesn't do coding themselves, but understands a lot of the technical details	1	2	3	4	5
A network administrator at a company that configures security software to protect against hacking	1	2	3	4	5

Figure 3: Main page of the survey for Study 2

3.3.3 Participants

Participants were students in a large size sophomore software engineering class. Of approximately 175 students in the class 103 agreed to participate in the study. Students were not compensated for participating in the study.

3.3.4 Analysis

The primary analysis was to determine student conceptions of CS, based on their survey responses. Students were assigned to a single conception based on their answers to the first questions, or as "uncategorized" if their answers and inconsistent with any single conception. I also went through the open ended questions and assigned students to groups based on that. The study also looked at the relationship between some other factors (race, gender, previous CS classes) and student conception.

A detailed discussion of the results and the statistical analysis performed on them is in Chapter 6.

3.3.5 Differences From Proposal

There are three main differences between the Study 2 presented here and the one I proposed:

1. *Slightly different source population.* I proposed to give the survey to 2 classes, each with 50-60 students. Instead, I gave the study to one class with 150 students.

2. *No compensation.* I proposed to give the students extra credit for participating in the study. The survey was shorter and more straightforward than I anticipated, and compensation causes complication, so I decided that extra credit was not necessary to ensure students did a reasonable job.

3. *Cohen's κ* . I anticipated that the survey would be primarily be open–ended questions evaluated with a rubric. For that approach, having multiple researchers and inter–rater reliability using *Cohen's κ* was appropriate. Based on Study 1, I decided that closed questions were actually better at eliciting conceptions. For the small number of open–ended questions that occur on the final survey, inter–rater reliability is not as important.

3.4 Summary

This chapter has outlined the method for two studies to understand student conceptions of CS. In order to know if student conceptions are important from an educational perspective, we need to understand what conceptions exist, if and or how they affect student educational decisions, and how prevalent these conceptions are in students.

Study 1 focused on what student conceptions exist and how they affect student educational decisions. Grounded theory was selected as the method because of its flexibility to explore unanticipated results and focus on developing a theory true to the understanding of participants. Interviews with student advisors, and interviews with students provided the data for the study. Theoretical sampling was used to determine who to interview as the theory develops. As part of the process, a survey instrument was tested to elicit conceptions of CS in a similar way to interviews.

Study 2 used questions similar to Study 1 to attempt to understand how prevalent different student conceptions of CS are. One CS class filled out the survey. The prevalence of the various conceptions observed in the classes give an approximate idea about the extent to which various conceptions of CS exist in the population.

This chapter outlined the design of two studies to address the research questions put forward in Chapter 1. Detailed examples of the materials for Study 1 can be found in Appendix A. The survey used in Study 2 can be found in Appendix B. The next three chapters discuss the results of the studies. Chapters 4 and 5 discuss the results of Study 1 (student conceptions of CS and student educational decisions respectively). Chapter 6 discusses the results of Study 2.

CHAPTER IV

CS FIELD CONCEPTIONS IN CS UNDERGRADUATE STUDENTS

This chapter presents the results of Study 1 as they relate to student conceptions of the field of CS. The theory presented here is the result of a grounded theory analysis of interviews with undergraduates in CS degree programs at three different schools. There are three main conception categories of CS I observed in undergraduates with more than a few courses of CS experience (sophomores and later):

1. *Theory–View: CS as Mathematical Study of Algorithms.* Students who held this view thought of CS as a primarily theoretical and mathematical discipline. The design of conceptually difficult algorithms was most central to CS, as were other mathematical ideas like Big O and NP–Completeness. Programming was viewed as useful but more peripheral to CS, and students often emphasized that CS could exist without any physical computer.

2. *Programming–View: CS as Programming–Centric but Including Supporting Subfields.* Students who held this view considered CS to be mainly about programming, but emphasized that other subfields were also necessary to do good programming. Writing programs to solve large and technically challenging problems was the central activity. Students with this view varied on how important non–programming subfields of CS were.

3. *Broad–View: CS as Having Many Different Subfields.* Students who held this view thought of CS as mix of many different computer–related subfields. Theory, Robotics, Programming, and (often many) others are all equally important

parts of a broad CS 'umbrella'. In this view, comparatively little knowledge was considered 'essential' to a Computer Scientist; students emphasized the differences between subfields and the freedom to pursue different paths.

Within each of these categories students had a range of views, encompassing both students with correct and incorrect views about their CS courses. I asked two kinds of questions — questions where students would "reflect" on courses they had already taken, and questions where students would "anticipate" the content of courses they had not yet taken. Recall from Chapter 2 that I judged a student's reflections and anticipations "correct" if the student identified content areas similar to what was actually in their school's curriculum. In addition to the three main categories described above, there was considerable variation between students about whether certain activities (e.g. designing user interfaces, communication skills) were part of Computer Science.

In this chapter, I will approach the issue of student conceptions of CS in four ways:

1. I will describe the three main conceptions in detail, and compare and contrast them.
2. I will discuss potentially problematic conceptions that exist within students of all the main conceptions.
3. I will discuss how students reflected on the conceptions changing throughout their undergraduate experience in CS.
4. I will address the question of whether curriculum has an effect on student conceptions, and describe differences between the three schools in Study 1.

4.1 A Theory: Three Main Conceptions of the Field of CS

This section presents three categories of student conceptions of the field of Computer Science. These categories of conceptions have some variation within them, and there are even a few students who seem to straddle the boundaries between two categories. However, it is reasonable to say that these three categories represent three distinct views of CS that are useful when understanding student views of the field.

4.1.1 Theory–View: CS as Mathematical Study of Algorithms

> " . . . how to program is I think not completely different but is very different from Computer Science in general. Because theory definitely is important - very, very important to Computer Science in sort of understanding the more theoretical aspects like what people are working on, like what are the constraints, where are the known problems, that sort of thing . . . So computer science, I feel like is much more actually theoretical and programming is just another skill essentially."
>
> —P39

The theory–view of CS focuses on the theoretical and mathematical aspects of CS as the most essential part of CS. 'Theory' as students used the word encompassed more than just the contents of a theory course: it includes the abstract portion of most CS courses (e.g. data structures, LL and LR grammars), but algorithms are frequently mentioned as the central idea. In this view, programming is a useful offshoot of the mathematics of CS, but it's clear that it is an application and not the core. Students with this conception would frequently emphasize that Computer Science exists beyond actual physical computers: they were the only group to mention that CS exists in puzzle games or algorithms humans execute in everyday life. Elsewhere in this document, I refer to students with this conception as theory–view students.

For theory–view students, CS was an academic discipline. All of them agreed it was possible to do programming without doing Computer Science. Some suggested that probably every professional programmer encountered hard problems and therefore was doing CS, while for others the programmer had to be working on a hard problem of theoretical interest (my term – students would likely say it was a problem that hadn't been solved before, or simply a problem that was really had to figure out).

It's important to note that this view of CS did not coincide with an interest in doing theoretically–oriented CS work. All of them agreed that theoretical CS was important to know, but none of these students were interested in pursuing theoretical CS as a career. Just like students with other conceptions of CS, they expressed frustration that proofs were difficult and not something they could see themselves doing long–term.

This group tended to be accurate in anticipating the contents of later courses. They varied in their relationship to pragmatic skills like knowing tools or specific programming languages. Some viewed these skills as part of CS, but just a less–central aspect of CS than theory. Others expressed that these skills were useful to know but weren't a part of Computer Science itself. They generally did not consider user interface design or communication skills to be part of Computer Science.

4.1.2 Programming–View: CS as Programming–Centric but Including Supporting Subfields

> "I'd say that computer science is a study; is a discipline, and that programming is how it takes form; how it's actually represented in the world. So I'd say programming is probably the end goal behind computer science, but - I mean, it's like comparing the study of automobiles to building a car. Well, you can contribute to the study of automobiles without ever actually building a car; many people do. But yes, we study it so we can

build them."

—P27

In programming–view, programming is central activity of Computer Science but it is supplemented by several subfields that do not directly involve programming. In this case, programming encompasses topics like data structures and the implementation of algorithms (for example, doing a project in a graphics class). But students acknowledge that a computer scientist must understand ideas like Big O, incomputablity, the structure of a processor, etc. However, the central activity of Computer Science is definitely programming: that is, someone proving something about an algorithm is doing a less Computer Science oriented activity than implementing that same algorithm. Elsewhere in this document, I refer to students with this view as programming–view students.

Professonal programmers were seen as the exemplars of Computer Science in programming–view; CS was not primarily an academic discipline. Unlike the theory–view students, programming–view students did not emphasize that some programming is not CS. Students in this group valued expertise with particular technologies and often were interested in problems that had concrete technical aspects (e.g. an e–commerce solution with a database). Theory–view and Programming–view students agreed, however, that algorithms were extremely important and the ability to design algorithms to solve difficult problems was an essential skill.

Programming–view students varied in how 'supplemental' the non–programming subfields of Computer Science appeared to be. On one extreme, the non–programming subfields of Computer Science are clearly fields in their own right with practitioners (e.g. in the analogy quoted at the beginning of the section, the "many people" who contribute without building a car). At the other extreme, there was definitely some *content* that Computer Scientists needed to know beyond programming, but someone who worked exclusively on the supplemental aspects was seen as on the edge of the discipline (e.g. someone who works on proofs about programs might be more of a mathematician than a Computer Scientist).

Programming–view students also varied on whether skills associated with the development process, but not programming specific, ought to be considered Computer Science. Communication skills and people management skills (especially on software projects) were sometimes included as part of CS. User Interface Design was similarly sometimes included and sometimes excluded. Often, technical expertise was an important consideration: students would ask how much a manager on a development project understood about databases, for example, in order to classify him as a Computer Scientist or not.

Some students in this group exhibited potentially problematic conceptions about CS. They readily agreed that non–programming aspects of their courses were useful to them, but they often had difficulty articulating good reasons why. Programming–view students argued that courses like discrete math were supposed to teach good mental habits or logical thinking skills. When asked about computer architecture courses, programming–view students could readily say that understanding the hardware could promote efficient programs, but had much greater difficulty thinking of an example why. They also frequently incorrectly anticipated the content of future courses.

4.1.3 Broad View: CS as Having Many Different Subfields

> "I know you can work - you can basically almost work anywhere. You can work for these corporate business, Microsoft, Google. You can work for the government, CIA, FBI. You can work as a computer analysis; you can work for the police department . . . You can build programs for them. You could work in their database and organize their files. You could - what else I used to do - you can analyze various things like for the CIA, FBI, the government, you do various things with them."

—P29

The third view of CS was as a very broad category that was interdisciplinary and included many distinctive (and equally important) subfields. Students with this viewpoint almost universally emphasized that Computer Science was much more than just programming and that a degree in Computer Science had a wide variety of applications. Elsewhere in this document, I refer to students with this view as broad–view students.

Broad–view students struggled to articulate a division between using Computer Science and simply using a computer. They often gave examples of how programming could be used in interesting ways. They also often wanted to make clear that programming was not the only possible thing Computer Science could provide but it was difficult for them to come up with concrete examples. Occasionally students would veer into potentially problematic conceptions by strongly emphasizing fields in CS (like logic) that were not part of their school's curriculum. However, it is important to note that no one in this view subscribed to the simple notion that everything that involves using computers involves Computer Science.

Broad–view students included researchers, professional programmers, and others in their view of the field. "Researchers" in this case were generally academics working on some specific application of Computer Science. Broad–view students often incorporated user interface design as part of CS. CS Theory tended to have a limited role and on occasion was ignored entirely. Topics like data structures and algorithms were considered important regardless of which area of CS one wished to pursue. Broad–view students were also likely to mention ethics and communication skills as things needed by all Computer Science majors.

Broad–view students often had goals outside of a traditional computer programmer role. They often described initially viewing Computer Science as about programming, but then discovering a wider view. Not all had negative experiences with programming, but that was common.

On occasion, broad–view students incorrectly identified the contents of future courses (e.g. saying that Operating Systems was about Linux distributions). This was more pronounced in areas they had less interest in, like operating systems and architecture courses. Even though they generally brought up a greater variety of subfields of computer science than other students (especially interdisciplinary ones), they generally did not have detailed knowledge about them.

4.1.4 Commonalities Between The Three Main Viewpoints

Although the three viewpoints are different, there are a few key ideas that are common to all conceptions. These ideas are worth highlighting because they are things an educator can probably assume most of his students agree with (at least in courses for sophomores or later):

1. *Programming is an important skill.* Students of all groups expected to do programming in their courses. Even when they personally did not enjoy it, or when they did not feel it was the "core" of Computer Science, they still considered it a major part of a CS education.

2. *Programming is not all of CS.* Even programming–view students acknowledged the importance of other skills. Although students sometimes misidentified the purpose of learning specific non–programming skills, all seemed convinced that other topics could be useful to them. This is not to say students agreed with everything they were taught: students complained about useless content in certain courses (more on this in the next chapter). All students were open to the idea of learning new non–programming ideas.

3. *Algorithms are essential to CS.* Students in every category mentioned the ideas of algorithms as essential to CS. For all groups, the idea of someone sitting down

and coming up with an algorithm to solve a challenging problem was maybe the "most" CS–like activity possible. There are variations: theory–oriented students would probably talk about understanding the mathematics to know it will work, and programming–oriented students might describe the CS person as "coding" the algorithm rather than designing it. All the groups agreed that algorithms were a major part of what a CS major learns.

4. *No detailed knowledge of subfields of CS.* Undergraduates tended to reason about CS in fairly broad strokes. Parts of CS that corresponded well with the students' outside knowledge might get mentioned (e.g. networking, databases, and robotics for example) but other less obvious areas would tend to get lumped together. For example, students would talk about the "low–level" parts of CS which seemed to contain (approximately) architecture, operating systems, and compliers (and sometimes building hardware). Even when students were pursuing a particular field in CS, they were just beginning to do research and did not yet understand the different subfields of a larger area like graphics.

4.1.5 Discrepancies Between the Three Main Viewpoints

The three viewpoints disagree in many small ways. Oftentimes it comes down to a matter of perspective: if you look at individuals in the workforce with CS degrees, it's easy to think of CS as the "the science of professional programming." On the other hand, if you look at the courses labeled CS in a average curriculum (especially at smaller schools), the "mathematical study of algorithms" seems more appropriate. But there are two areas worth looking at more closely:

1. *Theory.* Students were most different from each other on the issue of theory. Even among students near graduation, students ranged from describing CS as essentially all theory ("basically a field of applied mathematics") to theory being a small footnote about Big O notation and nothing else. This study did not

identify any clear causes for this large difference. But it's clear instructors with a particular view about theory are likely teaching at least a few students with radically different views on the topic.

2. *Solving Problems with Computers.* Almost any student would be willing to describe CS as a field about "solving problems with computers." But this statement covers up some key differences in the groups. Theory–view students, it is most important that the problem be algorithmically interesting (and indeed, they often used games or other idealized problems as examples). Programming–view students would often discuss problems that were straightforward from an algorithmic perspective but involved interesting technologies or limitations (e.g. mobile apps). Broad–view students tended to think more about HCI issues and reach beyond CS itself. Some problems meet all three criteria but many don't; it seems to be easy to reach a false consensus when each group interprets the idea of "problem" using their own lens.

4.1.6 Students Attempting to Combine the Views

Most of the students interviewed fit fairly unambiguously into one viewpoint or another. There are a few that lie on the border between one viewpoint and another. Here's a student who has reconciled The Theory View and the Programming View by contrasting Computer Science (theory) with a field of "software engineering". Note that by the words "software enginneering" the student appeared to be thinking about programming – not the academic subfield of CS called software engineering.

> "In my view at least, they do a lot of research in sort of — well, they spearhead a lot of those really cutting edge fields like [muffled] computing or sort of a security encryption, algorithms or like just algorithms in general maybe with different applications and things like that. But for me like computer science, like they're much more sort of into the research

> aspects and pushing the technologies on the theoretical fronts and sort of the experimental stages.
>
> Whereas I would call - I guess call myself a software engineer where I use these technologies and I like to learn about these systems, these new technologies being developed and learn how I can actually combine and build a system that can support a service and application."
>
> —P39

Insofar as the term "computer science" is being applied to the theory side, one might say this is a Theory View but the view seems more nuanced than that. Another student provided a view between the Theory View and the Broad View by explaining that CS has a different character at different schools, some of which are more theoretically oriented and some of which specialize in areas like programming and entrepreneurship.

When talking to juniors and seniors, one definitely gets the view that students' viewpoints about Computer Science are still evolving. This was borne out in the interviews with graduated students; they usually discussed their view as continuing to change past their graduation. The three viewpoints are attractive to students: they provide a single coherent explanation and that makes students want to subscribe to one or another. However even experts don't agree on a simple definition of Computer Science, (see Chapter 2) so any simple view is inevitably going to have some contradictions. As students become more sophisticated, it is reasonable to expect them to combine views and allow for differing opinions about CS. This process seems to be just starting for most undergraduates; for most of the students in this study, a single view provided a sufficient explanation of CS.

4.2 Potential Problems With the Three Main Conceptions

One of the goals of this research was to identify potentially problematic conceptions. Potentially problematic conceptions are conceptions in conflict with the school's curriculum. Conceptions were evaluated in two main ways: students were asked to reflect on what the important content of courses they had already taken were, and students were asked to predict what the content of courses they would take might be. In Chapter 1, I argued that students who don't understand their curriculum have the potential to make poor educational decisions.

Based on interviews, students' conceptions were accurate at a high level. No students considered CS to be about application use or as just IT work, for example. No students felt that the content of CS was overall useless to them, and that they needed to learn real skills independently. All the main conceptions are at least reasonable views of CS.

But student views of CS also had some potential problems. These problems cut across all the main conceptions, and occurred with students both early and late their undergraduate careers. The main problems in student conceptions were:

1. Students knew few specifics about the contents of future courses.
2. Students did not often understand the role of theory in CS.
3. Students over-focused on programming languages rather than CS concepts.
4. Students often misinterpreted course content based on misleading names.

The following sections will example these potentially problematic conceptions in detail.

4.2.1 Lack of Specifics About Future Courses

> "Like I was signing up for fall classes. Okay, do I want to take processer design or operating systems class? And, to be honest, that stuff looks

> very similar to me from my shoes, right. I don't know anything about either one, so how am I supposed to distinguish them?
>
> So is there anything I wish like I'd been told? Well, yeah. I wish people would say like - I mean it's sort of impossible to tell you about it until you're actually in it and doing it . . . they don't sit you down and say, okay, look at this screen of assembly code. That's what you're gonna do if you go into platforms. Or look at this screen of Python code. That's what you're gonna be doing if you're in artificial intelligence, right?"
>
> —P24

Almost anytime I asked students to speculate about the content of a future course, students would explain they really did not know much about what the course entailed. This was true of required courses, elective courses they were looking forward to taking, or even courses they had signed up for in the next semester. It might not be fair to call this a "potentially problematic conception" because students had such a explicit assumption that entering a course with no concrete expectations was normal. That does not always mean that students could not speculate correctly. When I asked students to speculate and predict the content of their future courses, some of them could do it with fair degree of accuracy. The main point is that students weren't familiar with the specifics of their future courses and didn't normally think about them. This is consistent with vague nature of main conceptions: students were not aware of the subfields covered in their later courses, and so those subfields did not form part of their descriptions of the field of CS.

This particular issue challenged some of assumptions which began this research. Initially, questioning focused on cataloging concrete potentially problematic conceptions in specific areas (e.g. architecture, compilers, etc.). When it became clear that undergraduates do not reason about CS that specifically, the focus of the interview process changed away from detailed questions about particular areas.

4.2.2 Role of Theory

The place of theory in CS was definitely an area of student contention. For some students, it was central part of Computer Science, while for others it seemed almost a minor detail for analyzing an algorithm's speed and a few other obscure details. Even among theory–view students, oftentimes there was a feeling that large parts of the theory they learned were not useful to their goals:

> "Deterministic state machines are useful, finite automata. But like context-free grammar and things like that, the professor thinks it's important and good to learn because by knowing the limitations of the language you're working with, you can know what you can do inside that language. So the same thing applies for our compilers or our programming languages and things like that. If you know what the limitations of your programming language are, then that can help you better understand what you can do with it, and I understand that. But in my opinion I can know how to use Java without understanding all of the rules and conditions that would apply behind the scenes to build the language."
>
> —P20

The quote above is not an example of a student with a potentially problematic conception; this student could concretely articulate his instructor's likely viewpoint. I have included this quote to illustrate that not every student who wondered about the usefulness of theory had a problematic conception. Oftentimes students with the most detailed conceptions of CS wondered most about the contents of their classes.

Although there were some students with well–reasoned concerns about the contents of their classes, there were also students whose understanding of theory could be potentially problematic. It was common for students to overemphasize the coding aspect of CS, especially when the theory was a mathematical idea they found difficult to learn:

> "'Cause when I was working on the project I didn't have any idea what I was doing for the calculus part until I took out one of my friend to just like tell me, "These are the formulas you need to do." And then once I knew all the formulas I could just code them - like it wasn't a problem to code them. It was just I didn't know any of the formulas 'cause I don't really enjoy Calc 3 [for CS Majors]. So I feel like it's - like it should be subdivided.
>
> Like math majors should be able to know all that stuff if they need to but CS majors - that's not their priority. We don't need to know the calculus part. Like we can, I guess, talk to other people that are specialized in that. Like our specialty is creating code."
>
> —P23

Students also often explained that the purpose of techniques like induction and other math were to teach them logical habits of mind, but that they had no direct relevance to CS.

There were also students who's conception of CS simply did not include any aspect of theory or mathematical components of CS, even when pressed. This was especially true of Spelman students. Here is an example of senior at Spelman answering a question about the mathematical areas of Computer Science (after asking about the theoretical areas of CS didn't get much response):

> "Programming. Programming I think is a lot of math, as well as when you're first starting off. When you're learning binary, when you're learning about memory and RAM, and that type of thing. I think that is definitely

> where the math comes in . . . If you're trying to calculate something. If you're trying to build a program that is going to give you the sign, or cosine or a tangent, or the sign or cosign of something anywhere, you would definitely have to know what that is."
>
> —P30

This definitely seems to be something that the student has forgotten or not did not understand, not simply an argument that theory was not useful. It's clear from the curriculum standards [10] that these topics were part of the student's curriculum.

Students difficulty with recognizing theory as an aspect of CS is interesting, because almost all students included the idea of algorithms as some part of their definition of CS. Algorithms can easily be seen as a very theoretical topic — involving Big O or correctness proofs, for example. But it definitely seems possible for students to view the idea of algorithms as central in CS, while leaving out or questioning the mathematical analysis of algorithms. For all students, the idea of taking some problem and devising a new algorithm to solve it was an important CS activity. But for many students, devising a solution to an algorithmic problem was simply a skill separate from formal mathematics.

To summarize, theory seemed to be a difficult area for students to incorporate into their conceptions of CS. Some students mentioned it but did not really understand its purpose, others omitted it entirely even when pressed. It seems to be a common source of potentially problematic conceptions in CS.

4.2.3 Languages Rather than Concepts

> "I'm not entirely sure [what is in the Information Internetworking Thread] right now. I know I'm really interested in SQL because I'm making my own website right now that's based around like a database in SQL. And it's using like AJAX and JavaScript to pull in information and all that

> stuff. So I'm really interested in that. I'm not really sure what else is like entailed in the thread."
>
> —P23

Another common potentially problematic conception was students overemphasizing learning new programming languages as part of their CS curriculum, rather than new concepts. Although there are plenty of (perhaps apocryphal) stories about students complaining they want courses in particular programming languages, in my interviews the reverse was more common. Students would take a course that had a new language and think of it as a course to teach that particular language. This occurred both when I asked students to speculate on the contents of their future courses as well as when I ask students to reflect on courses they had already taken, but it did occur less in reflection.

Languages are a very concrete thing for students, which I suspect provide an easy way to conceptualize a course. Students are excited about something to look forward to in courses that otherwise seemed abstract. This tendency of a students to use programming languages as a way to comceptualize also worked in a negative way: one course at Georgia Tech was taught in Smalltalk, a course students generally perceived as useless (i.e. not used by anyone for "real" purposes). This course was mentioned repeatedly as a source of anger, with Smalltalk's uselessness presented as the key failing. For good or ill, it definitely seems like students use the languages taught in particular courses to reason about them.

4.2.4 Misinterpreted Names

> "So [the media thread] could be anything in maybe the news, in broadcasting, certain media outlets like YouTube or Flash and basically design, I'd say. If you were designing a video game you might choose media as one of your threads if you wanted to design one."

—P14

The student above has a potentially problematic conception about the media thread at Georgia Tech. This thread covers computer graphics and computer audio. The thread does have some aspects of video game design (although it does not include GUI design, which the student asserts later). It does not have a great deal to do with broadcasting, YouTube or Flash. The student seems to be reasoning based on the name "media".

Students reasoning incorrectly based on a particular name was moderately common. While it may seem on the surface to be very concerning, part of it may be the artificially of the interview environment. Students are separated from the resources they would normally use to make decisions about classes and then are asked a highly specific question like "what do you suppose might be taught in your Operating Systems course?" Given those circumstances, students who don't know the course might well speculate that the course covers differences between Linux and Windows. It is difficult to know if the student would use that reasoning when deciding to take an Operating Systems course, given that the states are higher and more detailed information is just a few clicks away.

It is worth noting however, that the names a school uses in its curriculum is significant at least in students' causal reasoning. Naming an area of specialization "intelligence" may be more evocative than "machine learning, pattern matching, and search algorithms", and may encourage student interest. It also clearly presents the possibility that students will reason about it like it's a part of philosophy department.

4.2.5 Summary

In this section, we identified several common potentially problematic conceptions:

1. *Lack of Specifics About Future Courses.* Students generally did not know much about the content of future courses, both for courses they were required to take

and courses that were electives.

2. *Role of Theory.* Many students did not understand the purpose of CS theory, and a few left it out of their description of CS entirely.

3. *Languages Rather than Concepts.* Students often overemphasized the importance of learning programming languages in their courses, rather than focusing on larger more conceptual learning goals.

4. *Misinterpreted Names.* Students often drew conclusions of the content of their curriculum based on misinterpretations of the names of courses or specializations (e.g. 'media' is about broadcasting).

The main thing to take away from these potentially problematic conceptions was that many of the potentially problematic conceptions were caused by a simple lack of familiarity with the curriculum. Students generally had not formed detailed conceptions about CS in general, and their reasoning in interviews seemed quite tentative. Nowhere did we see students expressing conflict with their instructors and attempt to take their learning in a different direction (as observed by Nespor [60]). Sometimes the omissions in students views of CS were smaller (e.g. not having an idea what is covered in an Operating Systems course) and sometimes larger (e.g. missing theory entirely), but students did not seem to have detailed conflicting views of CS.

Combining the potentially problematic conceptions with the 3 main viewpoints in the first section, the overall picture of student conceptions is one that is accurate at a high level but with problems with regard to specifics of courses or subfields. In the next section, I discuss how students reflected on their own changing conceptions and look at how some of these potentially problematic conceptions evolved in at least a few students. In the next chapter, I discuss how students make educational decisions about things like classes and why even bright students don't generally have a detailed view of CS as a whole.

4.3 Change of Conception

This section discusses how students believed their views of CS had changed over time. Because this is based on student reflection rather than longitudinal interviews, it's important to treat this data with caution. That said, student reflections on how their views changed provides insight into what changes seemed significant to students in retrospect, even if it is not a completely accurate view of the entire process of conception change.

4.3.1 CS is Not Just Programming

> "I mean, since I was little I just saw, when I think of computer science I thought of my dad all the time and all he did, he was a coder, a developer. So I just like imagine him when someone says computer science. Oh dad, what does he do? Code, that's what he does. And I come to high school and even in high school all they taught us was Java. And Java is just coding. So we just sit in front of a computer coding. And then I come to college and then I think after coming to college my idea of what computer science actually changed . . . I think it changed pretty quickly. Like joining the different organizations I saw people always talking about different threads they're taking and CS 1100. . . "
>
> —P13

Many students talked about how initially they viewed CS as just programming but that view changed either in high school or early in college. Many students had computers science classes in high school that they described as basically programming. Even before students had taken a high–school CS class, they had somehow heard it was programming. Students described expecting CS to be learning additional programming constructs, or languages, or specific applications (e.g. how to build webpages) when they initially enrolled in CS. In addition to talking about non–programming subjects or careers with CS, student also emphasized that this programming–only view did not realize the importance of algorithms which they later came to appreciate. Many students remarked that CS seemed more interesting after it became clear it was not just programming (even students who were enjoyed programming).

Students often contrasted their current views with thinking of CS an earlier view of CS as "just programming", but only one student actually asserted that CS was just programming in the interview. This change seemed to begin quite early in the curriculum — just about the time students were introduced to data structures. Some views that I classified as programming–view often were very close to just–programming in that students could not think of examples of CS activities outside of programming. Students have heard that CS is not just programming, but it seems like their view of the non–programming aspects of CS evolve over time.

4.3.2 CS is Not Application Use

> "...people come into computing are a lot like people who like to look through a telescope and think they wanna go into astronomy, and the analogy that everybody uses. They don't realize what does computing mean. Many times they think that they're going to come in and learn how to use the applications. They don't realize what you can do with computing ..."
>
> —P7 (student advisor)

Both in my own work with high school students (see related work chapter) and in my interviews with student advisors, there seemed to be hints that students might initially come to computer science with conceptions that include application use or 3D–modeling for example. For the undergraduates I talked to, there did not seem to be conceptions of this sort, at least as a "main" view of CS. For example, a student might argue that subfield of computer graphics might include some art training while still having a programming–centric view of CS as a whole. Students did not recollect many conceptions of this sort, but several did say they really had very little understanding of CS before their first courses.

I suspect that the reasons I did not see conceptions of this sort was because even the freshmen in my study all had taken at least most of a course in Computer Science. Student advisors definitely agreed that application oriented conceptions exist, but it seems that a single course in CS is enough to revise this conception. Because introductory courses tended to focus on algorithms and coding, I suspect these students revise their conception to a programming–view or (as I did see in a few students) a breadth–view that still maintained a few incorrect assumptions about later curriculum. All that seems to be clear is that application–oriented conceptions and other very unusual views of CS did not seem to be represented in the students I interviewed.

4.3.3 CS Deeper Than Expected

> "I guess 1331's a really easy class, I guess, because it's an intro to objects, so they make it really easy. And I was like, 'Oh, I get this programming stuff.' And I got to 1332, and I was like, 'Whoa, I don't get this.' And so that was - I realized then that it was a little more complex. And I got it at the end of the class, but I was kind of more apprehensive about taking any more classes after that. And then after that I took 2110 and then I was even more apprehensive."
>
> —P19

Similar to discovering that CS was not just programming, many students remarked that a lot more went into CS than initially anticipated. This viewpoint change also occurred early in the curriculum — anywhere from the first intro course to computer architecture. This was seen a less positive change by students: students were often attracted to CS because it seemed easy and they performed better than their peers. Students experiencing this change often began to wonder if they had the made the right decision to major in CS.

The exact topic matter that was deeper than expected varied. Some students talked about designing code as being more challenging than they anticipated. Others were surprised about learning details of hardware. In all cases, it seems that the students conceptually found the new material interesting, but it was also more challenging than they first imagined.

4.3.4 Learning About Subfields of CS

> "Like before when I thought of robotics, it was kind of like two different classifications. You had either the robotics like industrial robots, which was just an arm doing some kind of task, moving something, or you had a humanoid robot, which was trying to walk around or do something or interact with the environment, but it was humanoid. And then through the things I saw in the class, I saw that there are very broad fields of robots . . . there's robots that hop on one leg, robots that hop on two legs. We saw robots shaped like snakes that wiggle around and can climb up poles . . . Just things like that I was like I had no idea we were even trying to do that much less that you could."
>
> —P20

This last change of conception was in many ways the most interesting because it seemed to represent an elaboration of the main conceptions identified above. Among all the three main types of conceptions, students generally had a very vague understanding with regard to particular subfields of CS. Students from a broad viewpoint might mention that you could build robots in Computer Science but (maybe beyond one example) they could not elaborate on robotics or any area in particular. But, for a few students, a recent change occurred that encouraged them to deeply look into an area of CS. As a result of this research, they had significantly greater detailed knowledge which significantly expanded their idea of what was possible in Computer Science. In most the cases I interviewed, these students were seniors who had just started looking into this new subfield. They did not have details on the connections between their field and other areas of CS. This suggests that there are potentially more elaborated viewpoints of CS for some students after graduation.

4.3.5 Summary

Student viewpoints about CS are clearly changing as they experience the undergraduate curriculum. All of the changes students noted suggest a greater respect for the complexity of CS, and a feeling that CS had more possibilities than before. The fact that students are experiencing significant changes in their views, even near graduation, suggests there are likely to be further changes to student conceptions of CS after they leave their undergraduate careers.

4.4 Effect of Curriculum on Student Conceptions

The other sections of this chapter have focused on student conceptions with CS: what they are, what their problems are, and how they change. This section focuses on a different question: does a school's curriculum affect student conceptions? In Study 1, I interviewed students at three schools. Although each school was an accredited Computer Science program with courses in data structures, algorithms and other traditional CS topics, each school's curriculum also differed in important ways.

1. *Georgia Tech.* One key difference between Georgia Tech and other schools was the Threads^{TM}program [37]. Students select two "threads" as part of their degree off a list of eight: Devices, Information Internetworks, Intelligence, Media, Modeling and Simulation, People, Platforms, and Theory. The courses in these

threads make up over half their required CS courses. As a result, students at Georgia Tech were required to specialize more and at an earlier stage in their degree program.

Georgia Tech's CS degree program was also the largest, allowing the greatest variety of specialized CS courses. Although there were some courses designed to introduce students to the various specialties of CS, generally students did not mention them as a impact on their conception of CS. Students interviewed often had one or more CS courses in high school.

2. *Duke.* Duke's program had fewer elective course offerings than Georgia Tech and a larger set of required courses. Duke's program also encouraged multiple degrees; many of the students I interviewed were combination CS/ECE majors. Duke had two introductory programming courses (one for engineers and one for non–engineers), both of which were fairly programming focused. Students interviewed often had one or more CS courses in high school.

3. *Spelman.* Spelman College is a traditionally African–American Woman's college. Unlike Duke and Geogia Tech, Spelman does not have a Computer Engineering program separate from CS. Spelman's introductory CS course was not programming focused, and students frequently remarked that it changed their view of Computer Science. Students interviewed usually had not taken CS course prior to coming to Spelman.

Overall, student conceptions were similar even at different schools. Each school contained students representing each of the three main conceptions, with the exception of Spelman which did not a have a theory–view student (at least among those interviewed). The curriculum at each school may have encouraged students toward one view or another: given the small sample size and variation between individual students, it's not possible to draw any conclusion about conception prevalence from this study. However, there were some noteworthy differences between students at the three schools.

4.4.0.1 Differences Between Duke and Georgia Tech

In general, Duke and Georgia Tech students seemed similar in their conceptions of CS. There did seem to be some affect of the Threads$^{\mathrm{TM}}$program: students at Georgia Tech had considered their threads somewhat and made tentative selections even at early stages. Duke students by contrast had no decisions to make earily and therefore did not know much about later courses. Neither group had detailed knowledge of particular subdisciplines of CS or the content of future courses. The pre–college background of both groups was similar as well.

4.4.0.2 Differences Between Spelman and Other Schools

Spelman students were different from Georgia Tech and Duke students in several key ways. Most obviously, Spelman students' background tended to be different. The majority of CS students interviewed at other colleges enrolled with some programming experience, Spelman students often selected CS without either high-school courses or personal experiences coding. Perhaps as a result of this, Spelman students often felt their initial Computer Science course was a large influence on their view of CS as a whole.

Spelman students often emphasized that an important part of CS was communication and group work skills:

> "I think biology I would have just been kind of studying and taking the tests, and that's all. But, I feel like for computer science I definitely — we have a lot of group projects, there's a lot of times where we have to do things together . . . But, a lot of the times we worked together and had to really figure out how to solve a specific problem right then and there. So, I think that my Computer Science courses helped me to really get that,

> whereas if I were in biology or any other subject, I don't think that I would have been able to get that type of experience, and be able to figure out how to quickly come up with a solution, or a solution for that matter, at all."
>
> —P30

Part the reason for this emphasis may be the introductory course. Spelman's introductory course emphasized that a variety of career options were possible from Computer Science including managerial roles. Another contributor was that more Spelman students were seeking non–programming careers after graduation (they did not usually have a concrete idea of what the wanted to do after graduation, but they had decided not to do programming). Students outside Spelman were much more ambivalent. A few students mentioned communication skills as useful, but almost no one mentioned them as a key component of CS knowledge. Other non–Spelman students contrasted technical programming knowledge (as CS) and project managment skills (explicitly non–CS).

Spelman students were also much more likely to include the building of computer hardware as part of CS. Outside of Spelman, students usually confidently enforced a strong delineation between CS as software related and Computer Engineering as hardware related. Inside Spelman, building computers was frequently mentioned as a CS activity. This may because Spelman did not have a Computer Engineering department.

4.5 Summary

This chapter has focused on answering the following research question:

RQ1: What types of CS field conceptions exist in CS undergraduate students?

Student conceptions of CS on the whole had several key things in common with each other. Student conceptions included programming as a part of CS, but also included non–programming. CS students acknowledged that algorithms were an essential part of CS, and that developing algorithms to solve particular problems was an important CS activity. In that sense, student conceptions were in line with the curriculums of the their schools.

Student conceptions fell into three main categories. Each of these categories had variations, but had specific characteristics that set them apart:

1. *Theory–view: CS as Mathematical Study of Algorithms.* Theory–view students thought of CS as a primarily theoretical (and academic) discipline. The design of conceptually difficult algorithms was most central to CS, as were other mathematical ideas like Big O and NP–Completeness. Theory–view students emphasized that CS existed even without real computers. Students varied in their association between CS and programming — from programming as clear part of CS (but slightly less central than theory) to programming as related but different (more engineering–oriented) field. Theory–view students tended to be accurate in their reflection on previous courses and anticipation of the content of their later courses.

2. *Programming–view: CS as Programming–Centric but Including Supporting Subfields.* Programming–view students considered CS to be mainly about programming, but emphasized that other subfields were also necessary to do good programming. Writing programs to solve large and technically challenging problems was the central activity. Professonal programmers were clearly Computer Scientists. Programming–view students varied on how important non–programming subfields of CS were: everything from small helpful fields to important subfields in their own right. Programming–view students occasionally

exhibited potentially problematic conceptions, especially with regard to understanding how non–programming subfields related to the activity of programming.

3. *Broad–view: CS as Having Many Different Subfields,* Broad–view students thought of CS as mix of many different computer–related subfields. Theory, Robotics, Programming, and (often many) others are all equally parts of a broad CS umbrella. There was no single central activity of Computer Science and no central practitioner. In this view, comparatively little knowledge was considered essential to a Computer Scientist; students emphasized the differences between subfields and the freedom to pursue different paths. Although broad–view students identified (and was excited by) interdisciplinary fields related to CS, they generally did not have detailed knowledge about them. Broad–view students had difficulty anticipating contents of their future courses, occasionally exhibiting potentially problematic conceptions (like OS class involves differences between Linux and Windows).

All three conceptions represent a potentially productive view of CS (at least insofar as correctly reasoning about the curriculum is concerned).

Students also exhibited a few potentially problematic conceptions. Students had difficulty understanding the purpose of theory. Students overfocused on learning new programming languages. Students made incorrect inferences about course contents based on the names of classes and specializations.

The most common potentially problematic conception, however, was simply that students did not have a detailed view of CS or their later CS courses. Even among areas of CS that they were interested in pursuing, students generally did not have concrete ideas of what their courses would contain.

Although this chapter has discussed many small details of student conceptions of

Computer Science, there are two main points to remember that are important for the next chapter on educational decisions:

1. Students conceptions on CS varied, but most students had a conception of CS that aligned well with their school's curriculum at a high level. Students expected to learn about both programming and non–programming topics. Students acknowledged the importance of algorithms in CS. Students generally viewed their instructors as in–line with their own goals about Computer Science.

2. Student viewpoints generally lacked specifics. They generally did not anticipate the contents of particular classes. In that sense, they were ill–equipped to make large scale decisions about what areas of CS to focus on.

The next chapter will focus on how students made educational decisions given this moderately accurate but vague understanding of Computer Science.

CHAPTER V

STUDENT EDUCATIONAL DECISIONS

In the previous chapter, I introduced three main conceptions of CS. Students of all conceptions acknowledged the importance of programming and non–programming topics, including the study of algorithms. In that sense, students were aligned with their curriculums at a very high level. However, student conceptions about the field of CS generally lacked specifics. When students were asked to anticipate the contents of future courses (even elective courses they had selected) it generally seemed the interview was the first time they had seriously considered what their future courses would teach (see Section 4.2.1).

How can students select elective courses without a detailed understanding of their alternatives? How do students make larger educational decisions like the choice to specialize in a particular area of CS? This chapter will present a theory about how students make educational decisions and the ways conceptions of the field of CS do (and do not) affect these decisions.

Our interviews suggest students make educational decisions in a way that initially seems arbitrary. Students in CS generally did not have specific career goals or skills they were hoping to learn in Computer Science. As a result, they did not worry about which courses or specializations would best help them achieve their goals. Instead, they were mostly concerned about finding an area of CS that they would be well–suited for. They measured how well–suited they were for a particular area by how enjoyable they found classes in that area.

Having an unenjoyable experience in a CS course generally motivated students to further elaborate their idea of Computer Science. For students, an unenjoyable experience indicated they were poorly suited for an area of CS and encouraged them to change their educational plans to avoid it in the future. This was also a time when they sought out advice from experts or parents. Once students had identified a concrete goal they felt they were well–suited for, understanding the field became important and useful. With a concrete goal, students became able to identify courses that were useful and not useful based on their conception of CS. But students who had concrete goals were much rarer in our interviews: most students were still selecting courses based on the experience of enjoyment.

Because the experience of enjoyment is so important in student decisions, it stands to reason that knowing detailed information about the content of future courses is not important. Students arrive in classes expecting that what they learn will be very different from their expectations; what they are most interested in is whether they personally will enjoy learning it.

In this chapter, I will elaborate more fully on the idea of student educational decisions based on enjoyment. I will talk about the educational implications of this, and how these implications reflect on the idea of student conceptions on the field of CS. Finally, I will discuss a few circumstances in which problematic conceptions about the field of CS did seem to influence student educational decisions.

5.1 A Theory of CS Student Educational Decisions

In this section, I propose a theory of CS student educational decision making based on my interviews. I begin with some of the puzzling student behaviors that suggests that students make educational decisions differently than one might expect. Then I describe my overall theory:

1. Students do not have a concrete goal when they begin studying in a particular field, and don't attempt to gain a detailed view of the field quickly. Instead, they take courses as prescribed by the curriculum. They make the assumption that

the curriculum is designed so that (regardless on what they might eventually pursue) it will put them in a good position. I described these students as *abdicating responsibility to the curriculum.* If students have to make educational decisions, they will generally focus on *exploration*, i.e. selecting courses based on casual interest.

2. As students explore classes, they *make educational decisions based on enjoyment.* They view their enjoyment of their classes as a useful measure of whether they would enjoy pursing a particular area more. If all their classes are equally enjoyable, they generally continue to trust in the curriculum and explore. But if they notice a strong difference (especially if they have a bad experience in a particular course), it motivates them make educational decisions. Often, they will narrow their educational focus and more clearly define their goals. This is when they often seek advice from parents, advisors, and websites. It also motivates refining a conception of the field.

3. Once their educational focus is sufficiently narrow, students develop a concrete goal. At that point, students' approach changes to *making educational decisions based on long term goals.* At this stage, they do use their conception of the field to make educational decisions towards their goal. This occurs late in the undergraduate career if at all.

The theory presented here are similar to the tentative theory published in my paper on student selection of specialization [44], with some further elaboration. In that work, I also stressed that students made choices without a detailed knowledge of the curriculum based on enjoyment. However, in that paper I did not have an idea of how choosing based on enjoyment could transition to more goal–based choices. I also did not see students early behavior as exploratory, and was more concerned that students were making ill–informed decisions.

5.1.1 No Concrete Educational Goals

> "It's hard to remember [why I took a CS class at first] . . . I thought I was kind of interested in, cognitive psychology and stuff and there's basically one — cognitive science actually. There's basically cognitive science course and it has as its prerequisites one of the following and the intro to Computer Science was one of them. So I kind of had it in my head like 'Oh, I'll take that and that's offered in the fall.' So I couldn't take that in the freshman fall cause I hadn't taken any of the prerequisites. And then I ended up taking, like, all of [the prerequisites] and never taking that other class."
>
> —P36

The first thing to know about students' decision making is that they do not have a concrete educational goal in Computer Science. As with the student in the quote above, a student's decision to take classes in CS might have nothing to do with a particular interest in the major. Even for students who select the CS major before they come to college, they may have enjoyed programming on their own but they almost never have researched the field of CS or what job they would like after graduation. Not having a goal makes the process of student educational decision making much different than you would expect.

For example, when talking with a student advisor, the advisor estimated that a third of incoming CS freshman have a very off–base view of what CS is about. Given that, one might expect to see a fair number of students initially major in CS and then quickly shift to another major that is more in–line with their goals. However, this does not seem to be a problem because students really don't have concrete expectations for what they intend to do with CS. According to the advisors, students don't change majors just because CS was radically different from their expectations.

Instead students start leaving when their GPAs begin to go down. The advisor estimated that only two percent of major changes are students who are doing well academically but find CS doesn't match their expectations. Even acknowledging that off–the–cuff statistics probably have a fair degree of inaccuracy, this suggests that many students enter CS with an inaccurate conception of what CS is, yet — once they change their view of CS is — most students (at least initially) persist in CS. Poor grades are what motivate students to leave CS, not innaccurate views of the field.

A second example of student decision making without concrete goals is how students making long term educational decisions like which courses to specialize in. Even when students generally have an accurate conception about CS, they often did not reason and research about CS when they were making educational decisions. Here is one student discussing how he selected his CS specializations:

> "So when I was choosing my threads, I thought it was like choosing my classes from the World of Warcraft or something, right? They really make it look like that. Like the little pictures of like intelligence, platforms, theory. So it's really like choosing - all right, I'm kind of feeling the dwarf warrior. So I picked my threads sort of — I mean I'm not too old, but as you can see from my background, when I got here I really didn't want to fool around, so I'm not changing my threads, right? I just went with platforms, and I went with information internetworks.
>
> And I've come to think that it won't make a difference. I think that the threads are superficial, but when I say that, I don't mean it too negatively . . . It's fun. But they don't really - in my opinion, they only sort of direct your thinking. They don't limit what you're gonna be doing after college, so I don't think they're really important, if that makes sense."
>
> —P24

The student in this quote had a sophisticated view of CS that encompassed both the theoretical and practical aspects of CS. Given that he was capable of reasoning about the content covered in the specializations, and given that the specialized courses will make up about half of his CS curriculum, one might anticipate considerable care in selecting specialization. Other students took similar approaches, almost never doing research or asking for outside advice when making selections.

When I explicitly asked students about their post graduation goals, they rarely had a specific job or category of job in mind. Except for students recently involved in a job search, students' goals usually were not committed enough to suggest specific educational paths. One student was deciding between continuing in CS to get a Masters or Ph.D., joining the Navy, or web programming. The student did not have a plan for how to purse any of these goals by taking courses in CS. Some students suggested they might want to become a professional programmer for a company like Google, although they could not give any specifics about what they would like to do in such a job or what Google might be looking for. Many students admitted they had no idea where they would like to work or what they would like to do.

The fact that students don't have concrete goals early in their CS education is not necessarily an educational problem. But it does raise a question: how do students make educational decisions without a goal? What made students select CS initially if not an idea of what they might do after graduation? Based on conversations with student advisors, students obviously cared about their grades but students did not talk about maximizing their grades or avoiding work. Instead, students of all sorts seemed very willing to just allow themselves to go with whatever the curriculum offered.

5.1.2 Abdicating Responsibility to the Curriculum

> "[I found my classes valuable not because] I had some predefined idea of 'this is what's important in this topic' and 'he should be teaching this'. It was because all of Georgia Tech's professors are very well-known . . . so when you go into a Georgia Tech class and you sit down in front of a professor, it doesn't matter what he wants to say. You kind of listen because you know it's gonna be important. It's just the people they are, that you trust them to know what they're teaching is important, and that's why we come to Tech."
>
> –P20

At all the schools we talked to, CS students had a great trust that the content they learned in their CS courses would be valuable to them. Even when they were not able to articulate why a particular topic was valuable, they were confident they learned it for some reason (or at least that it was useful to some particular kind of CS major even if it was not useful to them). In this, they were completely different from the management majors interviewed by Nespor [60] who colluded to undermine their teacher's lessons. Students were also confident that whatever they would be taught would be useful in accomplishing their career goals, even though in general they were not sure what those goals were. As a result, students generally selected courses by looking at the degree requirements and selecting the next courses off the list.

Not only did student expect that the courses they would take would be valuable, but they also in general expected them to be enjoyable. Students talked about arriving in courses not knowing what to expect until they were handed the syllabus on the first day. This was a positive experience — many of them were excited by how consistently different CS was from their expectations:

> "It's hard for me pinpoint [what my later classes will cover]. It's still for

> me still, I've had this feeling that by the time I get to a class it's something I never really would've thought about, but then made simple. I think with [Computer Architecture] was the idea I never really would've thought that machines and bit level work would ever be something I could do or would do. . . . And I feel like that's something I never would've expected until I get there, and that's kind of the experience I'm expecting with everything else."
>
> —P28

Abdicating responsibility to the curriculum has both a good and bad side. On the good side is students strong belief that the content they were learning was good for them, and would somehow be useful to them. Even when students were demoralized and had classes they did not enjoy, they still felt they would probably be good to know in the long–term:

> "I think [Computer Architecture] is important because everyone should know the foundation of what they're working on. Like it's important knowledge, I was just not interested. But, it's important to know, it's just one of those things where you're like, 'Really don't want to do this, but I guess I should know it, because I'm going to be working on computers the rest of my life.' "
>
> —P19

The bad side about abdicating responsibility to the curriculum is that students' knowledge of what is important begins and ends with the curriculum of their school. The student quoted above, for example, dropped out of her second semester Computer Architecture course. Because that was allowed within the framework of the curriculum, the student felt it was safe. This is not to say that the decision to leave the course was a bad one: simply that the decision about whether a particular course is valuable would ideally be about the CS content covered and a particular student's goals. The extent to which students believe the curriculum protects them against bad long term educational choices is perhaps more than their schools' curriculum designers intended.

5.1.2.1 Choices Don't Matter

Students' trust in the curriculum also manifested itself in a belief that all course choices that were possible within the curriculum would be of equal value. Here is a Georgia Tech student who is interested in robotics talking about if it matters whether he or she specializes in devices verses artificial intelligence:

> "Actually I really don't think so. I don't think, because when you graduate from Georgia Tech you just have like a computer science, you're a computer science major. It doesn't specify anything about [specializations] but I feel like once you take classes here you just focus on different aspects but you have some similar classes as well. . .While we're talking about it I'd rather take a devices [specialization] just because there's a lot more ECE classes and I think once I graduate the jobs, I think I would have the same number of jobs available if I took either of them. I don't think it'd make a difference."
>
> —P13

This idea that choosing a particular specialization does not matter was very commonly expressed by students. Students often asserted that the various specializations covered the same material but just "different aspects" of the same material. This was true even though the selection of specialization determined about half of a student's CS courses and Georgia Tech's department website emphasized the different possibilities enabled by each choice of specialization rather than the similarities.

Obviously, some amount of trust in the school's curriculum is valuable. However students seemed to take this trust to an extreme level that would allow them to avoid making educational decisions themselves. In the quote above, for example, it is probably true that there are jobs available for both students in AI and devices. But the type of work in the two specializations is different. Other students selected specializations because they perceived certain specializations as "traditional" CS. The students argued that traditional was likely to be good and useful even though they did not know what topics were contained in the traditional specialty they selected.

This section has focused on Georgia Tech students selecting specializations. Similar abdication of responsibility was evidenced in students at other schools. Georgia Tech evidences the difficulty particularly well because it requires students to make important choices early in their degree. At other schools, students were possibly even less informed about the contents of possible future choices, but at those schools students had fewer decisions to make in the near future.

5.1.2.2 Does Abdicating Responsibility Cause Educational Problems?

CS curricula are designed so that students who follow the curricula learn something valuable. In that sense, students blindly following the curriculum is safer than students rejecting the curricula. No matter which specializations Georgia Tech students select, they learn some Computer Science (even if what they learn is not really different aspects of the same thing, as they imagine).

The greatest risk here are problems of omission. One of the graduate students I interviewed talked about being very interested in video game programming as an undergraduate. Going to a small CS program, there were no courses in Computer Graphics. The school had a variety of opportunities for independent study, but the student did not realize that subfield of computer graphics existed. Near graduation when applying for video game programming jobs, the student was surprised by computer graphics questions.

5.1.3 Making Educational Decisions Based on Enjoyment

> "Well, it was with the same professor. And I don't even know what I liked about it so much but I do remember specifically telling him at point during the semester that he made me resent doing all my other work because I just wanted to be working on whatever stuff for his class. I don't even know, I can't explain what I like about it so much."
>
> —P36

The fun of programming was a part of almost every students' experience in Computer Science and for many of them was a strong influence on the choice of CS as a major. Students who had enjoyed AP Computer Science in high school often talked about how that experience motivated them to select CS as a major. Students who ended up in an introductory CS course on a whim or to fulfill a requirement chose to continue on because they enjoyed programming. Even students who eventually decided they didn't like programming acknowledged that the feeling of getting a program to work for the first time had a unique appeal.

Students liked different things about programming. Some found particular application areas cool: robots, media, or even technical projects like building assemblers. Many found the activity of programming itself appealing: being able to try a variety of different approaches, the feeling of accomplishment when your code works. A few students mentioned they enjoyed the feeling of competence programming gave them: performing better than peers and doing something that was acknowledged to be hard. A few other students mentioned that programming became no longer fun once they felt that they were having more trouble than their peers.

The idea that students might be motivated by enjoying their classes is not surprising — for example the Eccles model [30] presents "interest-enjoyment value" as one of the motivators of achievement related choices. What was surprising was the extent to which enjoyment of classes was mentioned in students' reasoning about their educational decisions.

5.1.3.1 Enjoyment as a Guide

> "So I decided, when I got into Georgia Tech . . . I needed to major in something, and I like computers. And if I continue liking computers while being required to do them, then it becomes more than just a hobby, and I think it's a pretty good indicator that's what I wanna do. So I took my first programming course with the intent to see whether I would hate it once I was actually required to do it, and I found — not a surprise, but — certainly it was a pleasant thing that I loved it. And I got a high A in the course; went far above and beyond on all the homeworks because I liked it; I - it was fun."
>
> —P27

Students used courses as a mechanism to test their own enjoyment of the field of CS. If they found a CS course enjoyable, that was generally construed as confirmation that CS was a good choice. Involved in the experience of enjoyment was other things: you can see in the quote above that the student mentions a high A. It might be reasonable to suppose that if the student had gotten a lower grade, the course might not be remembered with such fondness. But the important thing is that the thing the student uses to make the educational decision to persist in CS is the fun and enjoyment of the CS class. Similarly, when students had a negative experience in a class, it was the experience and not the grade itself that they talked about motivating their reasoning:

> Interviewer: So you said that [developing the next great algorithm or solving known hard problems] isn't your forte. How do you know it isn't

> your forte?
>
> P39: I guess I don't really know. But I mean like I've taken computer algorithms and through that like just working [on that course] and working through the problem sets and things like that . . . There's naturally like more motivation for me, just naturally to tackle that challenge rather than a completely theoretical problem with no grounding. It was just like, 'What if we did this, what happens? What is you apply this algorithm in this case?' You know, like knapsack problem or something like that. It just didn't have as much interest for me at least . . .
>
> Interviewer: So did you have like academic difficulty in [your algorithms course]?
>
> P39: Not really. It was like I did sort of average, above average usually on the test and problem sets. But it just, I didn't really enjoy the course . . . That's definitely my worst grade in Computer Science.

You can see in the quote above, the student needs to be pressed to reveal that his grade in the course was the lowest he's received in Computer Science. The students' main point is that he wasn't interested in the course and found some of the problems frustrating. This is not to say that low grades might not be partially responsible for the students' dislike of theory. The point I wish to make is that the frustration is what is reasoned about, poor grades can contribute to frustration, but frustration and enjoyment are what the students' describe as their reasons.

You can also see from the quote above how enjoyment of courses is interpreted as a measure of suitability. The student has had one course in theory then, based on that experience, characterizes himself as poorly suited for theory. For the student above, discovering he was poorly suited for theory is not a big deal and the student is planning to continue on in other areas of CS. For other students, the implication that they were unsuitable for CS was extremely stressful and prompted revaluation of the CS major. Students treat their experiences of enjoyment and frustration very seriously.

Students used enjoyment as a guide to their educational decisions, even when they could easily identify other factors that might have contributed. One student decided to major in CS, although he admitted it probably had more to do with an exceptional instructor than the content of the course itself. Another student became very ill partway through his AI course, missed many classes, got academically behind, and had to drop out. This student decided to stop specializing in AI even though he admitted that getting ill might be partly responsible for why his experience was bad. Students seemed slightly sheepish about their decisions when pressed, but although they knew that intervening factors could have affect their experience, they still decided based on their experience of enjoyment.

A course being enjoyable however, was not the same as it being easy. Students found easy courses unenjoyable sometimes and no student mentioned that a course was enjoyable because it was easy. Some students even enjoyed courses that took a large amount of time (hard, by some definition). That said, students generally did not enjoy courses that they had great academic difficulty. Assignments were a large part: assignments that were perceived as fun (even when difficult) could contribute to an enjoyable course, assignments that were frustrating to students made the course unenjoyable. Feeling that others found the course easier than you made the course unenjoyable, even if overall grades were average. Students often mentioned feeling like they were doing better than their peers in courses they found particularly enjoyable.

5.1.3.2 Exploration

"So basically what I figured I wanna be kind of like well rounded . . . Information Internetworking was the other one and I figured that that would come in

> handy like pretty much anywhere, you know? Because I have a feeling there's lots of jobs for that . . . I mean, media seemed like a lot more interesting like the classes you take and such rather than modeling. I mean, those are pretty interesting too but I just, I felt like this was more interesting."
>
> —P25

When students were enjoying most of their CS courses, they selected courses in what I called an "exploratory" way. They selected courses they were curious about, given descriptions on the school website. They occasionally considered what might be good for a job after graduation, but this was usually based on instinct rather than any concrete data or specific companies they were aiming at. They did not get advice from instructors or advisors. Only rarely did they consider course difficulty. Overwhelmingly, what was most important was that the course or specialization seem interesting.

This exploratory behavior can continue even until junior and senior year. Obviously by the senior year students need to start making decisions about careers post graduation, but students with an exploratory approach still did not have a specific goal. They had some areas they were considering going forward (either in graduate school or in industry) but it was still an interest rather than a specific commitment. Students with a strong specific commitment generally could describe an experience of contrasting enjoyment that triggered their focus.

5.1.3.3 Contrasting Enjoyment Triggers Educational Decisions

> "Well, I just wanna explore more aspects of where I could go and what I could do in the future, and so maybe having a more people-oriented major, more literature basically, which might involve the major computational media, so maybe I could explore that, but I just - I know that

> I'm interested in languages, and I've become more interested in history, so instead of just technology . . . I found [my computer architecture class] boring, and I didn't grasp it so quickly, so that generally discouraged me and what was good about that AP computer science class was that it was really slow and everyone was at your same level or below you."
>
> —P14

Enjoyment of specific classes was an important aspect of student reasoning, but it was not the only things students considered. They solicited advice from parents, professors, and looked on the departmental website. However, when students reflected on their own significant educational decisions, it was almost always an unenjoyable class experience that initially triggered the crisis and forced the student to make the decision. Occasionally, it was a course that was simply OK at the same time as courses that were much better. Either way, the strong constrast in enjoyment that made the student reconsider and being thinking about making a new educational decision.

Unenjoyable experiences caused a student to reevaluate their options. This was when they would reach out and begin to do research into the various options within CS. This often gave them a more detailed view of the subfields of CS than other students. Students would also make decisions about themselves in relationship with Computer Science. Students would decide they didn't like the hardware–level parts of CS, or that they didn't want to program professionally:

> "I think that — I know I don't want to program, so I'm going to try to stay away from that . . . Yeah, after my C++ course, I liked it and I still had to do it, of course. But, I just knew that I don't think I want to sit here up all night doing this. I think that I would much rather — actually, I took a course, too. It was a software engineering course. And, so that was the life cycle — a life cycle process, and project management. And, I really,

> really liked that. I was kind of able to see a task through, and I didn't have to be the sole one programming, or the sole one doing one thing. I was able to talk to people, gather information, gather requirements — I really liked that."
>
> —P30

Students who had an contrasting experience would often explain themselves in terms of being a particular kind of person (e.g. a social person, who doesn't like just programming all the time). Students before this would usually talk about being curious about different areas of CS but not saying they were unsuited for a particular area of CS.

Students would sometimes have an unenjoyable experience but nonetheless choose to persist. This was particularly true of Spelman students, many of whom struggled greatly in their introductory courses but were encouraged to continue on by their parents. Although they continued on, the experience definitely seemed to change their relationship with Computer Science. Spelman students were far more likely to describe programming as very challenging but interesting as opposed to students at other schools who generally described it as fun.

This overall process of educational decision making seemed to occur at two levels during a student's undergraduate career: the selection of a particular major, and the selection of a particular specialization with in the major. A student would have a experience that would commit them to CS, for example, and then begin engaging in exploratory behavior to find a specialization within the major. Not every student talked about both stages — and for many students the selection of CS as a major came from an experience in high school.

5.1.4 Making Educational Decisions Based on Long–Term Goals

> "Well, I got interested in robotics. I was enjoying the class. Things were going well . . . but I wasn't sure exactly what I should go to towards learning robotics on my own and in the classroom. So I went to my robotics professor and asked him for some direction, and one of the things I asked was simply what threads would you choose . . . And I suppose the difference between when I changed my threads from when I originally picked my threads was originally I was thinking from what I like and what I do what would be good threads. But then when I chose the threads I'm working with now, it was more where do I want to go and how do I get there that made me choose them."
>
> —P20

Up to this point, we've discussed students who are choosing based on enjoyment and adopted an exploratory strategy. A minority of students had a different approach to educational decisions: they made educational decisions based on a relatively specific long–term goal for themselves. Most of these students had a contrasting experience that focused them in a particular area and encouraged them towards a particular long term goal. For example, the student quoted above had a very good experience in a robotics course. He changed his threads and started strategically selecting courses to further a career in robotics — a change from his previous exploratory strategy. Not every contrasting experience would do this: for example, a student might have an experience that settled them on majoring in CS versus something else, but within CS classes the student would still adopt an exploratory approach.

Students approaching CS based on long term goals had much more use for reasoning about the field of Computer Science. They often had done research beyond their classes into what was necessary for their long–term goal. They would even take non–required classes that they anticipated disliking, because they believed they would be useful for their goal. This was very different from students adopting the exploratory approach, who would exclusively select classes based on what they imagined they would enjoy (within the framework of the curriculum).

Two students seemed to have this approach even from entering the CS program: both of them were interested in programming video games, and that interest persisted throughout the major. There were also students near graduation who still basically had a exploratory approach. Based on my interviews overall (not a representative sample), I hypothesize that for many students, this shift in perspective occurs near the end of their undergraduate career (depending on the time when they have a contrasting enjoyment experience).

5.1.5 Peers, Parents, Advisors, and Professors

> "It's an architecture class. And I got there and I was like, 'I don't understand any of this. I don't really like it.' And I was kind of hesitant at first when I talked to my — the advisor in the CS department . . . and she was like, 'Well, modeling and simulation would be a lot better, but you can still do what you want to do with this and just might have some extra like outside learning.' I was like, 'That's fine, because I don't want to take this class.' "
>
> —P19

A little should be said about the involvement of other people in student's process of making educational decisions. The first is that students were fairly independent: although many students did mention some others at some point in their process, generally they described most decisions as being self–made (perhaps with a little advice). The departmental website was by far the most commonly referenced resource.

But when students did solicit external advice, they tended to use each group in different ways:

- *Peers.* What was most surprising was how little peers tended to come up in student discussion of educational decisions. Although students definitely talk with each other, they generally do not talk about (or at least retain) information about the concepts discussed in later CS classes. They do talk about the difficulty of courses, although plenty of students I talked to did not even have information about that. Students did not evangelize particular specializations, and it was even rarer for students to talk about being attracted to the major by others. What peers did seem to provide was gossip about particular specializations (e.g. 'everyone knows' the theory specialization is really hard), which students did occasionally use in their decision making process.

- *Parents.* Parents were heavily involved in some students' decision making, especially when initially selecting a major. Parents generally seemed to encourage students to make educational decisions with an eye towards careers. Some students seemed to talk about consulting with their parents frequently, some mentioned it hardly at all.

- *Advisors.* Georgia Tech was the only school with explicit departmental advisors that students had to meet with every year. At Duke and Spelman, students were required to meet with CS professors yearly. Students did mention going to advisors when they experienced a contrasting experience. The advisors themselves mentioned that students mostly sought them out to ask about graduation requirements. No student mentioned an advisor that they regularly met with for advice.

- *Professors.* Some students had a professor they had developed a personal relationship with after enjoying a particular course. Students in a such a relationship frequently talked about getting advice about educational decisions. Most other students did not mention getting advice from professors, even when considering changing specializations or having a bad experience in a particular course.

5.2 Implications of the Theory

In the previous section, I outlined a theory of how CS majors make educational decisions based on interviews. To summarize:

1. Students *abdicate responsibility to the curriculum* and rely on it to teach them what is important without attempting to gain a broad view of the field. If students have to make educational decisions, they focus on *exploration* and select courses based on casual interest.

2. As students explore classes, they *make educational decisions based on enjoyment*. They view their enjoyment of their classes as a useful measure of whether they would enjoy pursing a particular area more. If all their classes are equally enjoyable, they generally continue to trust in the curriculum and explore. *Contrasting enjoyment triggers educational decisions*: Enjoying one course much more or less than others will make them reevaluate their current situation and focus their goals.

3. Once their educational focus is sufficiently narrow, students develop a concrete goal. At that point, students' approach changes to *making educational decisions based on long term goals*. At this stage, they do use their conception of the field to make educational decisions towards their goal. This occurs late in the undergraduate career if at all.

In this section, I will look at several implications of the theory. I will look at the relationship between this theory and other models of educational decisions. I will argue that the theory predicts students will rarely object to the contents of courses, except when they have unenjoyable experiences. Finally, I will argue that detailed conceptions of CS are not perceived as useful to student educational decision making.

5.2.1 Relationship with Existing Theories

5.2.1.1 Eccles Model of Choice

Eccles and colleagues [30] have done considerable work investigating the relationship between students choices (including college major choice) and a variety of social identities (including race and gender). In Eccles's model, student educational decisions are influenced by two main factors: subjective task value and expectation of success. Both factors have a relationship with the idea of "enjoyment" as described by students in interviews.

"Subjective task value" in Eccles's model refers to the benefit choosers believe they will receive by making a particular choice (e.g. specializing in a particular area of CS). It has three components:

- *Interest-enjoyment value.* Interest–enjoyment is how enjoyment of a particular activity would be classified in Eccles model. The fact that students mentioned it frequently is consistent with Eccles model.

- *Attainment value.* Attainment value is the perceived benefit placed on a particular choice by a valued social identity. For example, if one's gender is normally associated with 'helping others' then choosing a specialization involved with helping others reinforces a valued identity. I did see some students particularly mention that they were interested in helping others and a few others interested in being "social." Overall, though, students did seem to focus more on interest–enjoyment than attainment. It could be the educational choices this research study focused on might not be perceived as having different attainment values (e.g. a specialization in artificial intelligence was just a valuable as architecture), perhaps because students did not know many details about them.

- *Utility.* Utility is the usefulness of a particular choice, leaving aside issues of identity. Students did consider issues of getting a job and logistical concerns like needing to graduate on schedule. These seemed to be secondary most of the time, except in certain obvious cases (dual–major students tried to select courses they could count twice, etc.).

“Expectation of success” is the second aspect of Eccles model, basically measuring students’ estimation of their likelihood of success given a particular choice. This is an interesting area of difference between the Eccles model and my own work, because of how intermeshed expectation of success and interest/enjoyment value seemed to be in my interviews. Eccles model is an expectancy value model — decisions are driven approximately the value of a particular choice (subjective task value) and their estimated likelihood of getting that value (expectation of success).

In my work, student enjoyment seemed almost interchangeable with expectation of success. Students who had an unenjoyable time felt they were unsuitable for a particular subdiscipline (i.e. subjective task value changes expectation of success). The reverse also occurred in my interviews: feeling that one was doing well compared to peers for example, seemed to motivate increased enjoyment (i.e. expectation of success changes subjective task value). In interviews, students would routinely treat what they enjoyed doing and what they thought they were good at doing as pretty much the same thing. For example, consider this quote:

> “I can come up with a solution or maybe a few solutions but they are definitely not the optimal sort of things. And when you ask me to do your proofs, I can do proofs. It just — I don’t know, maybe it’s just like I

> haven't learned the correct pieces here and there but it I didn't really have a firm understanding of it and a lot of sorts of more complex algorithms like why they work, how they work, why the runs times that — it takes me a bit more time to understand those aspects. But in terms of actual application and coding, I'm personally fine.
>
> [omitted discussion of if theory classes require more effort/time]
>
> There's naturally like more motivation for me, just naturally to tackle that challenge rather than a completely theoretical problem with no grounding. It was just like, 'What if we did this, what happens? What is you apply this algorithm in this case?' You know, like knapsack problem or something like that. It just didn't have as much interest for me at least.
>
> And for interview questions like, 'Oh yeah, we have this tower defense game and you have all these sprites going around. What's the most efficient way to sort of scan your surroundings to find the different enemies and how to decide whether to fire or fire at the next person or like the next enemy or whatever?' That's sort of a lot more interesting and I can sort of go through iterations and I come up with algorithms."
>
> —P39

Note that in paragraph two, the student naturally transitions from saying he perhaps doesn't have the background to correctly understand theory (expectation of success) to describing problems he's motivated to solve (subjective task value). The final sentence has both meanings: the student is saying that grounded interview questions are more interesting to him, and that he feels capable of solving them compared to problems with a specific optimal solution.

Looking at the quote above, it's impossible to determine if the student is saying he is not motivated to learn theory, therefore he isn't good at it or vice versa. I doubt that the student has an explanation himself. From an educational decision perspective, it doesn't seem to matter: students focus on areas they like/are good at, and avoid areas they dislike/are bad at. In this document, I will continue to use 'enjoyment' to describe student experiences in courses, which is similar to students' own language. But enjoyment as students refer to it has a component of self–efficacy as well.

Overall, this work is complementary with Eccles. In the case of educational decisions that CS majors make, it seems that interest–enjoyment value is more critical than attainment value for example, which would be difficult to predict from the Eccles model alone. However, the Eccles model does suggest an aspect of student decision making (self–efficacy) that it is difficult to assess accurately in this study.

5.2.1.2 Nespor's Study of Student's in Two College Majors

Nespor [60] observed students in two different majors (physics and management) and compared and contrasted how each group of student developed disciplinary knowledge. Physics students worked together in groups to understand the material from class and solve the challenging problems from the textbooks. Management students collaborated to circumvent their school's curriculum and practice the skills they considered important on their own. Nespor's approach was different than this study — Nespor actually observed groups of students working, as well as interviewing them. However, even without observational data, there are differences between Computer Science students and the two groups Nespor interviewed.

Computer Science students were clearly different from the management students who developed their own management curriculum and subverted the curriculum of their instructors. Computer Science students generally trusted their instructors implicitly and assumed content was valuable even when the did not understand why.

But the CS majors were also different from the physics majors: based on conversations with students, there's little evidence for the close–knit study groups that Nespor observed. Moreover, in Nespor's observation these groups often independently discussed content areas and even brought in external resources beyond the curriculum for a broader view. The CS students I interviewed definitely did not mention external resources beyond the curriculum; although CS students may be collaborating, I don't see evidence that they are using this collaboration to gain a broader view of the field.

What is similar to Nespor is that CS did seem to have a major culture that at least partially seemed consistent across distant schools. Consistent with Nespor's ideas, a large part of that culture seemed involved with the particulars of the sort of problems students solved: largely programming problems. In CS students, the enjoyment of solving these problems (and the lack of enjoyment, for certain problems) motivated students as they made decisions about their long term goals with CS. Given that, Nespor's hypothesis that different disciplines generate different cultures based on the problems they consider important seems consistent with the results of this work.

5.2.2 Students Rarely Have Preconceptions But Can Lose Interest

> "So [combinatorics is] kind of like an extension of [discrete math]. I mean, kind of I guess. Some of the stuff I could definitely see how it relates to CS like graph theory and such but some of the other stuff it's just like 'Why are we learning this?' You know, like recurrence relations. I don't see how this would ever come in handy in CS like ever. Like not even just like maybe it'd be useful in like a thread. I don't see when it would ever come into play. So, I mean, just maybe if it did, maybe kind of just make the class more relevant be like oh this is used here. You know? This is why you're learning it."
>
> —P25

Everything I've discussed thus far suggests that students do not have concrete content expectations for their CS classes. Students conceptions of CS aren't detailed enough to include specific expectations for classes. Students don't have specific educational goals for particular classes (or for their CS degree as a whole). Students generally abdicate responsibility to the curriculum and rely on instructors to tell them what is important. Given all of that, how do we make sense of the quotes such as the one above that seems to be explicitly criticizing course content?

Students do complain about courses, but they do not often complain about the course not covering expected content. Although students initially don't have expectations, as the course continues they can decide that the course has problems. Students may dislike the educational approach or even complain that the material is not useful to them. 'Not useful' in this case is strange because students don't have concrete educational goals, but students did use the term.

More often, students would not have complaints about the content they were learning but would not understand why they were learning it. They trusted that the content would be useful to them, even if they couldn't understand why:

> P35: I would say theory is more just what were we just doing? The foundation, induction, loop invariance, it's pretty vague in my mind.
>
> Interviewer: Does that sort of thing seem useful to you?
>
> P35: No, not really. Just understanding yes, but I don't really see why, just coding, I don't see why I have to know this when I'm coding. I don't see that right now.
>
> Interviewer: Do you think that you will at some point sort of see a relationship between coding or is this something else?
>
> P35: Maybe later, towards graduation if I have to do something and I have to find a more efficient way or I have to change the way this program

is implemented, then I can see it. But now, I don't.

The key point here is that oftentimes instructors attempt to explain the purpose of content, but that purpose does not sink in. Students' lack of understanding the purpose of content can interfere with both learning and motivation [16]. Students seem are willing to attempt to learn content without understanding it, so if the purpose is important students should not be relied on to volunteer the fact they don't understand the purpose of what they are learning.

Similarly, professors should not expect most students to be able to request specific courses or complain when content is missing from a course. In my interviews, I often asked students what additional courses they would like to see: in general students wouldn't have ideas or they would suggest a course in a particular programming language. Similarly, student advisors said they students almost never requested new courses be added to the curriculum, but that they would complain if there was a course in the catalog that was no longer offered. Students at Georgia Tech who got to choose between specializations said they liked it, but students at other schools did not mention feeling that they had insufficient control in their curriculum.

5.2.3 Detailed Conceptions of CS Don't Help Make Educational Decisions

The way students approach educational decisions is related to their conceptions of the field of CS. Students use enjoyment to measure their suitability for a particular major or specialization. A detailed understanding of CS would not let a student know what her or she is really interested in: 'What part of CS is enjoyable to me personally?'

Once a student has a particular goal, a detailed conception of CS becomes useful. With a particular goal in mind, a student can reason about courses that would or would not be valuable — independent of the question of whether a particular course would be enjoyable. If the students I interviewed are representative (and they may not be) then students do not generally decide on a particular goal in CS until late in their undergraduate curriculum, after most major educational decisions have already been made.

This answers one of the research questions I posed: do potentially problematic CS conceptions effect student educational decisions? At least in the case of decisions like which courses and specializations to select, my research suggests the answer is no. Students have several conceptions of CS all of which are accurate at a very general level. For most of the students I interviewed, that is sufficient because enjoyment is the primary way students reason about their courses and a more detailed understanding of the field would not help make a decision.

Given that students use enjoyment as a measure of suitability, the strategy students adopt makes sense. Students take courses required by the curriculum. Students rely on the curriculum to ensure they are exposed to a variety of areas all of which are potentially valuable long–term. Where choices exist within the curriculum, students select what sounds interesting but it is not a problem if they are surprised.

5.2.4 Summary

This section has discussed implications of the theory of CS undergraduate educational decisions, based on my interviews. The main points to recall are:

1. The theory is consistent with existing research into educational decisions, such as Eccles [30] and Nespor [60]. Eccles work is particularly interesting, as it suggests that there is a complex relationship between self–efficacy, enjoyment, and student educational decisions.

2. The theory suggests that students generally do not have concrete content they intend to learn in particular classes but can lose interest in a course over time. Also, students are usually not able to identify missed content that would be useful to them or courses they would like to take.

3. Because students use enjoyment to measure suitability, a detailed conception of the field of CS is not useful in making educational decisions about CS. As a result, the general level understandings identified in the previous chapter are likely accurate for the reasoning students do.

5.3 Some Educational Problems

In the previous section, I suggested that, in general, students conceptions of the field of CS were sufficient to make reasonable educational decisions. However, in the interviews I did see a few key examples of problems caused by conceptions of the field of CS. We will review two of them in this section.

5.3.1 Realizing It Was Useful Later

> "Some of the [coding problems] I remember looking at and thinking 'How could this algorithm ever be useful?' ... But I'm definitely more open to the idea that, maybe not in the context they presented it, it wouldn't be useful, but I can definitely see how this kind of thing would be really, really important down the road. Especially like a tree traversals for example is one that I was like 'Why would you ever do this?' And now I'm seeing trees are everywhere so I use them all the time. So that would be an example of a big one."

Occasionally students would reflect that they now understood the purpose of certain topics, even though they seemed arbitrary at the time the first learned them. This was especially mentioned in context of courses the student had originally struggled with. Students would often describe the initial course as unenjoyable, and it would also even have sometimes caused students to reconsider their options in CS (or even other majors). The student often reflected that they wished they had understood the purpose initially: that learning the material without understanding how it could be used made them work less hard.

Viewed in the context of the theory of educational decisions, this problem has some interesting complications. Based on theory, it's unlikely that the student entered the course expecting to learn particular topics. So if the professor imagined the students understood that context, that could be part of the problem. However, most of the content students talked about (as in the example above) was introductory material that presumably most instructors would have attempted to motivate — but perhaps that motivation was not explained clearly. It is also possible that students, experiencing frustration with the course, decided the material they were having difficulty learning was 'useless' simply because of the bad experience.

In either case, given that students make educational decisions based on enjoyment, this definitely has the possibility to cause educational problems. Many students talked about useless material that convinced them to move out of a particular specialization. It is impossible to know how many of them would have later decided it was useful if they had persisted in that specialty. Presumably there are also students who found useless material motivated them to leave the CS major entirely — but they would not have been available to interview.

5.3.2 On the Edge of Computer Science

> "I would like [some CS courses that] just relate computer science and the things we can do with computer science with business, like how can we help this person's business be more effective, how can we implement this kind of technology into this business, how can we help the systems within the organization. It's just more of an interest to me."
>
> –P35

Most students did not have a concrete goal in Computer Science, but generally liked most of their Computer Science classes and enjoyed programming. They generally did not have good knowledge of future courses, but it seemed likely they would find some aspect of Computer Science they enjoyed. A few students, however, did not enjoy some aspects of CS but persisted. The student quoted above, for example, did not enjoy programming and seemed to be learning towards a job supporting technical infrastructure at a company. Similar to other students, this student had inaccurate views of later classes, including thinking that the Operating Systems course would be comparing and configuring OSes like Windows and Linux. In the case of this particular student, this is more concerning because their assumption about the Operating Systems course is reinforcing the idea that CS has many IT–oriented courses (which is partly true at Spelman) but there are also many more traditional CS programming courses than the student probably anticipates.

For students of this sort, who were not enjoying aspects of CS, their conception of CS seemed more potentially problematic. Their goals were not more specific than other CS majors, and their conceptions were not objectively worse. The combination of poor understanding of the future CS courses plus their lack of enjoyment of parts of CS which will make up a considerable part of later courses is potentially problematic. Students of this sort would probably benefit from a more detailed understanding of CS (or more advisement), but were relatively rare among those I interviewed.

5.4 Summary

This chapter has focused on answering the following research question:

> RQ2: Do potentially problematic CS conceptions affect student educational decisions?

To answer that question, this chapter presents a grounded theory on how the students I interviewed make educational decisions.

Students do not approach educational decisions in the way we might initially expect. Even when students found their classes to cover content very different from their expectations, that did not motivate them to switch classes or majors. On the other hand, receiving poor grades in classes did seem to provoke switching — even though (based on my interviews) students did not seem to be overly concerned with maximizing grades. In our interviews, even students with detailed understandings of the field of CS treated educational decisions like which area to specialize in very casually. Students did not seem to get much advice from advisors or professors. In short, students do not seem to be reasoning about the field of CS when making educational decisions.

The theory of how students make educational decisions comes from two basic ideas. One: students do not have a concrete idea of what career or skillset they would like to pursue in CS; they are trying to figure out their goals within the CS program. Two: the primary way students evaluate what their goals ought to be is by examining their enjoyment of classes. Enjoyment of particular classes is used as a test for how suitable that area of CS is for them.

This situation creates three main behaviors:

1. *Abdicating Responsibility to the Curriculum.* Students do not have a concrete goal when they begin studying in a particular field, but rather than attempting to gain a detailed view of the field for themselves, students rely on the curriculum to teach them. They also assume that the curriculum is built in such a way that anything which is possible within the requirements of the curriculum is viable in terms of a long–term career path or that all choices are basically teaching the same content in a different way. Either way, students educational decisions "don't matter" and students are safe to pursue any area that strikes their interest within the curriculum.

   Abdicating responsibility is not always a bad thing from an educational perspective. Students arrive in class with very few preconceptions about what they expect to learn and (at least initially) assuming the professor is an expert with their best interests in mind. It does put students at risk for ignoring valuable content that (maybe due to financial or logistical constraints) is not part of the curriculum at their school.

2. *Making educational decisions based on enjoyment.* They view their enjoyment of their classes as a useful measure of whether they would enjoy pursing a particular area more. Students are aware that other non–content factors (e.g. unhelpful TAs) can affect their enjoyment, but they rely on it anyway. Enjoyment in this case is not simply good grades but frustration or great academic difficultly did make courses unenjoyable. If all their classes are equally enjoyable, students select courses in an exploratory way. They choose courses of casual interest, while keeping in mind course requirements.

   If they notice a strong difference in how enjoyable some courses are (especially if they have a bad experience in a particular course), it motivates them to make educational decisions. Often, they will narrow their educational focus and more clearly define their goals. This is when they often seek advice from parents, advisors, and websites. It also motivates refining a conception of the field.

3. *Making educational decisions based on long term goals.* Once their educational focus is sufficiently narrow, students develop a concrete goal. At this stage, they do use their conception of the field to make educational decisions towards their goal, and often describe research activities to refine their conception. They also engage in behaviors that are very unlike students making educational decisions based on enjoyment, like taking non–required courses they expect to dislike because they will be useful.

This overall theory is consistent with descriptions of student behavior by Nespor [60] and Eccles [30]. CS students behaved different than both Nespor's physics and management majors, but consistently across several different schools. Eccles model of choice that students also heavily consider self–efficacy in their decision, which suggests that within students descriptions of ‘enjoying’ particular classes they also consider their likelihood of success.

5.4.1 Do potentially problematic CS conceptions affect student educational decisions?

For larger educational decisions such a selecting majors, courses, and specializations, the answer seems to be “no”. Students are most interested in the emotional experience of enjoyment in their courses and not the particular content covered. For that reason, a detailed conception of the field of CS is not useful to them. The conceptions identified in the previous chapter — usually accurate, but very general — are sufficient for the sort of reasoning students do about educational decisions.

One area where student conceptions do seem to cause some problems is within the content of specific courses. Students did mention that content that seemed useless in earlier courses later turned out to be valuable. It is possible that students considering content ‘useless’ had more to do with frustration and bad experiences, but either way it is an concern. This educational problem was common, but by no means universal.

There were a few students whose view of CS had potential to cause problems. Students who did not enjoy all aspects of their CS classes, but chose to persist occasionally reasoned about CS in a way that was concerning. These students were often interested in a part of CS which was near some other field, but envisioned more content of interest to them in their upcoming courses than the curriculum supported. For these students, a deeper understanding of the specifics of later courses or more careful advisement might be beneficial.

In the previous chapter we discussed student conceptions about CS. In this chapter, we examined how students used (and did not use) those conceptions to make educational decisions. The next chapter will attempt to quantify the prevalence of the three main conceptions in the student body.

CHAPTER VI

PREVALENCE OF CONCEPTIONS AMONG CS UNDERGRADUATES

In Chapter 3, we identified three main conceptions about the field of CS: the theory–view, the programming–view, and the broad–view. Although the different conceptions did not seem to cause students to make problematic educational decisions, each group of students does have different expectations about the focus of their CS classes. It is reasonable to ask how common each of the conceptions are and if factors like gender and CS classes might affect students' conception of the field.

In this chapter, I present the results of Study 2. In Study 2, I tested a preliminary survey instrument based on the results of Study 1. The survey included both open–ended sentence response questions and closed–form questions. The survey was tested and revised using a thinkaloud protocol with five students (see the final Survey instrument in the Appendix) and then was taken by 103 students in a CS course. I evaluated the surveys to determine student conceptions based on both the open and closed–form responses. I also developed an algorithm to attempt to determine student conceptions based on just the closed response questions.

In this chapter, I present the prevalence of conceptions in the class I surveyed. I also do some preliminary analysis of the relationship between student backgrounds and their conceptions of CS. Finally, I talk about the accuracy of determining student conceptions programmatically and some problems with the survey based on this first large–scale test.

Table 3: Results of Study 2. N=99

Conception	Broad: 27 Programming: 41 Theory: 8 Uncategorized: 23
Gender	Female: 36 Male: 67
Ethnicity	African–American: 6 Asian: 19 Hispanic: 3 White: 67 Other/Blank: 6
Major	Computational Media: 20 CS: 70 CS Minor: 7 Other/Blank: 2
Threads (CS Majors only)	Devices: 16 Media: 23 Modeling and Simulation: 4 Information Internetworks: 23 Intelligence: 24 People: 24 Systems and Architecture: 17 Theory: 6
Previous Classes	High School CS: 37 Freshman Intro Course: 47 Discrete Math: 65 Introductory Computer Architecture: 26 Operating Systems: 2 Computer Graphics: 2 Algorithm Analysis: 2

6.1 Results from Study 2

Table 3 summarizes the results of Study 2. The study participants were students in Georgia Tech's Objects and Design course. This course follows after introductory programming and can be taken either before or after Computer Architecture. This course is taken traditionally in the spring of the sophomore year; although this course was in the Fall, so students are likely to be either ahead or behind the overall schedule. This course is required for all CS Majors (regardless of specialization) and all Computational Media Majors. From a class of approximately 175 students, 103 elected to participate. Four surveys were not included in the results due to very incomplete responses.

From Table 3, it's clear all three of the major conceptions were recognized in students. The programming–view was the most common (41%) followed by the broad–view (27%) followed by the theory–view (8%). There were also 23 students whose conception could not be identified. All these results are based on my analysis of the student responses based on both the open–ended questions and the numerical questions.

The number of uncategorized conceptions is caused by several factors. Ten of the twenty–three uncategorized were students who ranked both theory and programming as highly CS–like but few other 'broad' activities were considered CS–like. Some of these students are students with one of the three conceptions which could not be accurately distinguished by the survey. For example, a student who ranks both theory–view and programming–view questions highly might believe programming more central, but because both are ranked quite high that might not be obvious from the responses. Based on one student I interviewed for a thinkaloud protocol, I also suspect there might be some students whose conception of CS is a mix of the theory–view and the programming–view. It is also possible that there are conceptions of CS that were not identified in Study 1. Although every attempt was made to sample a broad

Table 4: Chi–squared tests of the independence of various categories vs. conception. Note that the p–values do not account for the fact that these tests are post–hoc. N=92

Category tested vs. Conception Type	χ^2	p–value
Underrepresented Groups (Female or not White/Asian)	6.7709	0.0337
Gender	5.1275	0.077
Took Introductory Computer Architecture Course	3.9136	0.1413
In Media Thread	3.3869	0.1839
In Intelligence Thread	2.2564	0.3236
In People Thread	2.1994	0.333
Took CS Course in High School	1.8249	0.4015
Took Freshman Leap Course (general into to CS department)	1.5357	0.464
Took Discrete Math Course	0.6093	0.7374
In Information Internetworks Thread	0.3589	0.8357
CS majors vs. other majors	0.3323	0.8469

array of students, with an interview oriented technique only a small part of the overall population can be interviewed. Therefore, it is possible that alternate views of CS might exist in some students.

Another factor that has inflated the uncategorized percentage is students' very brief responses to the open ended questions. At most, students would respond in one sentence and occasionally would leave the questions entirely blank. As a result, when student responses were inconsistent, there was often little to go on. In the future, it may be necessary either to increase the number of closed–ended questions (so a view can be extracted even if a few responses are out of place) or provide compensation to encourage students to spend more time (and probably accept lower response rates due to the complication of compensation).

6.2 Influence on Conception Selection

Now that we have data about the conceptions of a large group of students, it is reasonable to wonder if student conceptions are related to other factors. After Study 1, I did not have any specific statistical hypothesizes I wanted to test. The goal of this section is to see if the data suggests an interesting avenues for future research.

Because the data collected here is categorical, the chi-squared test of independence is a likely candidate. Basically, the chi-squared test estimates the probability that two categorical variables are independent. For example, we might expect (as a null hypothesis) that a student's gender does not affect a student's conception of CS. If true, then it's reasonable to expect that the percent of male students with a particular conception would be the same as the population as whole. If the percent of male students with each conception differs from the population as a whole, the chi–squared test estimates the probability that this is due to random chance.

Looking at Table 4, you can see chi-squared test of independence applied to a variety of the data collected in the demographic survey, sorted by p–value. In each case, what is evaluated is if the category tested is independent of conception type. Students with uncategorized conceptions were removed for the purposes of these tests. The p–values given are for a single planned test (not a large number of tests looking for interesting correlations). Because these tests are post–hoc and there are a large number of them, it is not reasonable to make statistical claims based on these data.

As you can see from Table 4, there was not evidence of a relationship between student conception and most factors. Threads and courses with less than 20 students in the sample were not tested due to the unlikelihood of producing a meaningful chi-squared. I provide a more detailed look at the factors that suggest some dependence in the sections below.

6.2.1 Underrepresented Groups

Woman and non–Asian minorities have traditionally not been well represented in CS. Based on my interviews, underrepresented groups often had different viewpoints and (occasionally) negative experiences with parts of CS. Therefore it is reasonable to wonder if underrepresented groups as a whole have different conceptions of CS, so I included underrepresented groups as one of the backgrounds I tested. A student was

Table 5: Frequencies of conception divided by underrepresented group membership. Number in parenthesis are what the frequencies that would be expected if membership was independent of conceptions. Students were considered underrepresented if they were female or if they were not white or Asian.

	broad–view	programming–view	theory–view	*totals*
Underrepresented groups	10 (10.7)	20 (17.4)	0 (3.2)	30
White/Asian Males	17 (16.3)	21 (24.8)	8 (4.6)	46
totals	27	41	8	

Table 6: Frequencies of conception divided by gender and ethic group.

	broad–view	programming–view	theory–view	*totals*
Female	9	17	0	26
Male	18	24	8	60
African–American	0	4	0	4
Hispanic	2	0	0	0
Asian	7	7	0	14
White	17	28	8	53
Other	1	2	0	3

considered a member of an underrepresented group if they were female OR if their ethnicity was not white or Asian. The analysis suggests that underrepresented group membership may affect student conceptions (see Table 4). Table 5 shows student conception frequency divided by membership in underrepresented groups. Members of underrepresented groups were less likely than their counterparts to have a theory–view and more likely to have a programming–view. Although underrepresented groups being underrepresented in theory is consistent with my experience in interviews, I have no explanation for why this is true.

The distribution for gender is similar to Table 5. Ethnicity was not incorporated by itself (i.e. without gender) because there were too many empty cells which is a threat to the validity of a chi–squared analysis. The complete breakdown of frequencies by gender and ethic group can be seen in Table 6.

Table 7: Frequencies of conception divided by students who took an introductory computer architecture course. Number in parenthesis are what the frequencies that would be expected if membership was independent of conceptions.

	broad–view	programming–view	theory–view	*totals*
Took Computer Architecture	3 (6.4)	12 (9.7)	3 (1.9)	18
Others	24 (20.6)	29 (31.3)	5 (6.1)	58
totals	27	41	8	

6.2.2 Computer Architecture

With a p–value of .14, the effect of computer architecture is not significant even without accounting for post–hoc analysis (see Table 7). Still, it is worth noting because it is the largest effect observed for a course or specialization. Note that for courses, students were instructed to only select courses they had completed (rather than courses they were currently enrolled in). In this case, students who had taken the architecture course were less likely to adopt a broad view of CS. Although this is not conclusive, this may be an indication that certain courses have an effect on student conceptions.

6.3 Programmatically Determining Student Conceptions

The results in the previous sections are based on conception classifications that came from both open–ended and closed–form response questions. Given that closed–form questions make up a large part of the survey, I was interested in asking if it was possible to identify student conceptions programmatically. A closed–form survey that can be evaluated programmatically is much more useful for larger scale research (and for instructors simply curious about their students' conceptions).

For the programmatic evaluation, I focused on a few questions that (based on my interviews) distinguished the three conception types. For each of these questions, students were asked to "rank how much each of these people could be considered a 'Computer Scientist' and how much what they do could be considered 'Computer

Science' " (see the Appendix for the full instrument). Table 8 shows the questions related to each conception. The following sections discuss how each view was evaluated programmatically.

6.3.1 Evaluating Programming–View

Programming view was fairly straightforward. If a student ranked three of the four programming questions as Computer Science (4 or 5 on the survey), they were classified by the algorithm to have a programming view. The activities were selected to include activities of programming skill but little theoretical interest to help differentiate from the theory view. Programming–view students were most mixed on whether knowing obscure features of a programming language or writing reusable code were Computer Science: many of them ranked one or the other highly but not both.

6.3.2 Evaluating Theory–View

The best differentiator for the theory view was ranking "A researcher who writes a mathematical proof that one algorithm is more efficient than another" as strongly CS. Students varied greatly in their answers to this question: 26 ranked it as one or two (basically not CS) and 24 ranked it at five (a great example of CS). The other two questions were more poorly posed (see Table 8 for the theory–view questions). Students from other views often ranked them highly so there was less clear division. If a student ranked all three of the theory questions as a 4 or 5, they were considered to have the theory–view.

6.3.3 Evaluating Broad–View

Broad–view was the most difficult to classify. In interviews, students with a broad view agreed that programming and theory questions were CS, but also had a broader conception that incorporated things like managing software projects that theory–view and programming–view students considered not CS. The difficult aspect was

Table 8: Instrument questions matched to conception

Programming–View Questions

1. A programmer who works for Microsoft on the next version of Microsoft Powerpoint
2. A programmer who works for a bank and codes algorithms to predict insurance rates
3. A programmer who knows a lot of obscure features of the C++ programming language
4. A programmer who writes really easy to read reusable code

Broad–View questions

1. A graphic artist who makes 3D special effects for movies using existing 3D graphics programs and occasionally programming small scripts
2. A researcher who studies how the elderly use social networking apps like Facebook and Google+
3. A designer who makes a really easy to use user interface for a new app, but doesn't program it themselves
4. Someone who fixes broken computers (e.g. replaces hard drives, reinstalls operating systems)
5. A manager of a large software project that doesn't do coding themselves, but understands a lot of the technical details

Theory Questions

1. A researcher who devises new algorithms for encrypting data
2. A researcher who writes programs to analyze network traffic and detect new kinds of computer viruses
3. A researcher who writes a mathematical proof that one algorithm is more efficient than another

Table 9: Classification problems with the programmatic classifier. N=99

Correctly Matched	68
Skipped due to blank response	7
Incorrectly Classified as 'Uncategorized'	10
Incorrectly Classified as 'Broad–view'	4
Incorrectly Classified as 'Programming–view'	10
Incorrectly Classified as 'Theory–view'	0

that broad view students in interviews often had idiosyncratic views about which subfields were CS and which were not. A student might have a broad view and still not consider user interface design part of CS, for example. To test for broad–view students, I chose five activities that many broad view students would consider part of CS but that many other students would not (see Table 8). If a student rated at least three of these as fully CS (at 4 or 5), I considered them part of the broad view. If a students selected at least 2 of these and also ranked the theory–view and programming–view questions highly, the student was classified as broad–view.

6.3.4 Uncategorized

The classifier was designed to be fairly conservative in that if it did not have strong evidence for a particular conception it would not classify a student. Students who corresponded to both theory–view and programming–view but not broad were classified as Uncategorized, as were students who did not meet the qualifications for any view. Some students often left some of the questions in Table 8 blank – the classifier did not attempt to categorize these students.

6.3.5 Classification Accuracy

Of the 99 surveys collected, 68 had programmatic classifications that matched my classification with open–ended responses (69%). The classifier did not classify students with missing responses; if these are removed from the consideration then the accuracy rises to 74%. You can see the complete breakdown of errors in Table 9.

It is possible to compute inter-rater reliability between myself and the programmatic grader: Coken's $\kappa = 0.62$. However, this statistic is not useful because inter-rater reliability is a measure with two raters considered to be equally accurate. In this case, the program cannot be more accurate than my analysis (which can take into account anything the program might consider and more). Simply saying that the program was 74% accurate is probably the most reasonable way to describe the result.

Overall, based on these results, the current version of the survey and grader are probably accurate enough for an instructor curious about student conceptions to use. Given the opportunity to revise the survey instrument and fix some problematic questions, I suspect it might be possible to greatly improve the accuracy. Building a closed form survey of student conceptions of CS for research purposes seems to be possible, although the preliminary instrument used in this study is not yet accurate enough to be used in that way.

6.4 Summary

This chapter has focused on answering the following research question:

RQ3: What is the prevalence of different kinds of conceptions among the CS major population?

Study 2 looked at a particular group of CS students, and it's important to be careful on generalizing the population of one particular CS class to CS students as a whole. However several implications can be drawn:

- Students of all three major conceptions are represented, and definitely have different opinions of CS as can be seen by their responses on the survey instrument.

- For this particular class, the percentages of students with each conception type were: Broad–view 27%, Programming–View 41%, Theory–View 8%, Uncategorized 23%.

- Based on Chi-squared analysis of demographic and educational factors, students from underrepresented groups in CS may be drawn to different conceptions than white/Asian males.

This chapter also looked at the possibility of evaluating student conceptions of CS programmatically. Programmatic evaluation based on the surveys collected in in Study 2 achieved an accuracy of 74% (once surveys with missing responses were removed). Although this accuracy is probably not sufficient to be used on its own, it does suggest that with refinement a closed–form survey of conceptions of CS may be possible.

Study 2 suggests that students with different conceptions of CS are present within CS classes. Although some conceptions are more common than others, all three conception types have a sizable representation. Although student backgrounds may have an effect on conceptions, it clear that students from the same classes can nonetheless come to very different views about the field of CS. The next chapter summarizes the results presented thus far and discusses the educational implications of student conceptions of CS.

CHAPTER VII

CONCLUSION

This chapter will begin with a summary of the initial research questions and their answers. Then I will discuss how this work contributes to the CS Education community as whole, and discuss implications of my work to pedagogy and curriculum design. Finally, this work concludes with a discussion of possible future directions for this research.

7.1 Summary of Research Findings

RQ1: What types of CS field conceptions exist in CS undergraduate students?

H1. CS majors will exhibit a changing understanding of CS, initially potentially problematic but becoming more productive.

H2. Multiple productive conceptions will be found in graduating undergraduate students.

In Study 1 I identified three main conceptions in CS undergraduate students: a theory oriented conception, a programming oriented conception, and a broad conception. I also saw some evidence of programming–only conception that exists in students towards the beginning of the careers. There may be several other views that exist in students at the beginning of their CS curriculum but most of the students I talked to evidenced some variation of the three main conceptions. Hypothesis 2 seems to have been confirmed.

Although Study 1 was not a longitudinal study, there is evidence for student conception change. Students talked about their view of CS broadening, especially to encompass the idea that CS was not just programming. In addition, the theory–oriented conception requires some exposure to CS Theory that most students wouldn't be exposed to before college.

Although it may be reasonable to say student views of CS become more sophisticated as students progressed, it is not clear that student views are "initially potentially problematic". Students with all types of conceptions (even at later stages of the CS degree) have difficulty anticipating the content of their later courses. However, because students tend to abdicate responsibility to the curriculum, even a very vague view of CS is sufficient for the sort of reasoning students do about their classes.

RQ2: Do potentially problematic CS conceptions affect student educational decisions?

H3. Potentially problematic conceptions of CS will affect educational decisions.

Student conceptions of CS do not seem to have a strong effect on educational decisions, at least for most decisions about courses and curriculum. Most students do not have a long term goal (e.g. career goal) they are trying to achieve. Instead, students explore the curriculum using enjoyment of their courses to help guide them. If students enjoy most of their classes, then they continue to explore. If students have a strong contrasting experience in one of their courses (either a very positive or very negative experience) it triggers a reevaluation of their options and encourages them to commit to educational decisions. Throughout this process, students rely on curriculum to ensure the courses they take will not cause them problems after graduation.

For students who are exploring, a detailed view of the field of CS is not an important part of their reasoning process. The content of future courses isn't as important as how enjoyable these courses will be. For students who have decided on a particular educational goal, detailed content knowledge about CS is important and students do seem motivated to learn about the field of CS once they have an educational goal.

One aspect of student conceptions that is less clear from interviews is if student conceptions made it more difficult to learn CS content. Some students did mention thinking some CS content was useless in earlier courses but later discovering it was valuable. It is unclear if this has a significant affect on learning, however.

RQ3: What is the prevalence of different kinds of conceptions among the CS major population?

H4. Students conceptions will vary across the student population, with potentially problematic conceptions persisting after introductory coursework.

From Survey 1, I constructed a preliminary survey instrument. In Study 2, I analyzed the responses of 99 students in a sophomore–level CS course. The programming–view was the most common (41%) followed by the broad–view (27%) followed by the theory–view (8%). Twenty–three of the responses could not be assigned a conception. All the main conceptions were represented in the class I surveyed. Students conceptions did vary, but based on the results of Study 1, the survey did not attempt to identify problematic conceptions (because conceptions do not appear to be a large part of student educational decisions).

7.2 Contributions

I identify three main contributions from this work:

1. *A theory of student undergraduate student conceptions.* Prior to this work, there was no concerted effort to understand how undergraduate CS majors thought about the field. My theory provides a way for CS educators to think about students' expectations for their classes and field that has educational implications.

2. *A theory of student educational decisions.* CS undergraduates make decisions about their courses in unexpected ways. A theory about how students make educational decisions can help administrators and curriculum designers make CS curricula more in line with student behavior.

3. *A preliminary survey instrument to elicit student conceptions of CS.* Although some questions on the survey were not perfect, as it stands the survey can be used by educators curious about their students' conceptions of CS. It also represents an starting point for further research into student conceptions using more quantitative methods.

7.3 Educational Implications

7.3.1 Dealing with Vague Student Expectations for Classes

Students exhibited an accurate but high level view of Computer Science. They generally took an exploratory attitude toward class, and did not know specifics about the goals of the classes they selected. They usually did not have specific long–term educational goals about CS.

From an instructor's perspective, this confirms the initution that instructors must motivate the content they present. It seems logical to think that any student who registers for a particular elective must have some purpose in mind, but based on my interviews students generally take classes without any concrete goals for the class. Instead, students explore courses to see if the content is enjoyable. This provides freedom for the instructor to focus on what is important. However, it does mean that professor must persuade students the material is valuable (or maybe just interesting) because they have no particular reason for learning the content.

The three perspectives provide evidence that student expectations for CS are diverse. The good news is that the is little evidence for students with very problematic conceptions of CS (e.g. CS as application use, CS as learning a particular language, etc.). Students of all conceptions expect CS classes to have some non–programming and some programming topics. However for courses that emphasize a particular aspect of CS (e.g. theory courses, HCI courses), it may be a good idea to very clearly set expectations upfront. For example, students with a programming–view might like to know that a HCI course will focus entirely on eliciting user stories and doing mockups; programming–view students might not expect that in a course.

Because students tend to abdicate responsibility to the curriculum, student critiques that course content is 'useless' may be more about frustration than the usefulness of the content. In my interviews, students generally entered courses with few expectations about the content of their classes and trusted the material would be valuable. Students would criticize the content of their courses when they had a bad learning experience. Similarly, instructors occasionally complain that students have negative preconceptions about their course content. Students probably do not have good reasons to complain about the content of their course, but neither are instructors right that students have negative preconceptions.

7.3.2 Student Enjoyment

Enjoyment turned out to be a large component of student educational decision making. Students attributed enjoyment in a class to be a sign that they were well suited for a particular discipline. This was true even though students could often identify reasons for their enjoyment (or lack of enjoyment) that had more do to with pedagogical factors (e.g. frustration with TAs). Strong contrasts in enjoyment motivated students to make educational decisions and narrow their long–term options.

While it is not surprising that enjoying classes motivated students, what is surprising is the extent to which students conflated enjoying courses and being well suited for a particular discipline. From an educational perspective, this can be problematic because there are many factors that influence student enjoyment: difficulty in getting

TAs, courses with too much required content due to curricular issues, etc. But a bad experience in a particular course has a potential to be much farther reaching than instructors might expect. A bad experience in a course may convince a student they are poorly suited for a subdiscipline of Computer Science.

This suggests that unenjoyable classes, especially unenjoyable classes that are prerequisite for many others, can make students consider themselves unsuited for (and avoid) large areas of Computer Science. Courses of this sort often have many stakeholders which can encourage too much material in the curriculum. Obviously no teacher intends to make a course unenjoyable, but most curricula have a few courses that are considered especially frustrating. These frustrating courses may cause students to prematurely decide that they are not suited for certain areas of CS that they might otherwise enjoy.

Another aspect of the way student enjoyment can trigger educational decisions is the fact that students often make educational decisions quickly, and do not (at least in my interviews) contact an instructor. When students are dissatisfied with their courses they are also often concerned if they are well–suited for the field of CS. If our goal is to improve student educational decisions, simply encouraging them to talk to someone knowledgeable about the field at these stressful times could be beneficial. Most of the time students are simply exploring and don't require (or desire) explicit advice. Obviously, there are logistical difficulties with having CS experts available on short notice to give advice and social barriers than might make students reluctant to get advise from an instructor. But based on my interviews, when students are making educational decisions based on enjoyment of classes is when they could make the most use of detailed knowledge of the CS field.

7.3.3 Design of Curriculum to Accommodate Lack of Student Goals

In our interviews, students definitely liked curricula like the Threads program which gave them control over their classes. However, very few students used the freedom to select specific classes for specific goals. Instead, students tended to make educational decisions fairly arbitrarily, in line with their exploration of the major.

Based on my interviews, encouraging students to specialize early in their academic careers seems to be counterproductive. Before I undertook this project, I imagined that greater control of their curriculum might encourage students to develop a more detailed conception of CS in order to make good choices. This does not seem to have happened: students rarely talk about researching specializations or talking with their peers about CS content. When students are forced to make educational decisions prematurely, they choose without much consideration. If they have to specialize early, it reduces their ability to explore.

Students rely on the curriculum and assume that any really essential content will be taught to them regardless of their educational decisions. Students ignore the fact that a decision to specialize early inevitably comes at the cost of some other material. Early in the design of the survey instrument, I asked students to select elective courses for a student who wanted to 'keep their options open' in CS. Students found this question quite difficult, and it is difficult question for CS educators as well. My research suggests that many students actually really do want to keep their options open in CS, and I think the curriculum needs to provide guidance in that regard.

7.3.4 Summary

In this section, I have identified a few educational implications of my research:

- Instructors must motivate the content presented in their classes, especially for courses that certain conceptions consider peripheral to CS.
- Students who complain about content being useless may have more issues with

the difficulty or teaching style of the course than with the content.

- Courses that are frustrating are problematic because they may inadvertently convince students they are ill–suited for certain areas of CS.
- Students especially need guidance in CS when they're experiencing contrasting enjoyment, but current advisement structures don't seem to encourage students to talk with CS experts at these times.
- Curricula where students make educational decisions early in there careers may be problematic because students seem to make these decisions in a arbitrary way.

Student conceptions are diverse, but from an educational perspective I think the main thing to remember is that students (regardless of conception) have a vague view of the field and rely on instructors and curriculum. Eventually, some students do develop goals in CS and, at that point, students can act with more independence. However, for much of their undergraduate careers many students will not have goals and the instruction and curriculum should reflect this. Based on my work, I have submitted a letter to Georgia Tech about the potential limitations of their Threads program, it can be seen in Appendix C.

7.4 Future Work

7.4.1 Future Work on Enjoyment and Educational Decisions

The strong relationship between enjoyment of classes and student educational decisions was one of the surprise results of this research. There are a variety of open questions about enjoyment that could be looked at going forward.

7.4.1.1 Causes of Enjoyment

Although we know that student enjoyment of classes has an impact of student educational decisions, exactly what causes students to enjoy their classes still could be explored further. Self–effacacy and grades are obviously in a complex relationship with enjoyment. In terms of their decisions, students generally described their decisions in terms of enjoyment rather than terms of grades. That said, doing poorly on tests and assignments often formed parts of students' discussion of disliked classes. Beyond that, models such as Eccles [30] posit self–efficacy as key component of student decision making.

Although self–efficacy played some role in student decision making in my interviews, students did not simply enjoy easy courses which gave good grades. Students may be seeking a difficulty level to provide them with a reasonable but not overwhelming challenge [19]. Regardless, it seems that there is an interesting relationship between self efficacy, student enjoyment of courses, and educational decisions.

Beyond the issue of self–efficacy, several other factors seem to be involved. Interesting projects seemed to motivate some but also was a cause of stress and frustration for others. Students definitely talked about specific topics being intrinsically interesting, but it is not clear how that is different from less interesting content.

Based on my interviews, further pursing causes of enjoyment would have some methodology challenges. Students are unsurprisingly uncomfortable discussing doubts of their ability to succeed in their chosen major. One approach is to ground qualitative interviews in some concrete instrument (e.g. surveys for psychology literature, or other mixed techniques [35]).

7.4.1.2 Establishing a Quantitative Link Between Class Enjoyment and Educational Decisions

One of the concrete predictions that come out of my qualitative work is that enjoyment should predict student educational decisions, at least for students without concrete educational goals in CS. This is something that can be approached quantitatively by comparing student enjoyment ranking of their classes to other likely factors like grades, self–efficacy, etc. From a study design perspective, this is relatively straightforward. However it is important to test the predictions of theories developed with qualitative approaches when possible. If the results confirm the theory, it increases confidence on the theory at a large scale. If not, it suggests that more work needs to be done to understand the process completely.

If a quantitative link between student enjoyment and conceptions can be established, it should be possible to make curricular changes and see testable results. By changing introductory courses that (perhaps due to excessive content or other issues) are pushing students away from parts of CS, it may be possible to encourage students to explore a greater breadth of CS.

7.4.2 Future Work on Student Conceptions of CS

At the beginning of this work, I hypothesized that potentially problematic student conceptions might influence student educational decisions. Now that I know more about the ways students make educational decisions, I do not believe problematic conceptions of CS are a large educational problems. This removes some of the reasons to study conceptions further. However, there are a few reasons to study student conceptions beyond educational decisions.

7.4.2.1 Expansion of the Survey Instrument

In Study 2, I piloted a survey instrument. The result is not perfect, but overall the instrument was successful in eliciting conceptions. There are a few questions that could be answered by surveying a larger population of students:

1. *Determining how the distribution CS conceptions change over time.* It seems likely the certain conceptions (e.g. theory–view) might become more prevalent as students take more CS classes. Surveying students at the beginning and end of their undergraduate experience could determine if the distribution of student conceptions change over time.

2. *Determining if individual courses change student conceptions of CS.* By testing student conceptions before and after I could confirm more accurately if certain courses can change conceptions (as was suggested by the computer architecture class in Study 2). Although none of the main conceptions are problematic, if courses change conceptions in unanticipated ways (e.g. a computer architecture course moves students away from the Broad–view) that may be concerning.

3. *Determining if a relationship exists between student conceptions and grades.* It is a open question whether student conceptions might affect content learning. Student conceptions could be elicited, and then content clearly related and unrelated to their conceptions could be provided. If students find learning content related to their conceptions easier, that is evidence that student conceptions of CS could be influencing the ease with which material is learned.

7.4.2.2 Determining Effect of Breadth–Oriented Curricula

Some CS educators recommend a breadth–first introductory approach to teaching CS as a way of encouraging students view CS as more than programming [25]. Based on my interviews, many students do have a very programming–oriented view of CS. It would be very interesting to see if explicitly attempting to teach the breadth of the field changes students' conceptions. Explicit instruction might change the distribution of conception in students or it might even cause qualitatively different conceptions to develop.

It also might be interesting to examine breath–oriented curricula from the perspective of my theory of student conceptions of CS. Similar to alternative conceptions research, it's reasonable to imagine that students holding a particular conception of CS might be resistant to education. By understanding what existing conceptions students hold, it might be possible to make greater strides in changing students conceptions about CS with instruction.

7.5 Summary

The goal of this research was to understand how undergraduate students think about the field of CS. I hypothesized that students might have problematic views of CS that would lead them to make bad educational decisions. After a grounded–theory based study of 37 students and counselors, I came to a few conclusions:

- Most students fall into one of three main conceptions: the theory–view, the programming–view, and the broad–view.
- All the three main views are reasonably accurate characterizations of CS at high–level, although they lack detailed knowledge of specific subfields of CS.
- Students don't have concrete educational goals in CS. Students make educational decisions in an exploratory manner, which does not require them to have detailed knowledge about the field of CS.
- Contrasting enjoyment causes students to make educational decisions, and encourages them to narrow their focus and develop goals.

I also did a survey–based study to attempt to determine the prevalence of various student conceptions in one class.

My research has several educational implications. First, instructors need to be aware that students need help understanding the reason particular topics are important. Student views of CS are diverse and students often select courses without understanding their content. Second, instructors need to monitor student enjoyment of their classes. If students have bad experiences, they're motivated to make educational decisions that prematurely ignore areas of CS because of a negative course experience. Third, designers of curricula need to take into account that students do not usually have concrete goals early in their undergraduate program. Forcing students to make educational decisions early results in arbitrary decisions.

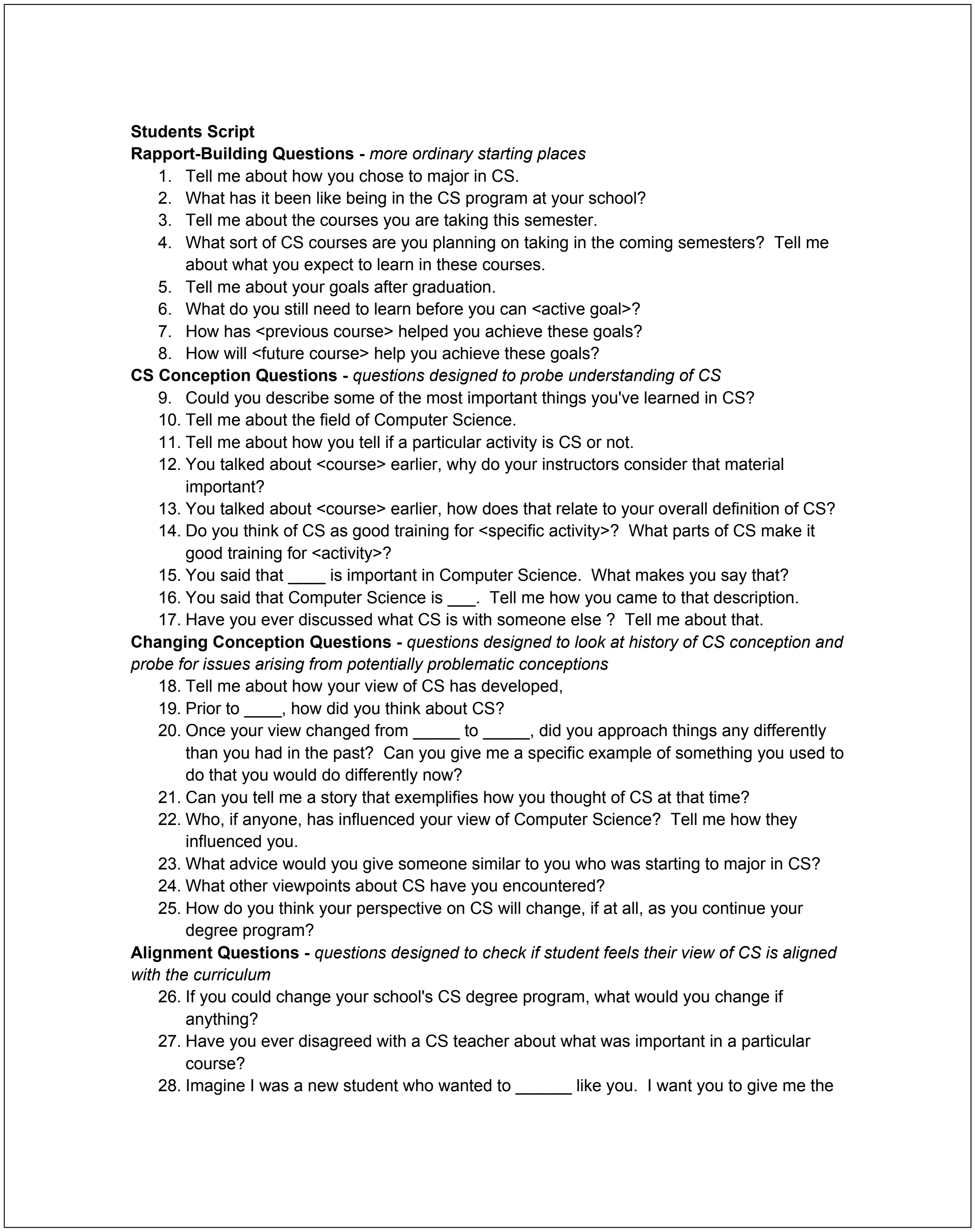
There are two main directions for future work based on this research. One direction is to further explore the relationship between student educational decisions and enjoyment of classes. The relationship between self–efficacy (not really explored in this study), grades, and student class enjoyment is likely to be complex. The second direction is to continue to expand examination of student conceptions. The instrument developed in Study 2 could be revised and used to answer several questions that remain open about student conceptions. The affect of breadth–first CS curricula on student conceptions could be studied. Overall, the area of CS conceptions and educational decisions has many interesting future directions.

Over the course of this research, my view of students' conception of CS has changed. When I began, I was concerned that incorrect knowledge about future classes might cause students to make poor educational decisions. Students conceptions of CS proved to be in some ways more sophisticated than I anticipated, and in some ways less. Student educational decisions initially seemed completely arbitrary but, thanks to the opportunity to interview many students, I now have a much better understanding for the reasons they do what they do. Overall, student conceptions of CS has proved to be a fruitful area of study thus far, with many interesting directions to pursue in the future.

APPENDIX A

STUDY 1 MATERIALS

A.1 Initial Interview Guide



Students Script

Rapport-Building Questions - *more ordinary starting places*

1. Tell me about how you chose to major in CS.
2. What has it been like being in the CS program at your school?
3. Tell me about the courses you are taking this semester.
4. What sort of CS courses are you planning on taking in the coming semesters? Tell me about what you expect to learn in these courses.
5. Tell me about your goals after graduation.
6. What do you still need to learn before you can <active goal>?
7. How has <previous course> helped you achieve these goals?
8. How will <future course> help you achieve these goals?

CS Conception Questions - *questions designed to probe understanding of CS*

9. Could you describe some of the most important things you've learned in CS?
10. Tell me about the field of Computer Science.
11. Tell me about how you tell if a particular activity is CS or not.
12. You talked about <course> earlier, why do your instructors consider that material important?
13. You talked about <course> earlier, how does that relate to your overall definition of CS?
14. Do you think of CS as good training for <specific activity>? What parts of CS make it good training for <activity>?
15. You said that ____ is important in Computer Science. What makes you say that?
16. You said that Computer Science is ___. Tell me how you came to that description.
17. Have you ever discussed what CS is with someone else ? Tell me about that.

Changing Conception Questions - *questions designed to look at history of CS conception and probe for issues arising from potentially problematic conceptions*

18. Tell me about how your view of CS has developed,
19. Prior to ____, how did you think about CS?
20. Once your view changed from _____ to _____, did you approach things any differently than you had in the past? Can you give me a specific example of something you used to do that you would do differently now?
21. Can you tell me a story that exemplifies how you thought of CS at that time?
22. Who, if anyone, has influenced your view of Computer Science? Tell me how they influenced you.
23. What advice would you give someone similar to you who was starting to major in CS?
24. What other viewpoints about CS have you encountered?
25. How do you think your perspective on CS will change, if at all, as you continue your degree program?

Alignment Questions - *questions designed to check if student feels their view of CS is aligned with the curriculum*

26. If you could change your school's CS degree program, what would you change if anything?
27. Have you ever disagreed with a CS teacher about what was important in a particular course?
28. Imagine I was a new student who wanted to ______ like you. I want you to give me the

real dirt. What should I do? What should I watch out for?

Closing Questions

29. Tell me about how your views on other things have changed since you started studying Computer Science?
30. On the whole, how would you characterize your experiences as a CS major here at <school>?
31. After having these experiences, what advice would you give to someone just starting in CS at <school>?

Is there anything you'd like to ask me?

Teacher/Counselor Script

Rapport-Building Questions

1. What events led up to you becoming a student advisor?
2. What your sorts of concerns do students bring to you most often?
3. When a student comes to you with a concern, what do you think about before you advise them?
4. Do you frequently advise students, who in your opinion, are making poor educational decisions? What, in your opinion, causes students to make these kinds of poor educational decisions?

Conception Questions

5. Tell me how you think about the field of Computer Science.
6. You talked about <course> earlier, how does that relate to your overall definition of CS?
7. You said that ____ is important in Computer Science. What makes you say that?
8. Compare your views with the views of students you talk to most frequently. Do you think that students have a different view of CS than you do?
9. Is that the only view of CS you encounter in students?
10. Can you give me an example of an interaction with a student you had where this was evident?

Changing Conception Questions

11. Tell me about how you think students viewpoints of CS change over time?
12. When students have <conception>, does that affect their educational decisions?
13. Can you think of a specific example of a student with <conception> and how that affected their decisions?
14. When you advise students, do you find it difficult to change their views?
15. What do you think causes students to have <conception>?

Alignment Questions

16. Do you find students find the curriculum does not include things that they feel are important?
17. What kinds of students have the most trouble fitting what they want to do into the way the curriculum is set up?

Closing Questions

18. Tell me about how your views have changed since you began advising students?
19. What advice would you like to give to all CS students?
20. What advise would you give to someone who is beginning to advise students?

Is there anything you would like to ask me?

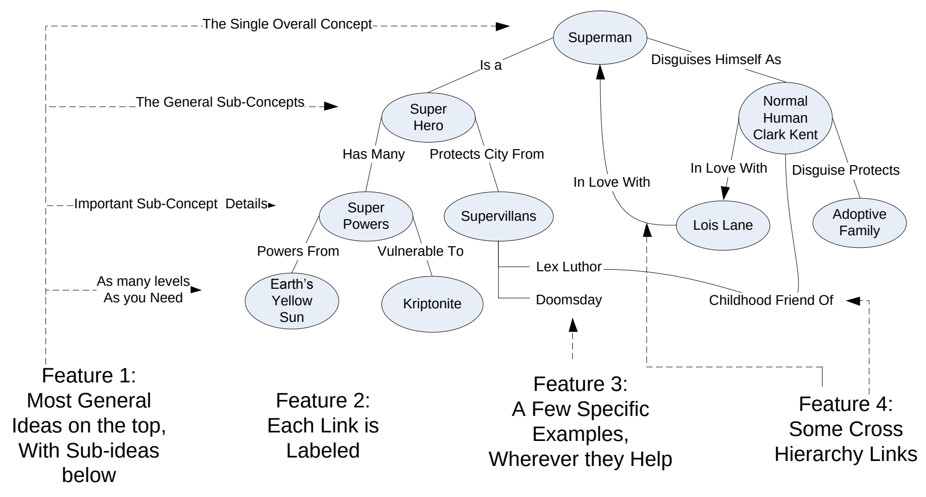
A.2 Initial Survey Document

Concept Maps

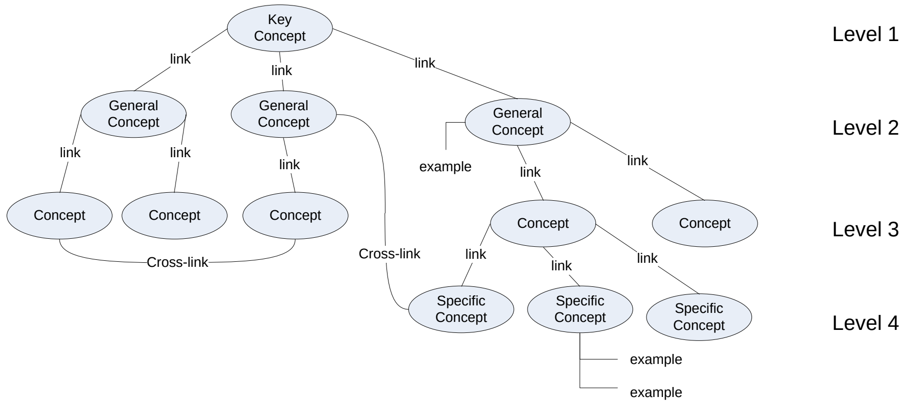
This survey asks you to build a Concept Map. A concept map is a hierarchical diagram representing a single overall concept, and all the important sub concepts that are part of it. A concept map has 4 key features, summarized in the example below. Before you begin, please look over the examples below and make sure you understand what a concept map is intended to do.

Please look on the next page for a more complex example.

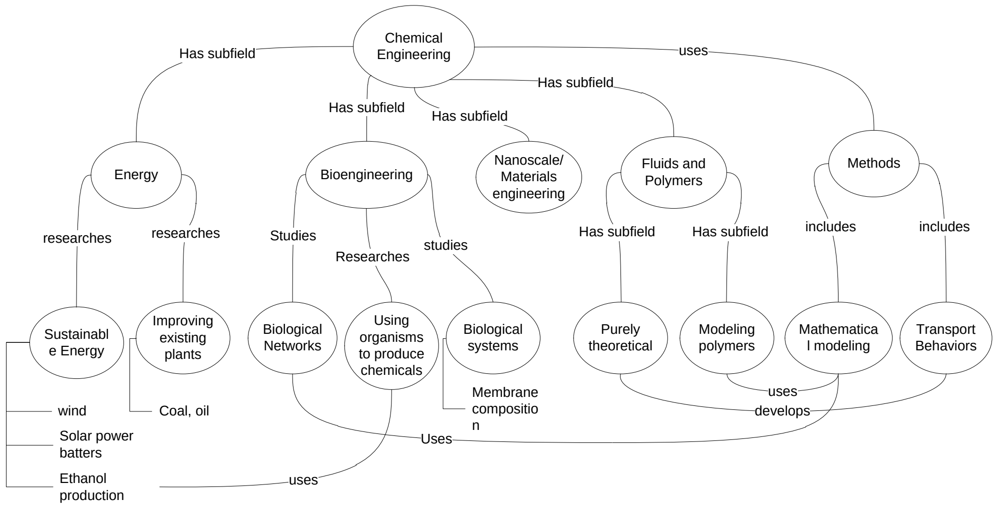
Example Map: Superman



Example "Generic" Map

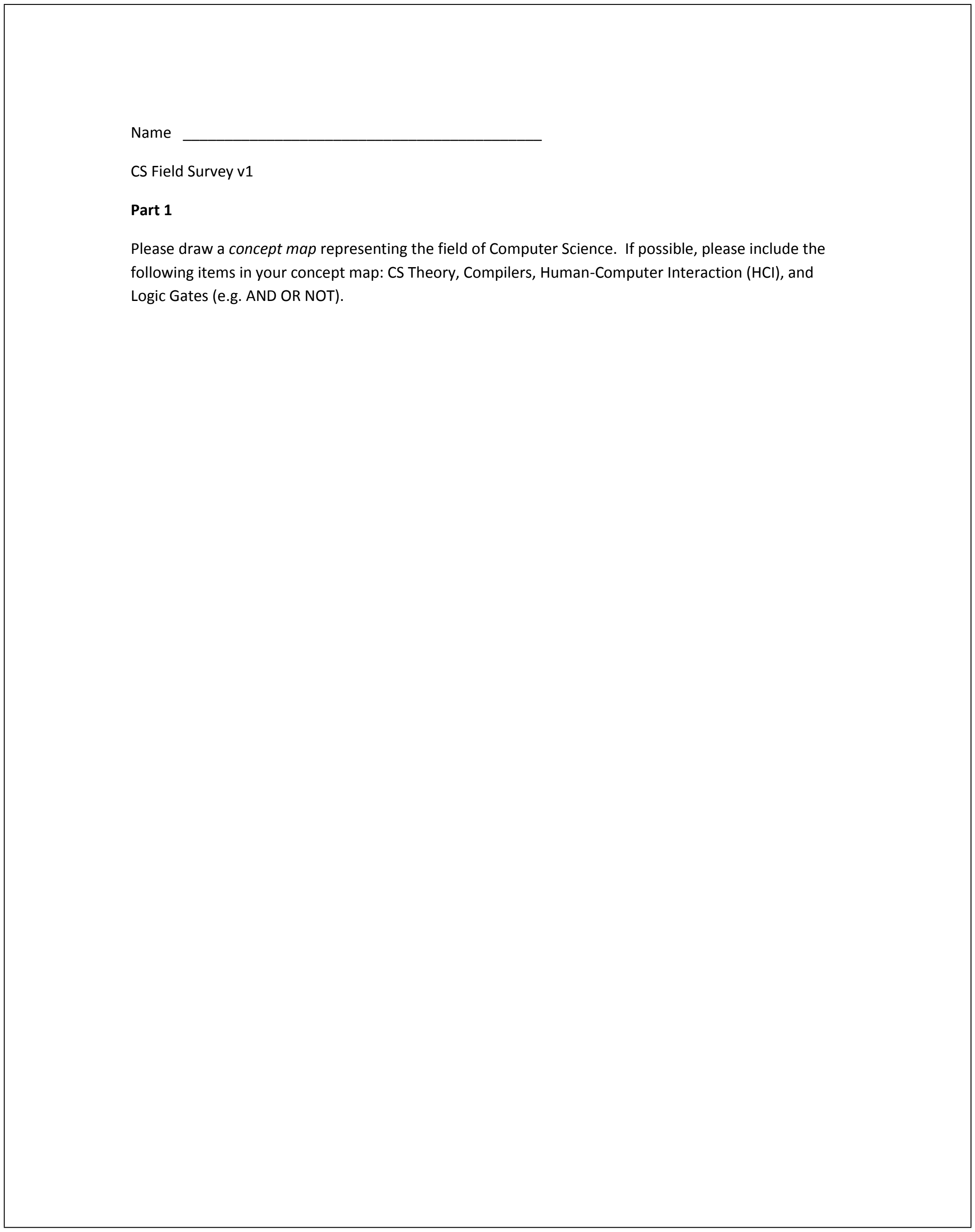


Below you can see an example concept map of the field of Chemical Engineering, as designed by a Chemical Engineering student. Note that there may be many possible correct Concept Maps for a single idea.



How to Build A Concept Map:
1. Decide what concepts you want to include in your map. Try to think about you concepts from a variety of perspectives to generate a lot of different sub-concepts.
3. Arrange these concepts from general to specific
4. Start trying to build your map (I recommend pencil). Note that the first time you try, it'll likely not be right. A good concept map takes several revisions.

Questions to Ask Yourself To See if Your Map Makes Sense:
1. Does the map show hiearchy? Is each subordinate concept more specific and less general than the concept above it?
2. Is the meaning relationship between two concepts indicated by a linking word?
4. Are there a small number of good examples?
3. Does the cross-hierarchy links show meaningful connections between 2 concepts in different parts of the hierarchy?



Name ____

CS Field Survey v1

Part 1

Please draw a *concept map* representing the field of Computer Science. If possible, please include the following items in your concept map: CS Theory, Compilers, Human-Computer Interaction (HCI), and Logic Gates (e.g. AND OR NOT).

Part 2

Please answer these questions.

1. How would you define Computer Science?

2. What would be your ideal job, post-graduation?

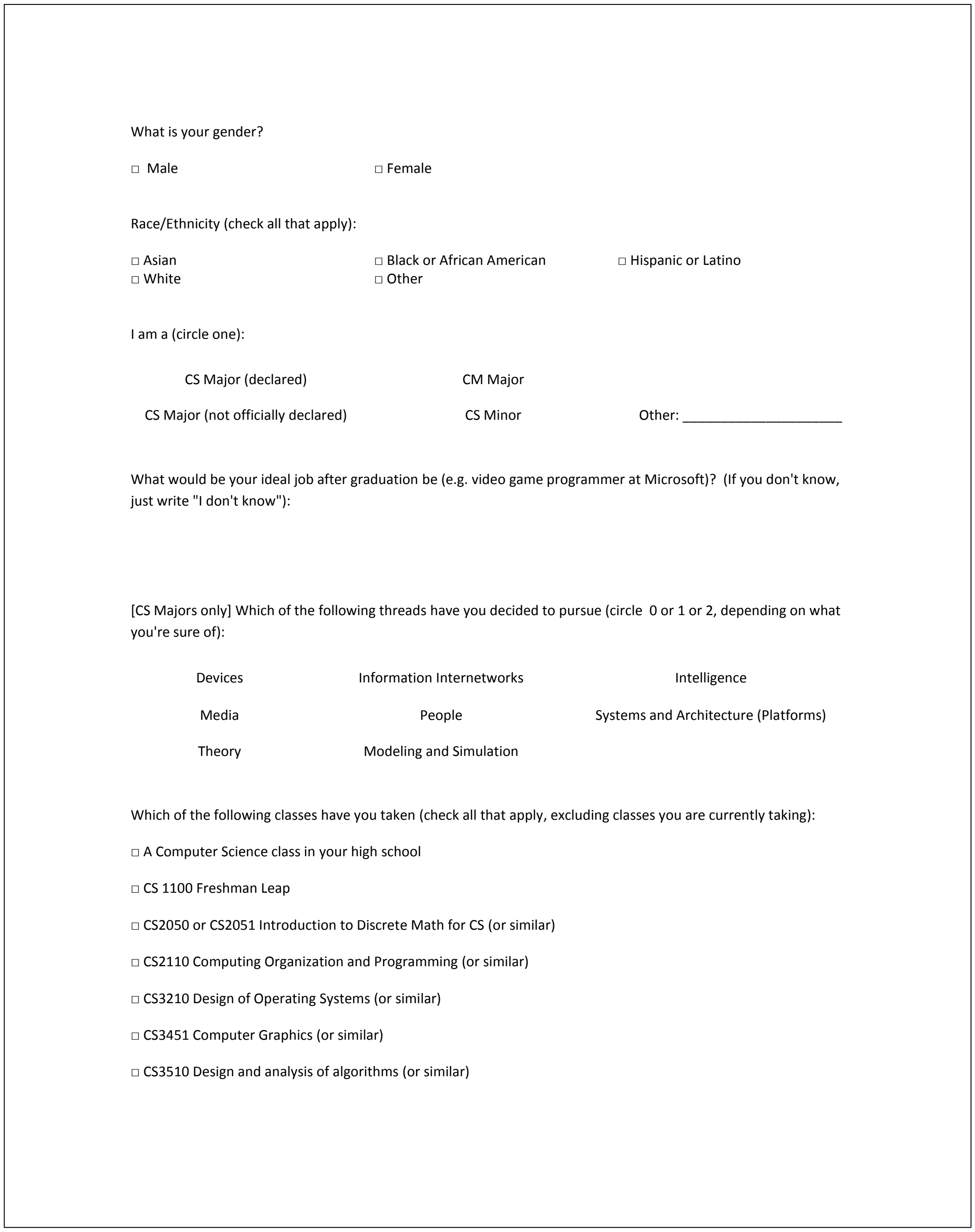
3. What areas of CS do you consider best prepare you for the job you selected in question 2 and why?

4. What types of skills/information would you like to see covered in greater detail in your school's CS curriculum?

APPENDIX B

STUDY 2 MATERIALS

B.1 Survey



What is your gender?

□ Male	□ Female

Race/Ethnicity (check all that apply):

□ Asian	□ Black or African American	□ Hispanic or Latino
□ White	□ Other	

I am a (circle one):

CS Major (declared)	CM Major	
CS Major (not officially declared)	CS Minor	Other: ______________________

What would be your ideal job after graduation be (e.g. video game programmer at Microsoft)? (If you don't know, just write "I don't know"):

[CS Majors only] Which of the following threads have you decided to pursue (circle 0 or 1 or 2, depending on what you're sure of):

Devices	Information Internetworks	Intelligence
Media	People	Systems and Architecture (Platforms)
Theory	Modeling and Simulation	

Which of the following classes have you taken (check all that apply, excluding classes you are currently taking):

□ A Computer Science class in your high school

□ CS 1100 Freshman Leap

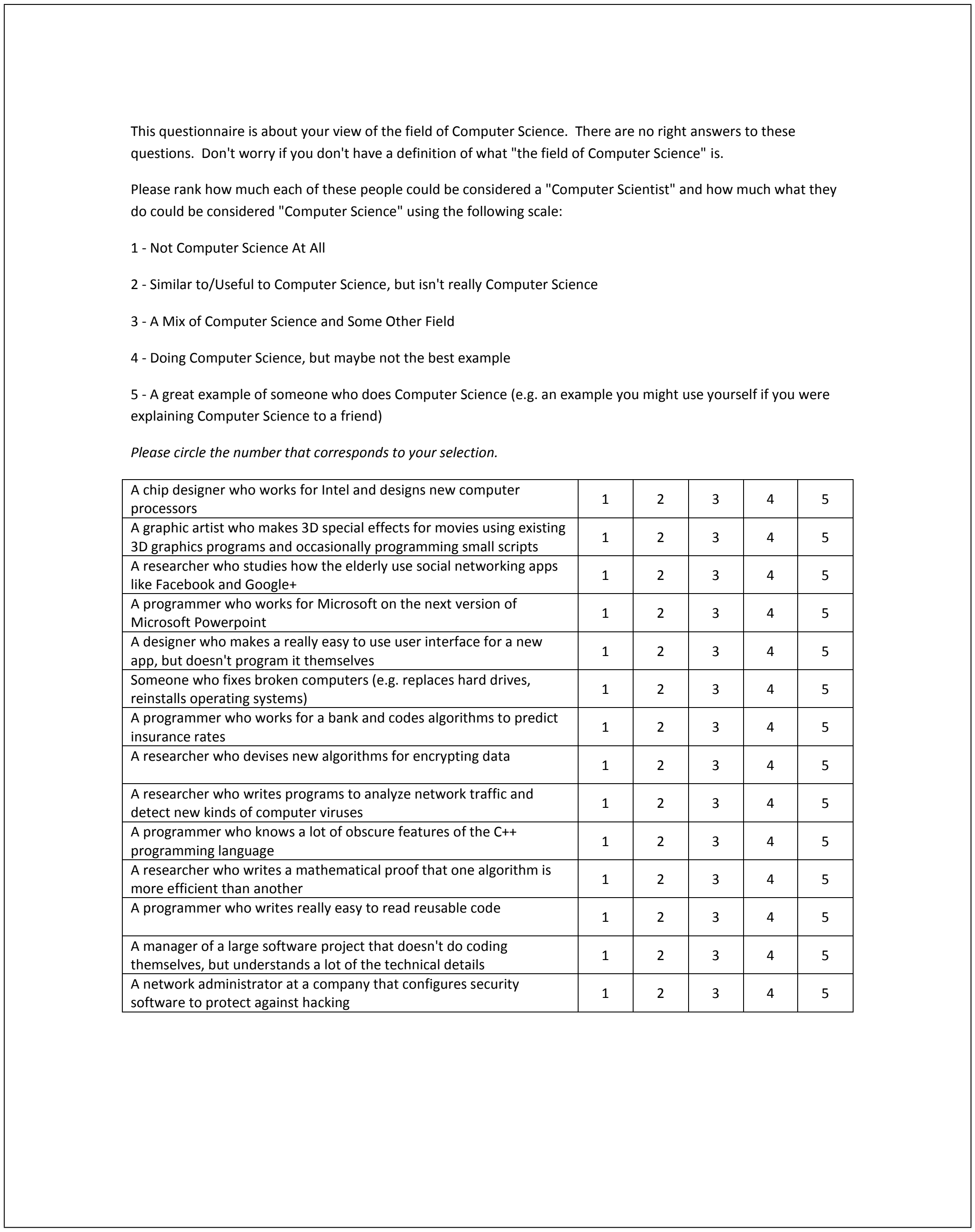
□ CS2050 or CS2051 Introduction to Discrete Math for CS (or similar)

□ CS2110 Computing Organization and Programming (or similar)

□ CS3210 Design of Operating Systems (or similar)

□ CS3451 Computer Graphics (or similar)

□ CS3510 Design and analysis of algorithms (or similar)



This questionnaire is about your view of the field of Computer Science. There are no right answers to these questions. Don't worry if you don't have a definition of what "the field of Computer Science" is.

Please rank how much each of these people could be considered a "Computer Scientist" and how much what they do could be considered "Computer Science" using the following scale:

1 - Not Computer Science At All

2 - Similar to/Useful to Computer Science, but isn't really Computer Science

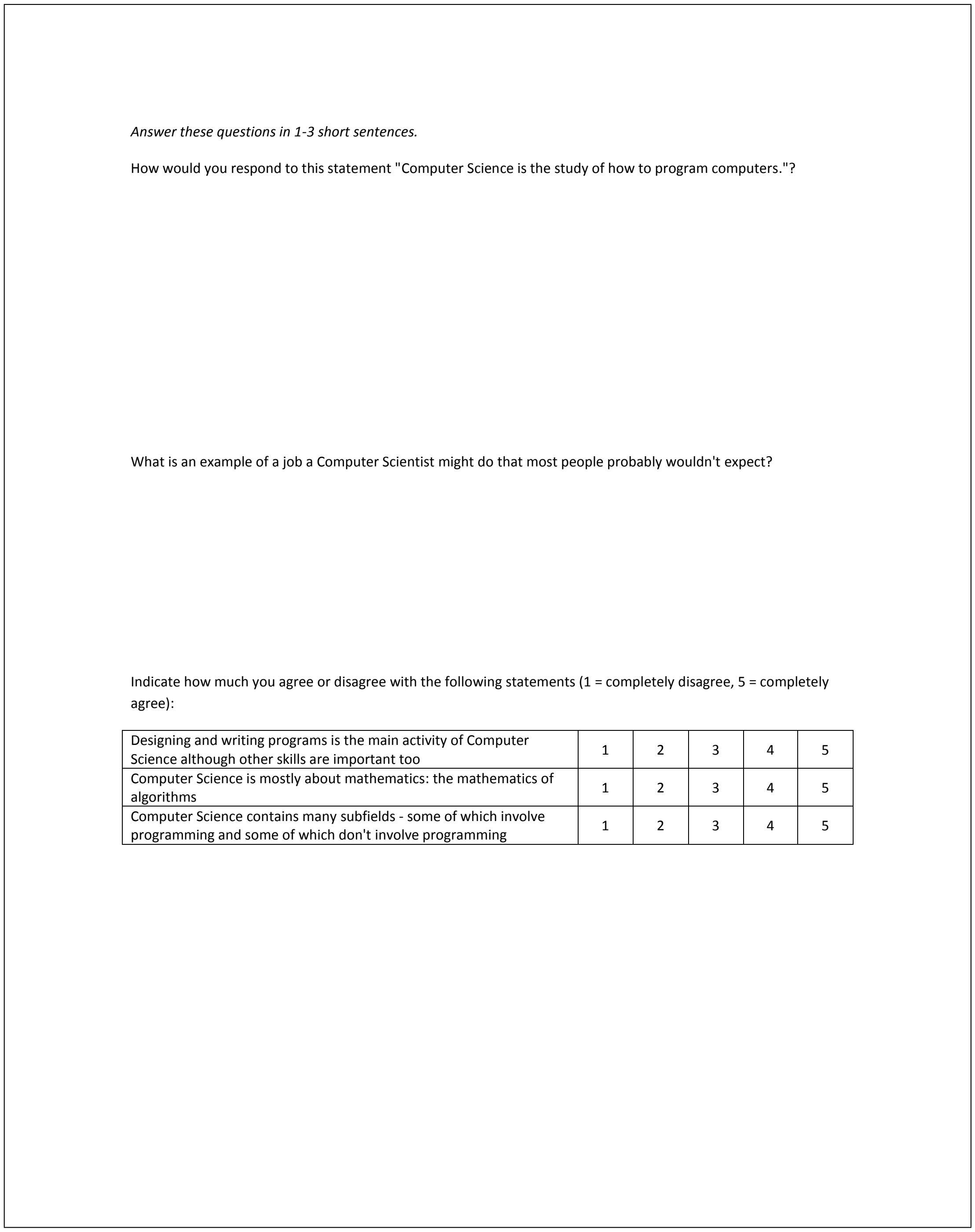
3 - A Mix of Computer Science and Some Other Field

4 - Doing Computer Science, but maybe not the best example

5 - A great example of someone who does Computer Science (e.g. an example you might use yourself if you were explaining Computer Science to a friend)

Please circle the number that corresponds to your selection.

A chip designer who works for Intel and designs new computer processors	1	2	3	4	5
A graphic artist who makes 3D special effects for movies using existing 3D graphics programs and occasionally programming small scripts	1	2	3	4	5
A researcher who studies how the elderly use social networking apps like Facebook and Google+	1	2	3	4	5
A programmer who works for Microsoft on the next version of Microsoft Powerpoint	1	2	3	4	5
A designer who makes a really easy to use user interface for a new app, but doesn't program it themselves	1	2	3	4	5
Someone who fixes broken computers (e.g. replaces hard drives, reinstalls operating systems)	1	2	3	4	5
A programmer who works for a bank and codes algorithms to predict insurance rates	1	2	3	4	5
A researcher who devises new algorithms for encrypting data	1	2	3	4	5
A researcher who writes programs to analyze network traffic and detect new kinds of computer viruses	1	2	3	4	5
A programmer who knows a lot of obscure features of the C++ programming language	1	2	3	4	5
A researcher who writes a mathematical proof that one algorithm is more efficient than another	1	2	3	4	5
A programmer who writes really easy to read reusable code	1	2	3	4	5
A manager of a large software project that doesn't do coding themselves, but understands a lot of the technical details	1	2	3	4	5
A network administrator at a company that configures security software to protect against hacking	1	2	3	4	5



Answer these questions in 1-3 short sentences.

How would you respond to this statement "Computer Science is the study of how to program computers."?

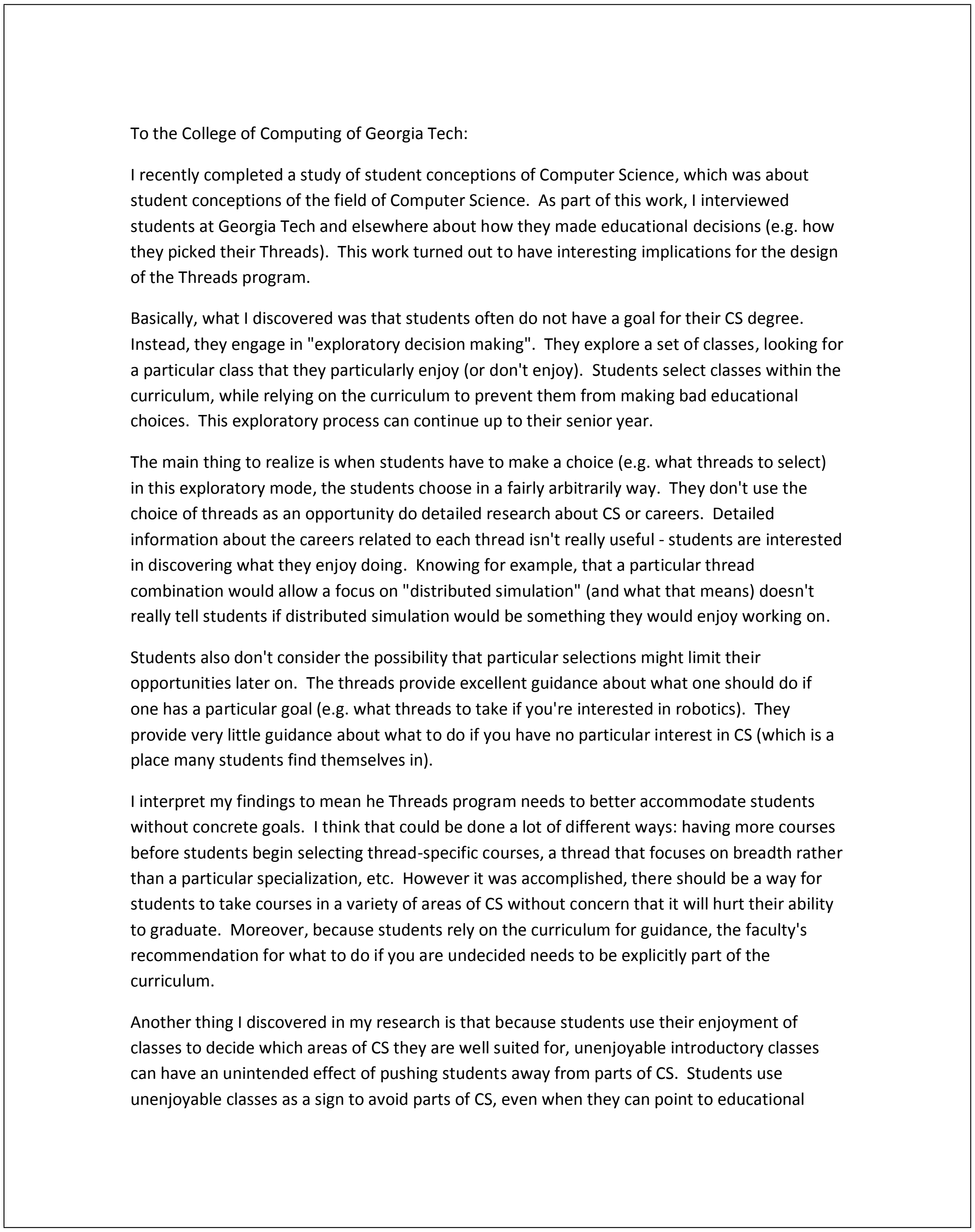
What is an example of a job a Computer Scientist might do that most people probably wouldn't expect?

Indicate how much you agree or disagree with the following statements (1 = completely disagree, 5 = completely agree):

Designing and writing programs is the main activity of Computer Science although other skills are important too	1	2	3	4	5
Computer Science is mostly about mathematics: the mathematics of algorithms	1	2	3	4	5
Computer Science contains many subfields - some of which involve programming and some of which don't involve programming	1	2	3	4	5

APPENDIX C

LETTER TO GEORGIA TECH



To the College of Computing of Georgia Tech:

I recently completed a study of student conceptions of Computer Science, which was about student conceptions of the field of Computer Science. As part of this work, I interviewed students at Georgia Tech and elsewhere about how they made educational decisions (e.g. how they picked their Threads). This work turned out to have interesting implications for the design of the Threads program.

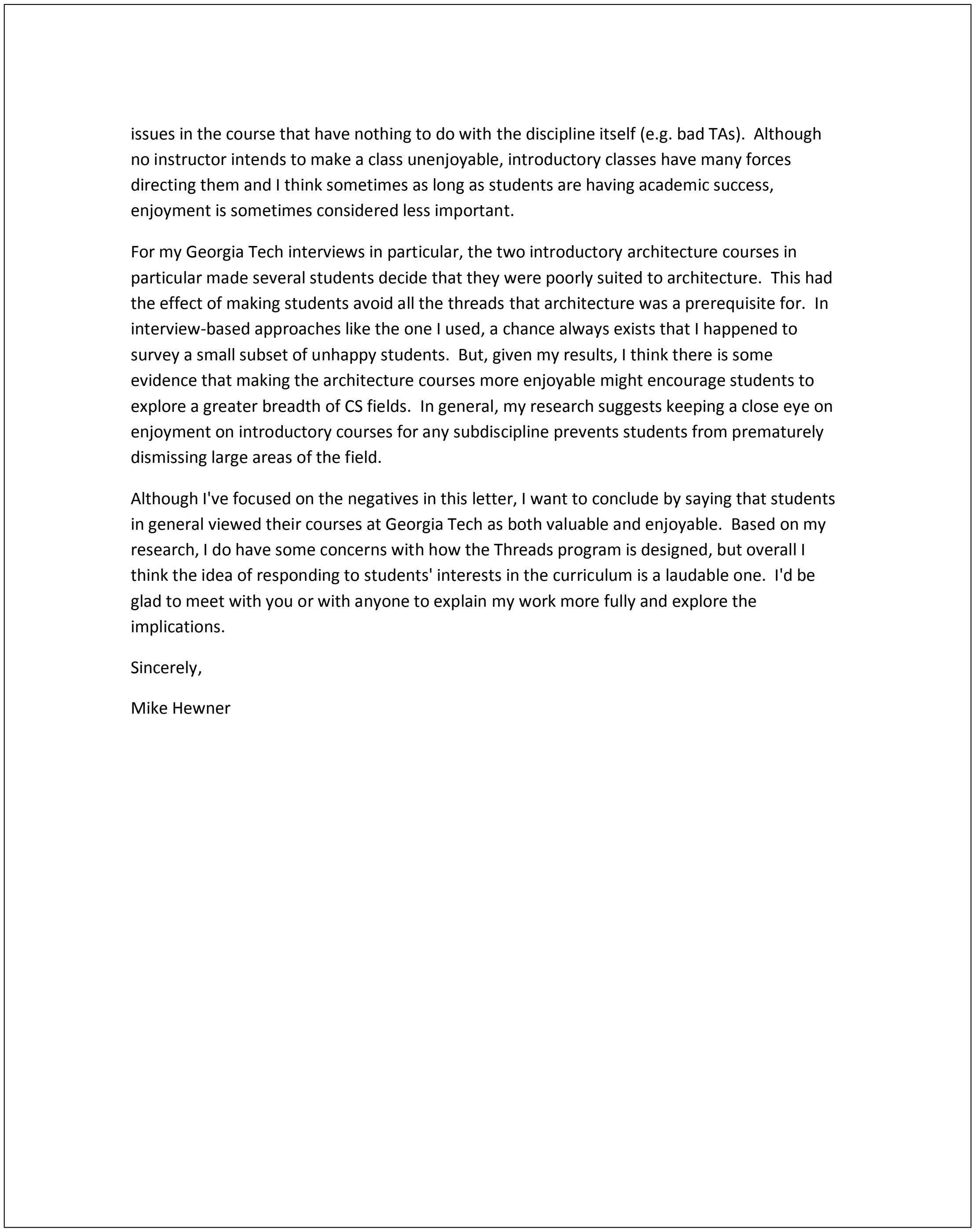
Basically, what I discovered was that students often do not have a goal for their CS degree. Instead, they engage in "exploratory decision making". They explore a set of classes, looking for a particular class that they particularly enjoy (or don't enjoy). Students select classes within the curriculum, while relying on the curriculum to prevent them from making bad educational choices. This exploratory process can continue up to their senior year.

The main thing to realize is when students have to make a choice (e.g. what threads to select) in this exploratory mode, the students choose in a fairly arbitrarily way. They don't use the choice of threads as an opportunity do detailed research about CS or careers. Detailed information about the careers related to each thread isn't really useful - students are interested in discovering what they enjoy doing. Knowing for example, that a particular thread combination would allow a focus on "distributed simulation" (and what that means) doesn't really tell students if distributed simulation would be something they would enjoy working on.

Students also don't consider the possibility that particular selections might limit their opportunities later on. The threads provide excellent guidance about what one should do if one has a particular goal (e.g. what threads to take if you're interested in robotics). They provide very little guidance about what to do if you have no particular interest in CS (which is a place many students find themselves in).

I interpret my findings to mean he Threads program needs to better accommodate students without concrete goals. I think that could be done a lot of different ways: having more courses before students begin selecting thread-specific courses, a thread that focuses on breadth rather than a particular specialization, etc. However it was accomplished, there should be a way for students to take courses in a variety of areas of CS without concern that it will hurt their ability to graduate. Moreover, because students rely on the curriculum for guidance, the faculty's recommendation for what to do if you are undecided needs to be explicitly part of the curriculum.

Another thing I discovered in my research is that because students use their enjoyment of classes to decide which areas of CS they are well suited for, unenjoyable introductory classes can have an unintended effect of pushing students away from parts of CS. Students use unenjoyable classes as a sign to avoid parts of CS, even when they can point to educational



issues in the course that have nothing to do with the discipline itself (e.g. bad TAs). Although no instructor intends to make a class unenjoyable, introductory classes have many forces directing them and I think sometimes as long as students are having academic success, enjoyment is sometimes considered less important.

For my Georgia Tech interviews in particular, the two introductory architecture courses in particular made several students decide that they were poorly suited to architecture. This had the effect of making students avoid all the threads that architecture was a prerequisite for. In interview-based approaches like the one I used, a chance always exists that I happened to survey a small subset of unhappy students. But, given my results, I think there is some evidence that making the architecture courses more enjoyable might encourage students to explore a greater breadth of CS fields. In general, my research suggests keeping a close eye on enjoyment on introductory courses for any subdiscipline prevents students from prematurely dismissing large areas of the field.

Although I've focused on the negatives in this letter, I want to conclude by saying that students in general viewed their courses at Georgia Tech as both valuable and enjoyable. Based on my research, I do have some concerns with how the Threads program is designed, but overall I think the idea of responding to students' interests in the curriculum is a laudable one. I'd be glad to meet with you or with anyone to explain my work more fully and explore the implications.

Sincerely,

Mike Hewner

REFERENCES

[1] Aikenhead, G. S., "Science education: Border crossing into the subculture of science," *Studies in Science Education*, vol. 27, no. 1, p. 1, 1996.

[2] Aikenhead, G. S. and Ryan, A. G., "The development of a new instrument: "views on science–technology–society" (vosts)," *Science Education*, vol. 76, no. 5, pp. 477–491, 1992.

[3] Anderson, J. R., Reder, L. M., and Simon, H. A., "Situated learning and education," *Educational Researcher*, vol. 25, pp. 5 –11, May 1996.

[4] Astin, A. W., *What Matters in College: Four Critical Years Revisited.* Jossey-Bass, Jan. 1997.

[5] Barton, A. C., *Feminist science education.* Teachers College Press, 1998.

[6] Biggers, M., Brauer, A., and Yilmaz, T., "Student perceptions of computer science: a retention study comparing graduating seniors with cs leavers," in *Proceedings of the 39th SIGCSE technical symposium on Computer science education*, (Portland, OR, USA), pp. 402–406, ACM, 2008.

[7] Bransford, J. D., Franks, J. J., Vye, N. J., and Sherwood, R. D., "New approaches to instruction: Because wisdom can't be told," in *Similarity and analogical reasoning*, p. 470497, 1989.

[8] Bruckman, A., Biggers, M., Ericson, B., McKlin, T., Dimond, J., DiSalvo, B., Hewner, M., Ni, L., and Yardi, S., ""Georgia computes!": improving the computing education pipeline," in *Proceedings of the 40th ACM technical symposium on Computer science education*, (Chattanooga, TN, USA), pp. 86–90, ACM, 2009.

[9] Carter, L., "Why students with an apparent aptitude for computer science don't choose to major in computer science," in *Proceedings of the 37th SIGCSE technical symposium on Computer science education*, (Houston, Texas, USA), pp. 27–31, ACM, 2006.

[10] Cassel, L., Clements, A., Davies, G., Guzdial, M., McCauley, R., McGettrick, A., Sloan, R., Snyder, L., Tymann, P., and Weide, B., "Computer science curriculum 2008: An interim revision of CS 2001," *Association for Computing Machinery and IEEE Computer Society*, 2008.

[11] Charmaz, K., *Constructing Grounded Theory: A Practical Guide through Qualitative Analysis.* Sage Publications Ltd, 1 ed., Jan. 2006.

[12] Clancy, M., "Misconceptions and attitudes that interfere with learning to program," *Computer science education research*, p. 85100, 2004.

[13] Clarke, A., *Situational Analysis: Grounded Theory After the Postmodern Turn.* Thousand Oaks, Calif: Sage Publications, 2005.

[14] Clough, E. E. and Wood-Robinson, C., "Childrens understanding of inheritance," *Journal of Biological Education*, vol. 19, no. 4, p. 304310, 1985.

[15] Committee for the Workshops on Computational Thinking and National Research Council, *Report of a Workshop on the Scope and Nature of Computational Thinking.* National Academies Press, 2010.

[16] Committee on Developments in the Science of Learning, *How People Learn: Brain, Mind, Experience, and School: Expanded Edition.* National Academies Press, 2 ed., Sept. 2000.

[17] Committee on Prospering in the Global Economy of the 21st Century, *Rising Above the Gathering Storm: Energizing and Employing America for a Brighter Economic Future.* Washington, D.C.: The National Academies Press, 2007.

[18] Corbin, J. and Strauss, A. C., *Basics of Qualitative Research: Second Edition: Techniques and Procedures for Developing Grounded Theory.* Sage Publications, Inc, 3rd ed., Sept. 2008.

[19] Csikszentmihalyi, M., *Flow : the psychology of optimal experience.* Harper Perennial, 1995.

[20] Deci, E. L. and Ryan, R. M., "The "what" and "why" of goal pursuits: Human needs and the self-determination of behavior," *Psychological Inquiry*, vol. 11, no. 4, pp. 227–268, 2000.

[21] Deci, E. L. and Ryan, R. M., "Facilitating optimal motivation and psychological well–being across lifes domains," *Canadian Psychology*, vol. 49, no. 1, pp. 14–23, 2008.

[22] Denning, P. J., "Is computer science science?," *Commun. ACM*, vol. 48, no. 4, pp. 27–31, 2005.

[23] Denning, P. J., Comer, D. E., Gries, D., Mulder, M. C., Tucker, A., Turner, A. J., and Young, P. R., "Computing as a discipline," *Computer*, vol. 22, no. 2, pp. 63–70, 1989.

[24] Denning, P. J. and McGettrick, A., "Recentering computer science," *Commun. ACM*, vol. 48, no. 11, pp. 15–19, 2005.

[25] Dodds, Z., Libeskind-Hadas, R., Alvarado, C., and Kuenning, G., "Evaluating a breadth-first cs 1 for scientists," in *Proceedings of the 39th SIGCSE technical symposium on Computer science education*, (Portland, OR, USA), pp. 266–270, ACM, 2008.

[26] Donald, J. G., *Learning to Think: Disciplinary Perspectives.* Jossey-Bass, 1 ed., Mar. 2002.

[27] Dweck, C., *Self-theories: Their Role in Motivation, Personality, and Development.* Psychology Press, 1 ed., Jan. 2000.

[28] Eccles, J. S., "Understanding women's educational and occupational choices," *Psychology of Women Quarterly*, vol. 18, pp. 585–609, 1994.

[29] Eccles, J. S. and Wigfield, A., "Motivational beliefs, values, and goals," *Annual Review of Psychology*, vol. 53, pp. 109–132, 2002.

[30] ECCLES, J. S. .-., "Who am i and what am i going to do with my life? personal and collective identities as motivators of action," *Educational Psychologist*, vol. 44, no. 2, p. 78, 2009.

[31] Edward, N. S., "Preaching to the converted," *International Journal of Electrical Engineering Education*, vol. 39, no. 3, pp. 230–237, 2002.

[32] Engle, C., "Software engineering is not computer science," *Software Engineering Education*, p. 257262, 1989.

[33] Erickson, G. L., "Children's conceptions of heat and temperature.," *Science Education*, vol. 63, pp. 221–30, Apr. 1979.

[34] Feil, A. and Mestre, J., "Change blindness as a means of studying expertise in physics," *Journal of the Learning Sciences*, vol. 19, no. 4, pp. 480–505, 2010.

[35] Fincher, S., Tenenberg, J., and Robins, A., "Research design: necessary bricolage," in *Proceedings of the seventh international workshop on Computing education research*, ICER '11, (New York, NY, USA), pp. 27–32, ACM, 2011.

[36] Foundation, W. E. and ACM, "New image for computing: Report on market research." http://www.acm.org/membership/NIC.pdf, 2009.

[37] Furst, M., Isbell, C., and Guzdial, M., "Threads: how to restructure a computer science curriculum for a flat world," in *Proceedings of the 38th SIGCSE technical symposium on Computer science education*, (Covington, Kentucky, USA), pp. 420–424, ACM, 2007.

[38] Gilbert, J. K., Osborne, R. J., and Fensham, P. J., "Children's science and its consequences for teaching," *Science Education*, vol. 66, no. 4, pp. 623–633, 1982.

[39] Gilbert, J. K., Osborne, R. J., and Fensham, P. J., "Children's science and its consequences for teaching," *Science Education*, vol. 66, no. 4, pp. 623–633, 1982.

[40] Greening, T., "Computer science: through the eyes of potential students," in *Proceedings of the 3rd Australasian conference on Computer science education*, (The University of Queensland, Australia), pp. 145–154, ACM, 1998.

[41] Heersink, D. and Moskal, B. M., "Measuring high school students' attitudes toward computing," in *Proceedings of the 41st ACM technical symposium on Computer science education*, (Milwaukee, Wisconsin, USA), pp. 446–450, ACM, 2010.

[42] Herman, G. L., Loui, M. C., and Zilles, C., "Creating the digital logic concept inventory," in *Proceedings of the 41st ACM technical symposium on Computer science education*, (Milwaukee, Wisconsin, USA), pp. 102–106, ACM, 2010.

[43] Hewner, M. and Guzdial, M., "Attitudes about computing in postsecondary graduates," in *Proceeding of the fourth international workshop on Computing education research*, (Sydney, Australia), pp. 71–78, ACM, 2008.

[44] Hewner, M. and Guzdial, M., "How cs majors select a specialization," in *Proceedings of the seventh international workshop on Computing education research*, ICER '11, (New York, NY, USA), pp. 11–18, ACM, 2011.

[45] Hewner, M. and Knobelsdorf, M., "Understanding computing stereotypes with Self-Categorization theory," in *Proceedings of Koli Calling International Conference on Computer Science Education*, (Koli National Park, Finland), 2008.

[46] James, R., "How school-leavers chose a preferred university course and possible effects on the quality of the school-university transition," *Journal of Institutional Research*, vol. 9, no. 1, p. 7888, 2000.

[47] Kaczmarczyk, L. C., Petrick, E. R., East, J. P., and Herman, G. L., "Identifying student misconceptions of programming," in *Proceedings of the 41st ACM technical symposium on Computer science education*, (Milwaukee, Wisconsin, USA), pp. 107–111, ACM, 2010.

[48] Kuhn, D., "Children and adults as intuitive scientists.," *Psychological review*, vol. 96, no. 4, p. 674, 1989.

[49] Kuhn, D., "Science as argument: Implications for teaching and learning scientific thinking.," *Science Education*, vol. 77, no. 3, p. 31937, 1993.

[50] Lave, J. and Wenger, E., *Situated Learning: Legitimate Peripheral Participation.* Learning in doing, Cambridge, UK: Cambridge University Press, 1991.

[51] Lederman, N. G., "Students' and teacher's conceptions of the nature of science: A review of the research," *Journal of Research in Science Teaching*, vol. 29, no. 4, pp. 331–359, 1992.

[52] Lewis, T. L. and Smith, W. J., "The computer science debate: it's a matter of perspective," *SIGCSE Bull.*, vol. 37, no. 2, pp. 80–84, 2005.

[53] Lincoln, Y. S. and Guba, E. G., *Naturalistic inquiry.* SAGE, 1985.

[54] Linn, M. C. and Muilenburg, L., "Creating lifelong science learners: What models form a firm foundation?," *Educational Researcher*, vol. 25, pp. 18 –24, June 1996.

[55] Maloney, J. H., Peppler, K., Kafai, Y., Resnick, M., and Rusk, N., "Programming by choice: urban youth learning programming with scratch," *SIGCSE Bull.*, vol. 40, no. 1, pp. 367–371, 2008.

[56] Margolis, J., *Stuck in the Shallow End: Education, Race, and Computing.* The MIT Press, Sept. 2008.

[57] Margolis, J. and Fisher, A., *Unlocking the Clubhouse: Women in Computing.* The MIT Press, Apr. 2003.

[58] Martin, C. D., "The case for integrating ethical and social impact into the computer science curriculum," in *The supplemental proceedings of the conference on Integrating technology into computer science education: working group reports and supplemental proceedings*, (Uppsala, Sweden), pp. 114–120, ACM, 1997.

[59] McGuffee, J. W., "Defining computer science," *SIGCSE Bull.*, vol. 32, no. 2, pp. 74–76, 2000.

[60] Nespor, J., *Knowledge in motion: Space, time, and curriculum in undergraduate physics and management.* Routledge, 1994.

[61] Novak, J. D. and Gowin, D. B., *Learning How to Learn.* Cambridge University Press, 1 ed., Sept. 1984.

[62] Osborne, R. J. and Gilbert, J. K., "A technique for exploring students' views of the world," *Physics Education*, vol. 15, no. 6, pp. 376–379, 1980.

[63] Papert, S., *Mindstorms: children, computers & powerful ideas.* Prentice Hall / Harvester Wheatsheaf, 1982.

[64] Rasmussen, B. and Hapnes, T., "Excluding women from the technologies of the future? a case study of the culture of computer science," *Futures*, vol. 23, no. 10, pp. 1107–1119, 1991.

[65] Rosenbloom, P. S., "A new framework for computer science and engineering," *Computer*, vol. 37, no. 11, p. 2328, 2004.

[66] Rushkoff, D., *Program or be Programmed.* OR Books, 2010.

[67] Ryan, A. G. and Aikenhead, G. S., “Students preconceptions about the epistemology of science,” *Science Education*, vol. 76, no. 6, pp. 559–580, 1992.

[68] Sandoval, W. A., “Understanding students’ practical epistemologies and their influence on learning through inquiry,” *Science Education*, vol. 89, no. 4, pp. 634–656, 2005.

[69] Schulte, C. and Knobelsdorf, M., “Attitudes towards computer science-computing experiences as a starting point and barrier to computer science,” in *Proceedings of the third international workshop on Computing education research*, (Atlanta, Georgia, USA), pp. 27–38, ACM, 2007.

[70] Shackelford, R., McGettrick, A., Sloan, R., Topi, H., Davies, G., Kamali, R., Cross, J., Impagliazzo, J., LeBlanc, R., and Lunt, B., “Computing curricula 2005: The overview report,” *SIGCSE Bull.*, vol. 38, no. 1, pp. 456–457, 2006.

[71] Shamos, M. H., *The myth of scientific literacy.* Rutgers University Press, 1995.

[72] Sherman, R. R. and Webb, R. B., *Qualitative research in education: Focus and methods.* Routledge, 1988.

[73] Smith, J. P., diSessa, A. A., and Roschelle, J., “Misconceptions reconceived: A constructivist analysis of knowledge in transition,” *Journal of the Learning Sciences*, vol. 3, no. 2, pp. 115 – 163, 1994.

[74] Songer, N. B. and Linn, M. C., “How do students’ views of science influence knowledge integration?,” *Journal of Research in Science Teaching*, vol. 28, no. 9, pp. 761–784, 1991.

[75] Stevens, R., OConnor, K., Garrison, L., Jocuns, A., and Amos, D. M., “Becoming an engineer: Toward a three dimensional view of engineering learning,” *Journal of Engineering Education*, vol. 97, no. 3, p. 355368, 2008.

[76] Sudol, L. A. and Jaspan, C., “Analyzing the strength of undergraduate misconceptions about software engineering,” in *Proceedings of the Sixth international workshop on Computing education research*, (Aarhus, Denmark), pp. 31–40, ACM, 2010.

[77] Tew, A. E., Fowler, C., and Guzdial, M., “Tracking an innovation in introductory CS education from a research university to a two-year college,” *SIGCSE Bull.*, vol. 37, no. 1, pp. 416–420, 2005.

[78] Tew, A. E. and Guzdial, M., “Developing a validated assessment of fundamental CS1 concepts,” in *Proceedings of the 41st ACM technical symposium on Computer science education*, (Milwaukee, Wisconsin, USA), pp. 97–101, ACM, 2010.

[79] Wandersee, J. H., Mintzes, J. J., and Novak, J. D., "Research on alternative conceptions in science," *Handbook of research on science teaching and learning*, p. 177210, 1994.

[80] Wing, J. M., "Computational thinking," *Communications of the ACM*, vol. 49, no. 3, pp. 33–35, 2006.

[81] Yardi, S. and Bruckman, A., "What is computing?: bridging the gap between teenagers' perceptions and graduate students' experiences," in *Proceedings of the third international workshop on Computing education research*, (Atlanta, Georgia, USA), pp. 39–50, ACM, 2007.