Commonsense Computing (episode 3): Concurrency and Concert Tickets

Gary Lewandowski
Department of Mathematics and Computer Science
Xavier University
Cincinnati, OH 45207 USA
lewandow@cs.xu.edu

Dennis J. Bouvier
Dept. of Computer Science
Southern Illinois Univ. Edwardsville
Edwardsville, IL 62026 USA
djb@acm.org

Robert McCartney
Dept. of Computer Science and Engineering
University of Connecticut
Storrs, CT 06269 USA
robert@cse.uconn.edu

Kate Sanders
Department of Math and Computer Science
Rhode Island College
Providence, RI 02908 USA
ksanders@ric.edu

Beth Simon
Dept. of Computer Science and Engineering
Univ. of California San Diego
La Jolla, CA 92093 USA
bsimon@cs.ucsd.edu

ABSTRACT

As the third in a series of projects investigating commonsense computing – the relevant knowledge that students have before any formal study of computing – we examine students' commonsense understanding of concurrency. Specifically, we replicated (with modifications) an experiment by Ben-David Kolikant. [2] Ben-David Kolikant's data were gathered from high-school seniors who had previously studied computing, at the beginning of an advanced class in concurrent and distributed programming. Modifying one of her questions to reflect our students' lack of background, we asked students at five different institutions, in the first week of CS1, to describe in English the problems that might arise when more than one person is selling seats to a concert.

Almost all students (97%) identified the problem of interest – that a race condition may occur between sellers. 73% of students identified at least one possible solution. We found that the categorizations developed by Ben-David Kolikant were also meaningful when applied to our data, that our beginning CS1 students are more likely to give centralized solutions (as opposed to decentralized ones) than Ben-David Kolikant's concurrency students, and that the granularity of solutions is finer among the more experienced students.

Categories and Subject Descriptors

K.3.2 [**Computer Science Education**]: Introductory Programming—*abstract programming concepts*

ICER'07, September 15–16, 2007, Atlanta, Georgia.

General Terms

Algorithms, Human Factors

Keywords

preconceptions, concurrency, constructivism, resources, CS1, novices, beginners

1. INTRODUCTION

This paper reports on the third in a series of projects investigating "commonsense computing": what students know about computing concepts before having formal instruction. Recent studies of computing students indicate that students lack certain skills: both their ability to write programs [16], and their ability to read and trace code [15] are well below what we might expect. Ben-David Kolikant [3] found that students apply a sense of "mostly correct" to their programs, suggesting they do not even know what it would mean for their programs to work. These results are independent of the programming language and paradigm of instruction.

Other studies have also demonstrated that students have considerable ability to reason about computer science topics. Gibson and O'Kelly [13] found pre-college students could solve a variety of search problems and beginning computer-science students could prove results about the Towers-of-Hanoi. This paper follows up work by Ben-David Kolikant [2], who found students beginning a concurrent-and-distributed-programming class could reason about concurrency.

This disconnect between demonstrated programming knowledge and demonstrated reasoning skills suggests students have considerable knowledge that we, as instructors, can leverage to teach computer science more effectively. To leverage students' existing knowledge, however, we must first determine what that knowledge is. As a result, we have begun to investigate what students with no prior computing instruction ("beginners") know about computing.

In our earlier projects, we found that most of the beginners could give an algorithm to solve a sorting problem. They used both conditionals and post-test loops. Many even suggested algorithms that are recognizable as selection or in-

sertion sort. Their understanding of the way numbers are represented was unexpected, however, and led to suggestions for possible interventions by instructors. [8, 23, 9]

In this project, we adapt a question from Ben-David Kolikant's study [2] to examine beginners' understanding of concurrency. Examining concurrency provides a broader perspective of commonsense computing. The answers of beginners provide evidence of computer science problem solving skills, in terms of task analysis to see the problem and in terms of an ability to suggest solutions. The particular advantage of replicating the Ben-David Kolikant study is additional perspective in both analysis and results. Using this study as a template constrains categorizations we make but also demonstrates that these categorizations can in fact apply to complete beginners. This study also allows a comparison between beginners and students who have significant computer science skills though they do not have significant experience with concurrency. This comparison provides a sense of how much additional sophistication has been gained by the more experienced students and how much problem-solving sensibility has been with the students from the day they entered the first course.

Our questions were:

- Would beginners be able to recognize the key concurrency issue regarding a critical section?
- Would they give answers similar to those reported by Ben-David Kolikant?
- Would the categories developed by Ben-David Kolikant even make sense when applied to our data?

We find, consistent with Ben-David Kolikant, that students are able to recognize at least one key concurrency issue (duplicate sales) but few are able to recognize the problem of interleaving accesses of non-atomic operations. In contrast to Ben-David Kolikant, the solutions are more likely to be centralized than decentralized. However, the categories developed by Ben-David Kolikant are reasonable when applied to the beginners, suggesting that students can reason about concurrency even as they begin studying computer science. As in our earlier studies, we thus see that students are not so much "learning problem solving" as they are learning how to express solutions in a programming language and to do a finer-grained analysis of problems.

The rest of this paper is organized as follows. In Section 2 we review the related work on preconceptions and concurrency, particularly Ben-David Kolikant's work [2] upon which this study is based. In Section 3 we discuss our research methodology, specifically how and where we collected the data, and how we analyzed it. In Section 4, we present our results, and discuss them further in Section 5. Finally we conclude in Section 6 with some suggestions for future work.

2. RELATED WORK

Although we are examining a different topic, this project has the same basic philosophy as our earlier commonsense computing projects [8, 23, 9]. In our earlier projects, we examined students' commonsense knowledge of sorting, asking students to write a paragraph describing how they would sort a list of ten numbers (or dates).

All of our commonsense computing projects are motivated by the constructivist view of learning, which holds that learning takes place by refining and extending what the student already knows [1, 7]. Bransford et al. [6] argue that learning must engage the students' preconceptions to be effective. Schwill [21] applies Bruner's notion of fundamental ideas [7] to computing, and argues that the fundamental computing ideas, which provide a framework for learning constructively, have meaning in everyday life, and can be described in ordinary language.

Several researchers have studied *student preconceptions*:

- Miller [17] analyzed "natural language" programs by students who had not had a formal programming course, with the purpose of exploring the idea of writing computer programs in natural language. He found that a number of standard programming concepts showed up in these natural language descriptions, but that there were differences between these and programs in computer languages, especially in terms of knowledge implicit in context or general world knowledge.
- Onorato and Schvaneveldt [18] also looked at natural language descriptions of a programming task, comparing subjects drawn from different pools: *naïve*–students with no programming experience, *beginner*–students currently taking their first programming course, and *expert*–students with a good deal of programming experience. Along with differences between experts and novices, they also found differences between the naïves and beginners though neither had experience programming.
- While studying misconceptions of novice programmers, Bonar and Soloway [5] specifically considered preprogramming knowledge, which they call "step-by-step natural language programming knowledge." They distinguish this preprogramming knowledge from knowledge of the programming language Pascal, which the students were learning in their introductory course. They found that many of the observed bugs could be explained by a mismatch between students' knowledge in these two different domains.
- Gibson and O'Kelly [13] looked at a variety of search problems (with pre-college students) and Towers-of-Hanoi problems (with beginning computing students), and found that both groups showed "algorithmic understanding" of how to solve these problems—they were able to consider and reason about the process used to solve the problem, not just find a solution.

Like Smith et al. [24], Hammer [14], and Ben-David Kolikant [2], we seek preconceptions that might be built upon to help students learn specific concepts within a particular context.

Many studies [4, 12, 20, 25, 26] have tried to *predict performance* in computing courses using beginners' characteristics such as math background, gender, age, attitude, and prior programming experience. The BRACE study [22] looked at beginning-student performance on non-computing tasks—paper folding, map sketching, and telephone directory searching (as in [18])—that could relate to student preconceptions about computing topics. They then used these data to predict student performance in a programming course, and found that some things detected in the task performance were positively correlated with final course grade attained. Rather than trying to predict performance, we

are trying to characterize student knowledge, as a basis for deciding how to teach. The data we use are also richer than many of the predictive factors.

There is a substantial body of work both in computing and other disciplines on *misconceptions*: incorrect concept understandings that need to be replaced with correct models. Clancy [10] provides a survey of this work in computer science; the National Academy's Committee on Undergraduate Science Education [11] (Ch. 4) gives a more general overview. Smith et al. [24] challenge this view in the context of math and science education, arguing that misconceptions are limited mental models that can be built upon to gain correct understanding. Like Smith et al. our intent is to build on our students' understandings, rather than to replace them.

Most important to this particular project is Ben-David Kolikant's work examining student preconceptions about concurrency. [2] She found that students without any background in concurrency were able to draw upon real-life experience to come up with the necessary mechanisms to solve simple problems.

Her data were gathered from about 140 Israeli 12th-grade high-school students in six classes in three different schools. The students had previously studied computing, and were taking an advanced class in concurrent and distributed computing. At the beginning of the course she gave these students a critical-section problem where multiple agents share a common resource: two offices selling tickets for the same movie. As shown in Figure 1, the question is posed in a detailed, pseudo-code format. It expects a relatively sophisticated answer, including a hardware system specification for supporting multiple machines servicing sales requests, pseudo-code for the solution, and an explanation of the answer and how it avoids duplicate ticket sales. Students' answers were graded as part of the course.

Ben-David Kolikant divided responses into two major categories. *Centralized* responses involve a solution in which communication and control for the solution is centralized. *Decentralized* responses implicitly or explicitly involve the sellers communicating with each other to achieve a concurrent solution.

Within the centralized solutions, she found three subcategories: those with a central entity (C1), essentially a master computer that makes all decisions; those in which the solutions involve an assumption either of a constant rate of operations or an assumption that operations happen in a particular order (C2); and those in which solutions assume that sellers have private resources (C3), each selling tickets for a separate area of the theater. She describes categories C2 and C3 as solutions that "attempted to solve a similar, but different problem" [2, p. 235]. Decentralized solutions are divided into those solutions in which communication is implicit (D1) and those in which is explicit (D2).

Ben-David Kolikant's discussion focuses on three aspects that together contribute to the solutions she evaluated. She describes these as "(a) the algorithmic goal of the system, (b) synchronization goals, and (c) reasonableness" [2, p. 236].

The "algorithmic goal" is the problem the system is intended to solve: in this case, selling the best ticket possible. The cinema-ticket problem was very well constrained, so the students' responses all shared this goal.

The synchronization goals of the cinema-ticket problem

The cinema tickets problem

A ticket office sells movie tickets for a certain cinema. The next client always gets the best available ticket. Computer software decides what is the next best available seat, and prints the ticket.
Assumptions:

- The movie is only screened once.
- This is the only office that sells tickets for this movie.
- Each client can only buy exactly one ticket.
- There are many people waiting to buy the tickets.

The following procedures are defined in the software:

Function: BestAvailableSeat()	Input: The Hall Return value: Best available seat in the Hall. -1 if no available seat.
Procedure: MarkAvailable-Seat(Seat)	Input: Seat is the place of an available seat in the Hall. On output: the place of Seat is marked as taken.
Procedure: PrintTicket(Seat)	Input: Seat is the place of an available seat in the Hall. On output: a ticket for place Seat is printed.

Handling a client is done as follows:

Seat ⇐ BestAvailableSeat()
If Seat <> -1 then
MarkAvailableSeat(Seat)
PrintTicketSeat(Seat)

Since the waiting time in the line was too long, the owners decided to add another ticket office. Both offices shall be opened at the same time and sell tickets for the same movie screening. Each office shall have its own printer for printing the tickets it sells. (for this exercise you don't need to deal with the money issues).
You have to develop the system according to the following steps:

1. What is the required hardware? (Screens, Printers, keyboards, others). Specify how the hardware is distributed in the system.
2. Write PseudoCode for the software of the required system (selling tickets in 2 offices). You may use the procedures in table 1. No need to redefine them.

Figure 1: Ben-David Kolikant's cinema-tickets assignment

are "to coordinate ... access to a common resource (the database) in order to avoid selling the same ticket twice" [2, p. 236] and "the prevention of the interleaving of the access to the database" [2, p. 237]. She found that students did not identify the problem of interleaving in their written answers. She followed up with interviews in order to investigate whether they had failed to write it down, or there was in fact a lack of understanding. She found that

the students assumed that the key actions were inseparable: "they assume that the two critical actions of checking and updating the database are always executed successively" [2, p. 238]. As a result, they assumed that the key issue was communication: making sure that all sellers were aware of the seats that had already been sold.

In general, she found that a significant number of students used centralized solutions, although this is somewhat less true for the cinema ticket problem than for her other two assignments. Here, 33% of students presented centralized solutions and 67% presented decentralized solutions.

The final aspect of the students' solutions that Ben-David Kolikant considered is "reasonableness:" solving the problem "in a reasonable or realistic way ... according to the context of the problem" [2, p. 238]. One might also think of this as modeling – understanding the relationship between the solution and the original problem domain.

Ben-David Kolikant found that some students did seem to understand the modeling issues, but still simplified the problem in a way they admitted was not realistic, just so that they could solve it. She found this not in the cinema-ticket problem, but in her second problem, which involved gardeners planting trees. Students made the assumption that all the gardeners worked at the same rate. One commented, "I assumed it, to make the algorithm succeed. In spite of reality" [2, p. 239]. This is reminiscent of the old joke about the engineer and the economist who fall into a deep pit. The engineer looks for stones that they can pile up so they can climb out of the pit. The economist says, "First, assume we have a ladder ...."

Ben-David Kolikant also notes several issues with regard to the students' answers. First, she argues that solution type C3 – dividing the tickets among the different sellers – oversimplifies the problem. While this is a workable solution that can be translated back into the real-world situation, it does not ensure that each buyer will get the best seat currently available. Second, she found that students were influenced by their real-world experience with concurrency and networks. In the real world, for example, it may seem that speaking is an action, but hearing just happens. (Of course, even in the real world, the listener is not guaranteed to be paying attention.) Finally, she considered the students' computational model. Her question required the students to suggest hardware and software to solve the problem. The students showed a strong preference for using multiple computers, rather than time-sharing a single computer.

3. METHODOLOGY

3.1 The students

In this study, in the first week of a CS1 class, students were each randomly assigned two tasks. One of the possible tasks was the concurrency task that we explore here. This task was completed by 66 students from five different institutions. The participating students were all beginners.

All but eight of the students completed these questions online (outside of class), by typing English answers into a text box. Eight students (all at the same institution) completed this question on paper in a laboratory setting. All subjects were given credit for completing the assignment, though the quality of the solution was not evaluated for credit.

Subject identifiers presented in this paper have been renumbered and do not reflect institutional affiliation. The institutions' characteristics, which vary significantly, are summarized in Table 1.

3.2 The task

Students were asked to answer the following question:

> Suppose we sell concert tickets over the telephone in the following way – when a customer calls in and asks for a number (n) of seats, the seller 1) finds the n best seats that are available, 2) marks those n seats as unavailable, and 3) deals with payment options for the customer (e.g. getting credit or debit card number, or sending the tickets to the Will Call window for pickup).
>
> Suppose we have more than one seller working at the same time. What problems might we see, and how might we avoid those problems?

There are several differences between this task and Ben-David Kolikant's. We modified the question so that it would refer to concert tickets, since our students likely have more experience with ordering concert tickets. Second, we removed the restriction that each buyer can only buy one ticket. And finally, due to our students' lack of background, it was necessary to phrase the question less technically and to ask for responses in English paragraphs, without pseudocode or detailed hardware and software specifications.

3.3 Analysis

After the data were collected, researchers read through all of the responses to get a general sense of the responses. We then set up categorizations for the responses based on the categorizations used by Ben-David Kolikant [2].

In our study, we consider C2 and C3 answers (along with C1 and D) to be "reasonable" solutions to the question posed – in part based on the less explicit form of our question. More detail on the differences in Ben-David Kolikant's methodology and analysis and that used in this study is given in Section 5.1.

Having read through the solutions, we determined that in many cases the student response was not clear about explicit or implicit communication – resulting in only one decentralized categorization. There were also cases in which the student response was ambiguous and could not be described as either centralized or decentralized. In others, the response could not be reasonably said to solve the problem. Some responses did not offer a solution. This set of categories comprises our "non-reasonable" solutions. Finally, some responses provided solutions to problems that were not our central focus, though they may have been interesting problems – some involving concurrency issues. Along with tagging responses that gave problems we did not focus on, we recorded the problem suggested so we could examine this list for commonalities across students.

Unlike Ben-David Kolikant's participants, the students in our study often gave more than one possible solution to the problem. We counted and coded each solution. After determining categories, all five researchers coded the data. There were only a few conflicts in our coding, easily resolved through discussion.

4. RESULTS

In this section we provide a sense of the responses from the students, first giving overall characteristics of the responses

Institution Characterization	n	Class Characteristics
Private institution on west coast of USA with approx. 3,200 undergraduates and a school of engineering	25	Students 18-20, a CS1 serving all engineering, most of class is electrical engineering majors with a prior matlab course
Public research institution on west coast of USA with approx. 27,000 undergraduates and a school of engineering	20	Students 18-20, a CS1 serving mostly CS majors
Public research institution on east coast of USA with approx. 21,000 undergraduates and a school of engineering	10	Students 18-20, a CS1 serving computer science and/or engineering and electrical engineering, approximately 75% of class is computing majors
Public regional institution on east coast of USA with approx. 2,000 undergraduates	8	Students 18-20, a CS1 serving mostly CS majors
Private liberal arts institution on east coast of USA with approx. 1,600 undergraduates	3	Students 18-20, a CS1 serving CS majors and minors and math majors

Table 1: Institutional Breakdown of Respondents. n is the number of students answering the concurrency question.

Accomplishment	percent of students
Number of solutions provided	
1 solution	70%
2 solutions	20%
3 or more solutions	10%
Problems identified	
Sell seat more than once	97%
Other	41%
Provided "reasonable" solution to concurrency problem	71%

Table 2: Number of solutions and problems identified by student ($n = 66$)

on a per student basis – how many solutions did the student provide? was the problem identified? did the student's solution seem reasonable? Next we look at the solutions, examining them from the perspective of the categorizations used by Ben-David Kolikant. Finally, we provide a qualitative look at the solutions with characteristic examples of responses highlighting important aspects of the responses.

4.1 Per student

The 66 students in the study collectively produced a total of 97 identified solutions. Table 2 summarizes characteristics of these solutions. Because of the descriptive nature of the solution requested, many students would discuss multiple issues they saw stemming from the problem statement, or outline several different solutions to the particular problem of trying to sell the same seat to more than one person at a time. The majority of students did discuss only one solution (70%), but 20% identified two solutions and 10% identified three or more solutions (with six solutions being the most identified by any one student).

Of the 66 students, 97% at least identified our main problem of interest – that it may be possible to sell a given seat to more than one person – good evidence that even novice students can identify this critical concurrency issue. Additionally, Six students explicitly noted the problems of interleaving access to the database that may result in one customer reserving but another customer buying the tickets. 71% of all students (73% of those who identified the main problem) did identify at least one "reasonable" solution to the problem (an answer that is categorized as C or D). Moreover, many of the beginners gave more than one type of answer. 12/47 (26%) of students who gave a reasonable answer, actually gave both centralized and decentralized answers.

Finally, many beginners (41%) identified a problem beyond that of selling the same seat. Other problems identified included: group sales of a large ticket block, having seats be reserved, but not paid for, problems with identity theft, choosing seats by price rather than best available, payment transaction delays, and data tracking and storage.

4.2 Per solution

As many students gave multiple solutions in their answers, it is useful to look at the diversity of responses out of the total number of solutions provided – 97. Table 3 provides the breakdown on solution types.

We find that 69% of the solutions provided are reasonable solutions to the multiple seat selling problem. 31% of the solutions are not reasonable solutions; the majority of these are cases in which the student described the problem (often correctly) but did not offer any solution to the problem.

Of the reasonable solutions we found 55% to describe a centralized solution where the selling entities passed the responsibility of making a seat assignment on to some central resource and 45% of solutions describing a method by which individual sellers in some way made decisions about seat assignments as individual entities.

The centralized solutions can further be broken down into three categories. 10% of reasonable solutions relied on implicit communication between "dummy" sellers and a master resource to make assignments. 13% of reasonable solutions uses the same master resource but requires some explicit ordering of communication or steps in the process, which may include lock-stepping or pipelining the process. By far the most popular centralized solution (31%) is also the most restrictive and involves distributing or dividing up resources either by portioning out seats in the concert venue to different sellers or serializing or otherwise pipelining the selling process.

4.3 Qualitative Results

4.3.1 Algorithmic Goals

Many students did not further refine the goal of their algorithm and either explicitly or implicitly used a goal of "best seats available" in their response. Some students elaborated further as they explained why they chose the particular so-

Category	Of all Solutions	Of reasonable Solutions
Reasonable Solutions (centralized and decentralized)	**69%**	**-**
Centralized	*38%*	*55%*
C1	7%	10%
C2	9%	13%
C3	22%	31%
Decentralized	*31%*	*45%*
Not Reasonable Solutions	**23%**	**-**
Bad Solution	5%	-
No Solution	16%	-
Ambiguous	1%	-
Solved different problem	**8%**	**-**

Table 3: Solution Breakdown by type. Column 2: considered over the set of 97 solutions. Column 3: considered over the set of 67 reasonable (C or D) solutions.

lution or even modified the point of concentration in their algorithm.

A number of students were concerned less about choosing the seats than in handling seats that are given up, for example:

> If the seats are marked as unavailable as soon as they are requested by the customer, other sellers cannot access these seats for their own customers at that time. This is a bad thing because those better seats reserved by the first customer may potentially still be open should the customer change their mind about the purchase or if payment information cannot be validated. If the seats are marked unavailable and the payment does not come through for whatever reason, the seats will remain unavailable and be empty during the concert. [ID415]

> One obvious problem that could appear is two sellers giving up the same seats at the same time. [ID405]

Some students changed their algorithmic goal when they recognized that one could not simply reserve and sell as one atomic action:

> ... if more than one seller is dealing with customers at the same time. In a very unlikely situation, the sellers could mark the seats unavaliable at the same time. However, in a more likely situation, one of them marks seats unavaliable and the other seller marks and sees that the seats are unavailible, but that seller was not the one that reserved them. Then there will be multiple tickets sent to Will Call for the same seats. [ID412]

Other students, while mentioning the concurrency issue, were also very concerned about dealing with the nature of group sales. For example:

> First of all, there could be an issue of finding group seating. Finding n best available seats will not necessarily do, if they have to be n best available seats together. In such a situation, each seat should be labeled with how many seats there are available in front, behind, and to the left and right of it. [ID425]

> scalpers and other ticket selling agencies will buy tickets and sell them at an increased price; with no limit on n, the number of tickets the caller is purchasing, one caller could buy every ticket to the concert. This problem is easily fixed by putting an upper limit on n of 8-12 tickets (large groups can call a special hotline and speak to an operator to purchase more). [ID430]

Even the notion of "best seat" received further attention. For one student, the double-selling problem was handled by dividing up the seats among sellers. Most of algorithmic attention was then focused on the following problem.

> First of all, there would no longer be a first come first serve basis and problems would arouse over who actually occupies the good seats first. If it's only one seller, she would be able to take one customer at a time. Two or more sellers would make it hard to decide which seller's customer actually receive the seats first. [ID431]

4.3.2 Identifying the Main Synchronization Problem

The degree to which students identified the problem varied. Most gave a fairly standard "sellers could mark the seats unavailable at the same time." [ID412] or "there could be double booking" [ID106]. Remarks that this scenario would be very unlikely were not uncommon, however. Some students identified computers or technology specifically as being the source of this problem:

> One computer may be operating slower than another, causing the seats that one seller saw to be taken by another seller. [ID406]

Others may not have addressed technology specifically, but did identify the key concept of time:

> One major issue is when, and how long it takes to mark a seat unavailable. [ID410]

A few students gave problem descriptions that went into quite a bit more detail and hint at the kind of analysis that will eventually be needed in constructing a full solution to the problem, including some recognition of the interleaving problem:

> The first, most obvious problem is that of overlap. If all sellers are working at the same time,

then the system might display to seller A that certain seats are open when, in fact, they have already been reserved by seller B. Thus A will have to find different seats, which might have, in the intervening time, been reserved by seller C. [ID 417]

ID417 elaborated:

> Reservation information from each of the computers would have to cross-pollinate to each of the other computers as soon as the seats changed status at all, to either of the three states. This introduces the problem of crossed signals. If seller A and seller B both book seats 145 - 160 at the exact same time, or within milliseconds of one another, the instructions for reserving those seats on each of the other computers would cross mid-stream, introducing a problematic double-booking, or even worse, no booking at all. [ID417]

4.3.3 Identifying Other Problems

Along with the main problem, students noticed other issues.

Payment and canceling of orders were two big issues:

> Another problem would be if the seats are marked unavailable before they are sold, the customer can change their minds before payment and possibly hinder the sale of those seats to another customer who might have wanted them at the same time. [ID420]

> ... 2. How will payment information be kept and how can customer information be shared between sellers? 3. If a customer cancels an order, how is that information transmitted to the other seller within a reasonable amount of time? 4. If the customer does not pay for the tickets at will call, what happens to it? ... [ID 422]

> ... Filters would be helpful for the credit card and debit card numbers to avoid erroneous values. Throw in a few fields, and again only have one seller able to work with it at any given time, avoiding the redudency problem. Unfortunately, if seats are not allowed to be made available through the database, a customer could call, reserve seats, and then hang up, resulting in empty seats reserved. ... [ID410]

> Further more, at the moment of receving the payments for those tickets, problems might come up such as; miss-communication between the sellers,and charging the customer double instead of one time. [ID419]

Will-call suggested problems for some students:

> ... Another potential problem arises when customers decide to place their tickets at will call. It is possible for people to have the same name, so more information such as phone number or address would need to be collected by the sellers in order to avoid confusion at the door. [ID105]

Reliability problems also were mentioned:

> The computers may malfunction and the seller may not be able to key in the requested seats. [ID406]

As noted above when discussing the algorithmic goals, group sales were noted as a particular problem.

> Another problem that could happen is the sellers not selling the seats efficiently to maximize the amount of people that can attend the event. Many people buy group tickets to events and vendors sell them seats adjacent to another. Sometimes there will be a few seats not sold next to those group of seats. Other larger groups won't be able purchase seats close to another due to the lack of 'group' seats. [ID412]

4.3.4 Centralized Solutions

The three variants of centralized solutions we saw had significant distinguishing characteristics. C1 solutions relied on implicit communication between sellers and a central system which makes the reservation or selection on behalf of the seller. A common characteristic of these implicit communications was that they be "fast".

> These problems might be avoided by having a computer system that automatically (to the second) inputs the seat reservation for that customer. [ID438]

Some answers were less specific about the solution, but still gave evidence of a centralized solution with implicit communication:

> The program would have to temporarily mark seats that are being looked at during a transaction as unavailable so that vendors couldn't sell seats simultaneously. [ID313]

Others were significantly more specific about the solution, even imposing additional restrictions, yet still leaving doubt as to the student's true understanding of the concurrency issue in question. Here we see evidence of an attempt to move the potential point of concurrent access in an expressed solution:

> The only real way to avoid this is and still have multiple sellers is to run the booking on a computer network, with a master list of the seats available. The process would then go something like this: a caller calls in and requests n number of seats. The master list can be ordered in such a way that it fills the seats front to back, left to right, and when a seller requests n seats, it gives the next n seats on the list. Seat orders that have been cancelled are inserted at the top of the available list, in order of precedence. The seller can reserve the seats, ask if the seats are acceptable to the customer, and if so, proceed with the transaction... This would avoid double booking because during the time the seller is offering the seats to the customer, they are withheld from the list, and the other sellers drawing from the list would not have access to those seats. [ID130]

C2 solutions differed from C1 in that they used explicit communication with a centralized resource which made seat assignments (sometimes identified as a database).

> In order to avoid this, we could set up the database so that only one person could access to the database at a time. This would slow sales significantly, but is the safest setup. [ID440]

Another variant of this explicit communication involved a particular ordering required to ensure a safe process including lock-stepping or pipelining the process.

> These problems could possibly be avoided if instead of multiple people selling tickets and being involved in every step of the process, the selling process was divided between two employees. This way, while the second employee was taking care of the payment of the first caller, the first employee could start to deal with the next sale, and then transfer the call to the payment employee. [ID120]

Another variant of pipelining doesn't require a computer solution for the concurrency issue at all:

> A possible solution to this problem would to have a stagger-start approach when more than one worker is on the phone. Example when the first caller calls, worker A picks up the phone right away and starts to do their job. Then the second caller calls right after the first has called. Worker B then wait until the phone rings 3 times then worker B picks up and starts the process. [ID121]

C3 solutions (the most common) involved distributing resources in some way to avoid simultaneous access. The most common resource to be distributed was the seats to be sold. Some students remarked on potential issues with this approach:

> Perhaps if each vendor were responsible for a section of the concert hall, and finding the best seats within their section, this problem would be solved. But that solution also means that some vendors will fill up the "good" seats in their section faster, and the customer won't get the absolute best seats they could. Chances are good, however, that the customer wouldn't be aware that there were better seats available, and would rationalize that the concert filled up quickly. [ID303]

Others were more specific about the technique they would use to distribute resources – by assigning a seller to a particular type of seat. They then noted that this could simplify or uncomplicate things – pointing out a possible benefit.

> One way we could fix this problem would be to assign a section of seats to each seller. This way no seats are sold twice and it would be more organized. One seller would be in charge of one price and one section, so this makes the selling of seats faster and more efficient. [ID404]

> The sellers can organize themselves to sell specific seat sections. There can be an operator that finds out the general section that is desired, and forward the call to the seller of that section. [ID323]

A derivative of this solution of resource distribution was to place the entire resource under the control of one seller:

> To solve this we could only have one seller working to process the seats properly and without overlapping. [ID416]

4.3.5 *Decentralized Solutions*

Decentralized solutions are distinguished from centralized solutions by one key component – did sellers themselves make decisions and actual seat assignments? If so, then the solution is categorized as decentralized.

A common decentralized answer may reference a shared resource (e.g. a database or document) but the sellers make decisions individually based on that resource, rather than deferring to a centralized entity.

> To resolve this issue, there should be some communication between the sellers. Ideally, the sellers would mark the seats as unavailable on the same documents, so that there could never be any doubling. [ID101]

Other examples of non-specific communication among distributed sellers includes "know[ing]... dr[a]w[ing] off of the same information that was updated with each transaction "[ID304], "inform the other sellers of this by some form of communication." [ID437], "using a program that is constantly updated." [ID434]

Speed is a common theme with words like "instantaneously" [ID425], "instantly" [ID426] [ID402], "constantly" [ID434], "continuously" [ID410], and "real-time." [ID417]

Some were more specific in how communication needs to occur and in some cases seemed to realize that the problem may not be completely solved:

> A much easier way however would be to use a computer program that networks each seller. This way, every seller has access to every seat available. As soon as a booking is made, it will automatically register on ever seller's screen, and the chance of there being a double booking will be close to impossible. [ID323]

One student constrained the problem, but provided very explicit communication directives:

> So I would change the order of operations so that the 2 or more people booking seats would be required to check with each other while booking as to not book the same seats, in that way adding another step and aleviating the two problems. [ID409]

4.3.6 *Non-computing-oriented solutions*

Several solutions distinguished themselves for their distinctive non-computing and non-technological approach, though they could all be classified as either centralized or decentralized.

> We could mark the same seat map with different colored markers for each sell. [ID106]

This solution is a decentralized one, since one imagines individual sellers, each with their own colored marker, racing forward to a large map to mark off seats.

> This problem could be avoided by only allowing one vendor into the concert hall at a time. But this would be unreasonable if the concert hall were too large, or there were too many vendors working to reserve seats. Perhaps if each vendor were responsible for a section of the concert hall, and finding the best seats within their section, this problem would be solved. [ID303]

This response provided two solutions, the first decentralized and the second centralized by division of resource. Here we imagine sellers with cell phones dashing around the physical hall, placing markers of some sort on actual, physical seats. Of note is the discussion of scaling issues in the answer.

4.3.7 Common Errors in Solutions

The two most common errors in student solutions were in thinking that the problem could be solved with a faster system and in devising solutions that simply moved the point of concurrency to another point in the algorithm.

As we noted when describing the decentralized solutions, students suggested speed was needed to avoid many problems.

> To avoid this problem we could have very high "refresh" rates or have a way of reservering "n" seats as the process is still going through. [ID 423]

> Second, there must be instantaneous updates to the availability of seats. As soon as n seats are marked unavailable - even before the payment processing - the seats need to be marked unavailable. This way, another seller cannot try to reserve a seat that has already been "reserved". In addition, the system (and screen) would need to be refreshed every time a reservation is made. [ID 425]

> These problems might be avoided by having a computer system that automatically (to the second) inputs the seat reservation for that customer. [ID438]

Many solutions moved the point of concurrency. Here, the point is moved to a preview step:

> One more solution would be to have the computer show the n seats as unavailable as soon as any seller has them pulled up on their screen. With this system, only one seller could see these seats available at a time. If one seller (seller A) pulls up n seats for a customer, then another seller (Seller B) searches for the best seats, those seats that seller A was looking at would not be shown to seller B. [ID122]

> show which seats are being worked with by which seller, so that other sellers can't choose the same seats at the same time [ID420]

Another re-targeting of the point of concurrency was to a graphical interface:

> Creating a visual representation of the concert hall using a computer would alleviate this problem. Each seller working would mark a certain number of seats for their clients, letting other sellers know which seats are being purchased (potentially) and which seats are free for booking. [ID413]

One student realized his distributed, graphical interface only hid the concurrency problem and made a novel suggestion that appears to use the inherent randomness of human interaction to deal with the problem.

> Each seller would have their own computer and all of them would be connected, so once a seat is claimed, all of the other sellers will see it. If two sellers happen to click at the same time, a seperate window will have to open and they both will have to try again. [ID402]

One interpretation of this solution is that the separate window opens when a conflict is detected and forces the sellers to back off and retry, assuming that it is unlikely the sellers will try again at the same time. This of course leaves many issues unresolved including how the conflict is detected and whether or not other sellers could get in and reserve the seat desired by these two sellers.

5. DISCUSSION

5.1 Comparisons with Ben-David Kolikant's study

While our study is strongly based on Ben-David Kolikant's work [2] there are a number of significant differences. In this section we compare the two studies' methods, and results, and discuss points of similarity and hypothesize about differences in the results. We also address possible cultural complications which may have affected solutions.

5.1.1 Research methodology

Ben-David Kolikant's research methodology involves a series of assignments in a course on concurrency, the first of which is the cinema ticket assignment presented in Section 2. This question is posed in a detailed, pseudo-code format. The pseudo-code given for one machine is specifically designed to support the sale of one ticket at a time, which will pick the best ticket. The solution form requested included a hardware system specification, pseudo-code, and an explanation of the answer. The work and its quality were assessed and graded as part of the course.

Our research methodology differed significantly. Our data comes from college students at a range of United States institutions, at the beginning of the CS1 course. These students have a diverse background and, due to the multi-institutional nature of our data collection, have different expectations for the course in which they are enrolled. However, it is unlikely that they have any expectation of studying concurrency in this course, and have never been exposed

to the idea of concurrency as a computing topic. Due to students' relative lack of experience and due to a reasonable need not to scare students in their first week of CS1, our assignment takes the form of a more open-ended, descriptive question, written in English. As a consequence, our assignment statement is more vague and can be interpreted more widely and varyingly. Because no specific answer was requested, the number and content of student answers varied. A complicating factor is that, since this was a first assignment in a new course (and in many cases in a student's first experience in college) concerns about what the instructor wanted in the way of an answer and motivation as to why this question was being asked influenced the answers received. Students received credit for completing this assignment, but the quality of the answer was not assessed for points. Additionally, the degree to which the answers to this assignment were integrated (or even discussed) in the subsequent course was not controlled.

The differences in both the questions posed and the types of response solicited did not allow us to categorize responses in as precise a manner as Ben-David Kolikant. We began by trying to apply the five categories defined by Ben-David Kolikant (three centralized and two decentralized). However, we found that the imprecision of our English responses did not allow us to accurately determine if communication was an implicit or explicit event in decentralized solutions – so we only report on decentralized solutions as a whole. Additionally, since our problem and solution type allowed for multiple answers and answers to different questions, we also introduced categories to account for different problems being addressed, not addressing the problem at all, bad solutions to the problem, and ambiguous solutions.

5.1.2 Comparison of Results

Ben-David Kolikant's discussion focuses on the three subgoals that together contribute to the solutions she evaluated. These are described as "(a) the algorithmic goal of the system, (b) synchronization goals, and (c) reasonableness." [2] We make comparisons with our study following the structure, and then look at an overall comparison.

Algorithmic goal In the Ben-David Kolikant study, the problem posed was very well constrained, so the students all shared the same algorithmic goal: to sell the best ticket possible. Because of the unconstrained nature o f our questions, our subjects did not have the same consensus: they identified a number of different goals, some of which are provided in the results section. Two goals stand out in particular: students cared about making it easy to make seats available again after being given up, particularly as it related to the goal of providing the best seats to later customers; and students were concerned with handling group ticket sales, making sure seats were available for large groups to sit together even as the venue filled.

Synchronization Goal 33% of the solutions in the Ben-David Kolikant study were centralized. Our study shows an even higher number of centralized solutions (55%). This is consistent with Resnick [19], whose studies show that in daily experience managing a centralized solution to a problem is easier than managing entities in a decentralized way. Interestingly, despite the explosion of decentralized entities, particularly the Internet, since Resnick's studies, the students in this study were still more inclined to a centralized solution. Ben-David Kolikant notes that Resnick believed an increase in exposure to decentralized entities would increase the likelihood of using decentralization, but we find no evidence this has happened. (We did not ask students about their Internet experiences. Anecdotally, however, we believe that all the students in this study were experienced Internet users.) We will return to discuss the differences in breakdown of centralized and decentralized solutions between these two studies at the end of this section.

Consistent with the Ben-David Kolikant study, we found that students concentrated on sharing information across sellers rather than preventing interleaving access to the database, with only six students discussing the interleaving of operations. However, it may well be that the nature of our task was simply not as suggestive of database issues as the specific pseudo-code given in Ben-David Kolikant's problem statement, particularly given the lack of experience in our CS1 students.

While Ben-David Kolikant is able to show that students "exaggerate the grain of an atomic action," assuming checking and updating the database is atomic, we find that a description of even this level of granularity is present only in the most explicit and detailed of our solutions. Many of the responses did not make this level of description of granularity of interaction clear, leaving it ambiguous as to how well they really understood the issue at hand.

Reasonableness In her interviews with students, Ben-David Kolikant found that students sometimes solved a simpler problem than the one assigned (the "assume-we-have-a-ladder" phenomenon). Ben-David Kolikant refers to these solutions as ones that do not fulfill the goal of being "reasonable" solutions. We see this phenomenon in our student solutions as well. From the wording of solutions, it is sometimes clear that students propose C2 (constant rate) or C3 (division of resources) solutions as "easy" or "simple" answers.

Interestingly, some of our students showed an ability to reason about the quality of their solutions. This student recognized the limitations of his or her solution, without suggesting a better one:

> Perhaps if each vendor were responsible for a section of the concert hall, and finding the best seats within their section, this problem would be solved. But that solution also means that some vendors will fill up the "good" seats in their section faster, and the customer won't get the absolute best seats they could. Chances are good, however, that the customer wouldn't be aware that there were better seats available, and would rationalize that the concert filled up quickly. Also, someone will move in to any "better" seats that are left empty after the start of the concert. [ID303]

This student gave both an "assume-we-have-a-ladder" solution and alternatives (s)he believed to be better:

> So the obvious solution would be to fire one seller, and just have one working at a time. (Just kidding – kinda). But the thing to do would be to 'assign' each seller a section of the arena or wherever the concert is taking place. One seller would be in control of half of the seats, and the other of the other half. There would be no conflicting seats. Or they could just switch systems to general admission. One more solution would

be to have the computer show the n seats as unavailable as soon as any seller has them pulled up on their screen. With this system, only one seller could see these seats available at a time. If one seller (seller A) pulls up n seats for a customer, then another seller (Seller B) searches for the best seats, those seats that seller A was looking at would not be shown to seller B. Therefore this would be a first come first serve system and would be flawless and not one seat would be shown as available at more than one time. [ID 122]

Influence of Real World Experience Ben-David Kolikant notes the influence of real-world experience with servers, databases, and networks which can be used to share information and allow communication among entities. We see similar references in our answers. Additionally, given our subject pool, we see significant influence of the U.S. cultural phenomenon, Ticketmaster©, and probably specifically the ticketmaster.com website. On this website, one selects an event of interest (performer, venue, date and time), then provides the number of seats desired and the price level one is willing to pay. The site then presents the best seats for the event at the price level. Customers have two minutes to pay for the tickets before they are released. One can also search again for different seats.

Overall Comparison Our results differ significantly from Ben-David Kolikant in the percentage of centralized solutions. It is our belief that our larger percentage of centralized solutions stem from two related causes. First, 82% of our centralized solutions are C2 or C3 solutions. Only 33% of Ben-David Kolikant's centralized solutions were C2 or C3 solutions. As Ben-David Kolikant brings up in the discussion of reasonableness, students will often make simplifying assumptions, not because they are reasonable, but because it will allow them to solve the problem posed to them more easily. Given our more open-ended response type, we believe that it was more "reasonable" to students to at least pose a C2 or C3 solution to our problem. A second and related effect also relates to simplicity and reasonableness. Given our less-well-defined problem statement and reduced direct engagement in the classroom setting of the problem we posed, students were not necessarily prodded to consider or outline a more complex decentralized solution in the same way that Ben-David Kolikant's students might have been.

5.2 Beginner Preconceptions and Pedagogy

Focusing on the issue of student preconceptions, we observe that students certainly understood the cause of problems inherent in supporting multiple sellers and expressed the problem well. Many were also capable of suggesting reasonable solutions. However, there are clear boundaries to the commonsense knowledge students brought to this topic. Many students chose to "pass the buck" on concurrency, pushing the issue from buying a seat to reserving it. In addition, they seemed to believe that if the sellers have a fast enough reservation system, there is no race condition.

In both correct and incorrect preconceptions, these students appear to essentially enter our first class at the same level of intuition as they enter whichever course is their first experience with concurrency. Instructors in concurrency courses may find it useful to leverage the preconceptions by conducting an exercise like this and picking out a variety of answers for further discussion to explicitly address the common errors, in particular emphasizing the real-world nature of the problem, pointing out that the same problem will exist for reservations as exists for sales, and that race conditions will come up even on a "fast system".

The combined effect of this study and Ben-David Kolikant's study suggest that dealing successfully with concurrency is not completely natural – students recognize the problems, but propose solutions that may simplify the problem or not quite handle the race conditions involved. It may be useful as instructors in early courses to point out interleavings, atomic operations and possible race conditions as they come up, for example, in the use of GUI interfaces. While a solution does not need to be provided at the CS1/CS2 level, the introduction of the issue may help students grapple with it more successfully in an operating systems or concurrency course.

6. CONCLUSIONS AND FUTURE WORK

This paper reports on a modified replication of the cinema ticket problem by Ben-David Kolikant [2] to identify preconceptions students have regarding concurrency issues. In this study CS1 students from five institutions in the U.S. were asked in their first week of study to answer, in English, an open ended question about the difficulties of having more than one person selling tickets for the same concert. Answers were categorized as centralized and decentralized solutions using the scheme developed by Ben-David Kolikant [2].

We find that students are able to recognize the big issue of selling duplicate tickets and that many are able to suggest solutions to this problem. We find that students gave solutions of sufficient quality that we could use Ben-David Kolikant's categorization scheme for analyzing solutions of more advanced students with few modifications. Some students were able to describe both centralized and decentralized solutions to the problem. We found a greater occurrence of centralized solutions than reported by Ben-David Kolikant, which may be a factor of the less specified problem statement or by the reduced engagement of the students in the response.

While not relating directly to most introductory computer science classes, this study provides more evidence that beginners are neither blank slates nor empty of useful skills. They bring analytical skills into the first day of computer science and this skill can be leveraged. It suggests again that instructors can leverage the inherent problem-solving and analysis skills of students by selecting problems with a context they understand, and then refining these problems in ways that illuminate differences and difficulties between their reasoning and solutions and those most easily created on the computational models we use.

We see evidence, in the comparison with Ben-David Kolikant, that experience provides more sophistication, for example in the granularity of the described solution and in the use of distributed resources. Going further in this direction, it may be worth asking students after the first course to address this concurrency problem to see what new features appear in the solutions. It is also worth investigating other real-world tasks that demonstrate these boundary conditions – cases in which most students can give a good start but that also provide an entry into showing the need for a deeper sophistication. Tasks relating to typical introductory computer science topics would be particularly valuable.

7. ACKNOWLEDGMENTS

The authors thank Sally Fincher, Josh Tenenberg, and the National Science Foundation (through grant DUE-0243242) who provided us with workspace at the SIGCSE conferences in 2006 and 2007. We also thank Renee McCauley, Sara Miner More, Tammy VandeGrift, and Suzanne Westbrook for help with Fall 2006 data collection. We also thank others who have collected data for us in other commonsense computing projects.

8. REFERENCES

[1] M. Ben-Ari. Constructivism in computer science education. *Journal of Computers in Mathematics and Science Teaching*, 20(1):45–73, 2001.

[2] Y. Ben-David Kolikant. Gardeners and cinema tickets: High schools' preconceptions of concurrency. *Computer Science Education*, 11(3):221–245, 2001.

[3] Y. Ben-David Kolikant. Students' alternative standards for correctness. In *Proc. of the 1st Intl. Computing Education Research Workshop (ICER 2005)*, pages 37–43, Seattle, WA, October 2005.

[4] J. Bennedsen and M. Caspersen. An investigation of potential success factors for an introductory model-driven programming course. In *Proc. of the 1st Intl. Computing Education Research Workshop (ICER 2005)*, pages 155–164, 2005.

[5] J. Bonar and E. Soloway. Preprogramming knowledge: A major source of misconceptions in novice programmers. In E. Soloway and J. Spohrer, editors, *Studying the Novice Programmer*. Lawrence Erlbaum Associates, Hillsdale, NJ, 1989.

[6] J. D. Bransford, A. L. Brown, and R. R. Cocking, editors. *How People Learn: Brain, Mind, Experience, and School*. National Academy Press, Washington, DC, expanded edition, 2000.

[7] J. Bruner. *The process of education*. Harvard University Press, Cambridge, MA, 1960.

[8] T.-Y. Chen, G. Lewandowski, R. McCartney, K. Sanders, and B. Simon. What do beginning students know, and what can they do? In *ITICSE '06: Proceedings of the 11th annual SIGCSE conference on Innovation and technology in computer science education*, pages 329–329, 2006.

[9] T.-Y. Chen, G. Lewandowski, R. McCartney, K. Sanders, and B. Simon. Commonsense computing: using student sorting abilities to improve instruction. In *SIGCSE '07: Proceedings of the 38th SIGCSE technical symposium on Computer science education*, pages 276–280, 2007.

[10] M. Clancy. Misconceptions and attitudes that interfere with learning to program. In S. Fincher and M. Petre, editors, *Computer Science Education Research*. Taylor and Francis Group, London, 2004.

[11] Committee on Undergraduate Science Education. *Science Teaching Reconsidered: A Handbook*. National Academy Press, Washington, DC, 1997.

[12] G. E. Evans and M. G. Simkin. What best predicts computer proficiency? *Commun. ACM*, 32(11):1322–1327, 1989.

[13] J. P. Gibson and J. O'Kelly. Software engineering as a model of understanding for learning and problem solving. In *Proc. of the 1st Intl. Computing Education Research Workshop (ICER 2005)*, pages 87–97, 2005.

[14] D. Hammer. Student resources for learning introductory physics. *Physics Education Research, American J. Physics Supplement*, 68(7):S52–S59, 2000.

[15] R. Lister, E. Adams, S. Fitzgerald, W. Fone, J. Hamer, M. Lindholm, R. McCartney, J. Mostrom, K. Sanders, O. Seppala, B. Simon, and L. Thomas. A multi-national study of reading and tracing skills in novice programmers. *SIGCSE Bull.*, 36(4):119–150, December 2004.

[16] M. McCracken, V. Almstrum, D. Diaz, M. Guzdial, D. Hagan, Y. B.-D. Kolikant, C. Laxer, L. Thomas, I. Utting, and T. Wilusz. A multi-national, multi-institutional study of assessment of programming skills of first-year CS students. *SIGCSE Bull.*, 33(4):125–180, 2001.

[17] L. Miller. Natural language programming: Styles, strategies, and contrasts. *IBM Systems J.*, 20(2):184–215, 1981.

[18] L. Onorato and R. Schvaneveldt. Programmer/nonprogrammer differences in specifying procedures to people and computers. In E. Soloway and S. Iyengar, editors, *Empirical Studies of Programmers*, chapter 9, pages 128–137. Ablex Publishing, 1986.

[19] M. Resnick. *Turtles, termites, and traffic jams: Explorations in massively parallel microworlds*. MIT Press, Cambridge, MA, 1994.

[20] N. Rountree, J. Rountree, A. Robins, and R. Hannah. Interacting factors that predict success and failure in a cs1 course. *SIGCSE Bull.*, 36(4):101–104, 2004.

[21] A. Schwill. Fundamental ideas of computer science. *Bull. European Association for Theoretical Computer Science*, 53:274–295, 1994.

[22] Simon, S. Fincher, A. Robins, B. Baker, I. Box, Q. Cutts, M. de Raadt, P. Haden, J. Hamer, M. Hamilton, R. Lister, M. Petre, K. Sutton, D. Tolhurst, and J. Tutty. Predictors of success in a first programming course. In *Proc. of the 8th Australasian Computing Education Conference (ACE2006)*, pages 189–196, Hobart, Australia, 2006.

[23] B. Simon, T.-Y. Chen, G. Lewandowski, R. McCartney, and K. Sanders. Commonsense computing: what students know before we teach (episode 1: sorting). In *ICER '06: Proceedings of the 2006 international workshop on Computing education research*, pages 29–40, 2006.

[24] J. Smith, A. diSessa, and J. Roschelle. Misconceptions reconceived: A constructivist analysis of knowledge in transition. *Journal of the Learning Sciences*, 3(2):115–163, 1993.

[25] P. Ventura and B. Ramamurthy. Wanted: CS1 students. no experience required. *SIGCSE Bull.*, 36(1):240–244, 2004.

[26] S. Wiedenbeck. Factors affecting the success of non-majors in learning to program. In *Proc. of the 1st Intl. Computing Education Research Workshop (ICER 2005)*, pages 13–24, 2005.