Identifying Student Misconceptions of Programming

Lisa C. Kaczmarczyk
University of California, San Diego
Sixth College, MC0054
lisak@acm.org

Elizabeth R. Petrick
University of California, San Diego
Department of History, MC0104
erpetric@ucsd.edu

J. Philip East
University of Northern Iowa
Department of Computer Science
east@cs.uni.edu

Geoffrey L. Herman
University of Illinois at Urbana-Champaign
Dept. of Electrical and Computer Engineering
glherman@illinois.edu

ABSTRACT
Computing educators are often baffled by the misconceptions that their CS1 students hold. We need to understand these misconceptions more clearly in order to help students form correct conceptions. This paper describes one stage in the development of a concept inventory for Computing Fundamentals: investigation of student misconceptions in a series of core CS1 topics previously identified as both important and difficult. Formal interviews with students revealed four distinct themes, each containing many interesting misconceptions. Three of those misconceptions are detailed in this paper: two misconceptions about memory models, and data assignment when primitives are declared. Individual misconceptions are related, but vary widely, thus providing excellent material to use in the development of the CI. In addition, CS1 instructors are provided immediate usable material for helping their students understand some difficult introductory concepts.

Categories and Subject Descriptors
K.3.2 [**Computers and Education**]: Computer and Information Science Education – *Computer science education*.

General Terms
Human Factors.

Keywords
Curriculum, Concept Inventory, Programming, Misconceptions, Pedagogy, CS1.

1. INTRODUCTION
Most Computer Science Educators will recall times when they were completely baffled by how their students expressed their understanding of a critical topic. Clearly, understanding a student's inaccurate conceptualization is a necessary prerequisite for helping them move toward an accurate conceptualization. Unfortunately we cannot read minds and we cannot speak in depth with every struggling student. Thus, it would be very useful to have a reliable method of rapidly gauging the most important areas of conceptual difficulty, and to reveal in what form these difficulties manifest themselves.

A promising assessment approach is the use of a concept inventory (CI). The original CI was developed by physics educators (Hestenes, et al.) and addressed concepts of Newtonian Force as taught in introductory physics [10]. The authors had previously discovered that many students did not develop correct conceptions of critical topics. In response, the authors produced a multiple-choice examination that could be used by all physics instructors to determine whether their students appropriately understood the concepts of Newtonian Force. Perhaps their most critical contribution has been that instructors can use the inventory results to gain "on the ground" insight into not only the concepts their students are struggling with, but what specific misconceptions they hold. This information can be immediately leveraged to adjust instruction.

Prior to the project of which this paper is part, there was some discussion and preliminary attempts to develop a CI for discrete mathematics [1]. A digital logic CI is currently being developed and is nearly complete [9]. No other CIs have been fully developed for any area of introductory computing.

The results reported here are part of a multi-institutional project to develop concept inventories for three introductory computing topics: digital logic, programming fundamentals, and discrete structures. The process is as follows: previously, Delphi studies were conducted to identify concepts considered both important and a source of difficulty for students [7]. The next step involves interviewing students who have been instructed on each topic to identify their misconceptions. Results for digital logic have been published [8]; initial findings from interviews on programming fundamentals are reported here. As will be discussed in Section 6, these data and additional data currently being collected, will be used to develop, test, and validate the CI instrument.

2. BACKGROUND
Student misconceptions of programming and closely related topics have been studied for some time. Early studies, such as Mayer's work on mental models of the actions of programming statements [16], were followed by Bayman and Mayer examining misconceptions related to individual program statements in

SIGCSE'10, March 10–13, 2010, Milwaukee, Wisconsin, USA.

.

BASIC [2]. They found that many students had incorrect understandings or outright misconceptions of "much" of even very simple statements. Bonar and Soloway looked more generally at groups of statements to examine student understanding of programming [3]. They recognized that student "step-by-step natural language programming knowledge" interacted negatively with the programming knowledge of formal instruction. These studies touched upon misconceptions, but their primary focus was on larger mental models and theories of cognitive representation, or in some cases discoveries of misconceptions were not fully followed up pedagogically.

Spohrer and Soloway [20] examined the source of programming errors or bugs to see if they result from "misconceptions about the semantics of programming language constructs". They concluded that bugs are more likely to arise from student errors in reading and analyzing specifications and failures to see negative interactions between segments of code. Pea [18] looked beyond language constructs altogether, seeking insight into misconceptions. He found a "superbug"—students' tendency to expect computers to correctly interpret student actions and do the right thing. Confrey [5] examined both theoretical and empirical literature on misconceptions in mathematics, science, and programming and noted that the primary focus of the research was to avoid misconception development though changes in teaching. These studies focus primarily on misconceptions and present important results, however for a wider pedagogical use in CS1, they provide insufficient breadth and depth.

More recent work has often been narrowly focused. Ma, et al. [14] examined the correctness of mental models of assignment (of values and references) that are held by students at the end of a programming course and found a substantial number or erroneous models for even simple assignment. Fluery [6] and Madison and Gifford [15] focused specifically on parameters. Holland, et al. [11] presented misconceptions related to Objects and recommendations for addressing them, however their results were anecdotal and not supported with data.

A variety of other work on misconceptions exists, but from the perspective of providing data needed to develop a CI for CS1, they either replicate the problematic issues above, or else focus on other areas of program related conceptions (correctness and grading) rather than programming conceptions (e.g. Kolikant and Mussai [12] and Sanders and Thomas [19]). Thus there remains a need to investigate student misconceptions across a wide variety of CS1 topics. Conducting in-depth interviews with methodological rigor is one way to provide the broad cognitive understanding needed. In the following sections, we report a first set of results to rectify this situation.

3. INTERVIEW & ANALYSIS PROTOCOL

Eleven students took part in interviews conducted at the University of California, San Diego (UCSD) in spring 2009. Students were recruited from the undergraduate student population who were currently or recently enrolled in Computer Science or Computer Engineering introductory courses CSE8a or CSE11 (two versions of CS1). Participation was voluntary; subjects were recruited through announcements made in courses and via email to CSE lists. Students were compensated for participation in the project.

The primary purpose of the interviews was to reveal misconceptions held by students on an initial group of ten of the thirty-two concepts identified by the Delphi experts [7]. There were eighteen problems, covering the following concepts:

1. Memory Model, References, and Pointers (MMR)
2. Primitive and Reference Type Variables (PVR)
3. Control Flow (CF)
4. Iteration and Loops I (IT1)
5. Types (TYP)
6. Conditionals (COND)
7. Assignment Statements (AS)
8. Arrays I (AR1)
9. Iteration and Loops II (IT2)
10. Operator Precedence (OP)

A secondary purpose of the interviews was to validate the Delphi experts' conclusions that these concepts were indeed difficult.

The interviews were semi-structured and used a modified think-aloud protocol [4]. Choosing a language for code examples was an unavoidable necessity in spite of an overarching goal to develop as language neutral a CI as possible. We selected Java for three reasons. First, Java is currently one of the most widely used introductory programming languages. Second, our Delphi experts explicitly identified a subset of troublesome concepts as Object Oriented (OOP) based. Third, our student population had been taught in Java. It is important to note that not all concepts required that actual code be presented to students in order to reveal misconceptions. A full list of the problems is available from the authors and is expected to be published in a subsequent longer article. In addition, we will address the language dependence issue further in Sections 5 and 6.

There were multiple problems per concept, in order to guarantee that results did not depend on a single question. The majority of problems were covered in at least two distinct variations. Pilot interviews revealed that some concepts were closely related (e.g. Control Flow with other concepts). Thus, misconceptions emerged for some concepts within discussion of problems designed for another concept. Additional interviews and analysis on several of these "overlap" concepts are underway.

Each student was given a subset of the problems. Each interview lasted approximately one hour. With a few exceptions, every student was provided questions for all ten concepts. The exceptions occurred when students worked slowly and time limitations prevented full coverage. To avoid order bias, the problems were given in a semi-random order to each student. The caveat to the randomness of problem ordering is that each student was given one or two simple questions in the beginning to reduce anxiety and acclimate them to the interview process. Students were given the problems on paper, and provided scrap paper to work on if they desired. At no time did the interviewer reveal correct or expected answers to the problems. We collected audio and video recordings of the interviews, along with any written work the students produced. The audio tracks of the interview recordings were transcribed verbatim. Video was used as a back-up and as a visual resource if needed.

We analyzed transcripts and written work from ten of the interviews. Due to equipment failure, one interview was lost. The interviews were analyzed using the following steps of grounded theory and qualitative data analysis as described by Kvale [13], Strauss and Corbin [22], and Miles and Huberman [17]:

1. All of the survey responses were selected for coding in order to avoid bias in selection.
2. All of the survey responses were read and analyzed independently by three researchers: the first and second authors, and a researcher from one of the project's partner institutions (the fourth author).
   a. Each researcher developed codes, operational definitions, and themes grounded in the textual responses.
   b. The three researchers compared their coding and thematic decisions. When there were divergent findings, only those encodings were retained in which all researchers agreed. An inter-rater reliability rating of 96% was achieved.
3. Thirty-two codes with operational definitions were agreed upon. Twenty-five codes addressed the ten targeted concepts.
   a. The codes describe the misconceptions students held and were grouped according to the important and difficult concepts identified by the Delphi experts.
   b. Additional codes addressed other concepts from the full Delphi expert list have been specifically targeted in further interviews during summer and fall 2009.

4. RESULTS

Four themes emerged from the students' misconceptions (see Table 1). Themes 1 and 4 are highly language independent and cover general misunderstandings. Theme 2 involves a number of misconceptions all related to an inability to properly understand the process of while loop functioning. Though not truly language independent, this theme and its misconceptions are applicable across several commonly used contemporary and historic languages. Finally, Theme 3 is a basic lack of understanding of the most fundamental aspects of Object-Oriented Programming.

For the purpose of building a CI, misconceptions are the key data, as they are used to create authentic distracter questions on the instrument. In this paper, we focus on three of the six misconceptions within Theme 1: "Semantics to semantics," (MMR1), "Primitive no default," (PVR1) and "Uninstantiated memory allocation" (MMR4) (see Table 2). Both of the Delphi process concepts these misconceptions fall under (MMR, PVR) were highly ranked overall for importance and difficulty. Of the ten concepts addressed in this set of interviews (see Section 3), these two concepts were ranked highest by the Delphi experts for difficulty.

The first misconception, "Semantics to semantics," (MMR1) occurred when the student inappropriately assumed details about the relationship and operation of code samples, although such information was neither given nor implied. This misconception is language independent although every language will manifest the misconception differently. For example, when examining a list of Java variable definitions and declarations whose inter-relationships are unstated (see Appendix: Problem 1), Student2 explains: "And then have the names of the songs in here, which – but this would be stored in library, I'm assuming, or in the library class. I don't know how they're linked together exactly."
In another example, with a different problem (see Appendix: Problem 2b), Student3 makes incorrect assumptions about connections between variables to the extent that the student makes a mistake concerning the types of the variables. As a result, the student places Objects of different types in an array whose type matches none of them: "And so because there's two arrays, cheese and meats, uh, all those turkey and ham and roast beef are gonna be sorted into the meats array."

In a third example, Student8 completely and repeatedly ignores a variable, because it does not fit with her/his assumptions of how these variables must relate. In a lengthy discussion of the supposed relationships between the variables (see Appendix: Problem 2a), the sole reference (verbally or written) to "sauceType" was the following statement at the very start of the problem discussion: "Usually all the variables go to describing the Object, but I don't think it would describe a sauce."

It should not be surprising that students bring their own assumptions to problems. The Educational Psychology literature has solidly established this basic function of human cognition (e.g. [21]). However, we found it surprising where these assumptions led in terms of confusion between syntax and semantics. Even when an assumption based confusion led to clearly contradictory beliefs and conclusions, the students still could not recognize that their assumptions caused a problem. In one example, a student realized that the syntax did not fit his/her semantic assumptions and, instead of questioning those assumptions, he/she assumed that the syntax must be logically incorrect. Fortunately, this problematic cognitive behavior (for the purposes of learning programming) has also been discussed in the psychological literature and we should be able to draw upon that

Table 1. Themes Emerging From Student Misconceptions

T1: Students misunderstand the relationship between language elements and underlying memory usage.
T2: Students misunderstand the process of while loop operation.
T3: Students lack a basic understanding of the Object concept.
T4: Students cannot trace code linearly.

Table 2. Misconceptions About the Relationship Between Language Elements and Underlying Memory Usage

MMR1	Semantics to semantics	Student applies real-world semantic understanding to variable declarations.
MMR2	All Objects same size	Student thinks all Objects are allocated the same amount of memory regardless of definition and instantiation.
MMR3	Instantiated no memory allocation	Student thinks no memory is allocated for an instantiated Object.
MMR4	Uninstantiated memory allocation	Student thinks memory is allocated for an uninstantiated Object.
MMR5	Off by 1 array construction	Student thinks an array's construction goes from 0 to length, inclusive.
PVR1	Primitive no default	Student thinks primitive types have no default value.
PVR2	Primitives don't have memory	Student thinks primitives without a value have no memory allocated.

field's expertise and resources to customize solutions for computing education.

The second misconception, "Primitive no default," (PVR1) relates to lists of instance variables. This misconception is related to OOP and is a Java specific misconception. Student3 discusses two boolean variables without assigned values (see Appendix: Problem 2b) and states: "I don't think any value is being created for them because there's no assignment there. You know, it's just being declared as a variable." Student5 similarly discusses an integer which is not assigned a value (see Appendix: Problem 2a): "And then int is empty too and it's just creating space to later store an integer."

The third misconception, "Uninstantiated memory allocation," (MMR4) reveals itself when students think that memory is allocated for Objects which have been declared, but not instantiated. This misconception is also related to OOP. For example, Student5 explains how the computer handles memory for the uninstantiated Object "turkey" (see Appendix: Problem 2a): "it's just going to be this blank turkey because we're not setting it to be anything but we're creating like free space to the mater [*sic*] later on declare it."

In another example, involving a similar problem, Student2 discusses the memory allocated for the uninstantiated Object "artist" (see Appendix: Problem 1): "I'm thinking it goes to wherever artist is defined and looks at that class. And I feel like the class would set aside memory.

5. DISCUSSION

We found both unsurprising and surprising results in these interview data. The primary unsurprising, but welcome, result is that the misconceptions we uncovered confirmed the Delphi experts' choice of concepts as difficult for CS1 students.

Two surprising outcomes relate specifically to student misunderstandings. First, the breadth of misconceptions about memory models was unexpected. Memory models are very difficult, but we did not expect such a high number and variety of misconceptions. This finding has an important implication for pedagogy. There are likely to be a diversity of strategies to address memory model misconceptions, without any one quick or universally applicable fix. This challenge is particularly apparent regarding the misconception about students applying semantic assumptions to syntax (MMR1). It will take creative thinking by each instructor, as well as further research, in order to determine the most effective way to leverage these results.

The second surprising outcome relates to Theme 3, not otherwise discussed in this paper: a dearth of even basic conception of an Object. Some students had not formulated misconceptions about Objects, as they had no conceptions at all. During the interviews, they either froze, admitted with some embarrassment to having no idea what an Object was, even when prompted in several ways, or simply changed the subject. This extreme difficulty is being further investigated and results will be reported in a future publication. Meanwhile, one important implication of a lack of knowledge about Objects is that perhaps, within the context of particular student populations, instructors can take a step back and re-think how to introduce the concept of Objects, and focus explicitly on what they consider most critical about Objects in their particular incarnation of CS1.

6. FUTURE WORK

Our findings are representative of our participant population. However, many of the misconceptions we found are generally believed to be universal, but play out differently in different languages, and as such will need to be dealt with in the inventory. As we move forward in developing the inventory, we will further address issues of language dependence. We are currently evaluating options to address this concern. We will also need to address issues of OOP. OOP was an important category of concern to the Delphi experts, and thus must be included. However, we also want to make the inventory as flexible as possible, because at some point in time OOP may no longer be the dominant paradigm. The tension between these competing needs may be our most challenging task.

In following good grounded theory based protocols we have already used the data gathered so far to inform our next steps. First, we have completed a set of interviews conducted in Summer, 2009 that address the remaining Delphi expert concepts as well as the "overlap concepts". We also conducted interviews in the Fall, focusing on concepts which we determined needed additional investigation. Additional interviews are currently taking place at a partner institution to broaden the demographic of student subjects. Next, we will build and test the inventory. Pilot tests will take place at multiple institutions with diverse populations and multiple languages. Many of the original Delphi experts have expressed interest in taking part in initial field tests. Pilot inventory test results will provide feedback about how to improve the inventory questions so that the instrument will be useful to the broadest population and demographic possible.

7. CONCLUSION

We have presented initial results describing three important misconceptions held by CS1 students, along with four broad themes encompassing a larger group of misconceptions. The misconceptions detailed in this paper explore memory model representation and default value assignment of primitive values. These data provide immediately useful information for CS1 instructors to help them understand their students' misconceptions. Finally, these results will be merged with additional data being gathered, and used in the development and validation of a CI for Programming Fundamentals.

8. ACKNOWLEDGMENTS

This work was supported by the National Science Foundation under Grants DUE-0618589, DUE-0618598, DUE-0943318, and CAREER CCR-03047260. The opinions, findings, and conclusions do not necessarily reflect the views of the National Science Foundation or the authors' institutions.

9. APPENDIX

Problem 1. You are setting up a database of information about all the songs you own. Each song has certain information associated with it. Diagram (or use pseudo-code) how this information would be stored in memory:

```
Library library = new Library();
SongList[] songList = new SongList[3];
Genre genre;
Artist artist;
Title title;
Album album;
```

```
int trackNumber = 2;
int year = 1961;
int rating = 5;
```

Problem 2a. You are setting up a database of information about sandwich ingredients. There are a number of information items associated with your database. Diagram (or use pseudo-code) how this information would be stored in memory:

```
Cheese[] cheeses = new Cheese[4];
Meat[] meats = new Meat[2];
Turkey turkey;
Ham ham;
RoastBeef roastBeef;
boolean lettuce = true;
boolean tomato = true;
SauceType sauceType = new SauceType();
int numMeat;
int numCheese;
```

Problem 2b was identical to 2a except for the following two declarations:

```
boolean lettuce;
boolean tomato;
```

10. REFERENCES

[1] Almstrum, V. L., Henderson, P. B., Harvey, V., Heeren, C., Marion, W., Riedesel, C., Soh, L., and Tew, A. E. 2006. Concept inventories in computer science for the topic discrete mathematics. In *ACM SIGCSE Bulletin,* 38, 4 (Dec. 2006), 132-145.

[2] Bayman, P. and Mayer, R. E. 1983. A diagnosis of beginning programmers' misconceptions of BASIC programming statements. *Commun. ACM* 26, 9 (Sep. 1983), 677-679.

[3] Bonar, J. and Soloway, E. 1985. Preprogramming knowledge: a major source of misconceptions in novice programmers. *Hum.-Comput. Interact.* 1, 2 (Jun. 1985), 133-161.

[4] Bowen, C. W. 1994. Think-Aloud Methods in Chemistry Education. In *Journal of Chemical Education*. 71, 3 (Mar. 1994), 184-190.

[5] Confrey, J. 1990. A review of the research on student conceptions in mathematics, science, and programming. *Review of Research in Education,* 16, 3 (1990), 3-56.

[6] Fleury, A. E. 1991. Parameter passing: the rules the students construct. In *Proceedings of the Twenty-Second SIGCSE Technical Symposium on Computer Science Education* (San Antonio, Texas, United States, March 07 - 08, 1991).

[7] Goldman, K., Gross, P., Heeren, C., Herman, G., Kaczmarczyk, L., Loui, M. C. and Zilles, C. 2008. Identifying important and difficult concepts in introductory computing courses using a Delphi process. In *Proceedings of the Thirty-Ninth SIGCSE Technical Symposium on Computer Science Education* (Portland, OR, United States, March 12-15, 2008).

[8] Herman, G. L., Kaczmarczyk, L., Loui, M. C., and Zilles, C. 2008. Proof by incomplete enumeration and other logical misconceptions. In *Proceedings of the Fourth International Workshop on Computing Education Research* (Sydney, Australia, Sep. 06 - 07, 2008).

[9] Herman, G. L., Loui, M. C., and Zilles, C., Creating the Digital Logic Concept Inventory. In *Proceedings of the Forty-First ACM Technical Symposium on Computer Science Education*, Milwaukee, WI, March 10-13, 2010.

[10] Hestenes, D., Wells, M., and Swackhamer, G. 1992. Force concept inventory. *The Physics Teacher,* 30 (Mar. 1992), 141-158.

[11] Holland, S., Griffiths, R., and Woodman, M. 1997. Avoiding Object misconceptions. In *Proceedings of the Twenty-Eighth SIGCSE Technical Symposium on Computer Science Education* (San Jose, California, United States, February 27 - March 01, 1997).

[12] Kolikant, Y. B-D. and Mussai, M. 2008. "So my program doesn't run!" Definition, origins, and practical expressions of students' (mis)conceptions of correctness, *Computer Science Education*, 18, 2 (Jun. 2008), 135-151.

[13] Kvale, S. 1996. *Interviews: An Introduction to Qualitative Research Inquiry*. Sage Publications, Thousand Oaks, CA.

[14] Ma, L., Ferguson, J., Roper, M., and Wood, M. 2007. Investigating the viability of mental models held by novice programmers. In *Proceedings of the Thirty-Eighth SIGCSE Technical Symposium on Computer Science Education* (Covington, Kentucky, United States, March 07 - 11, 2007). SIGCSE '07.

[15] Madison, A. and Gifford, J. 2003. Modular programming: Novice misconceptions. *Journal of Research on Technology in Education*, 34, 3 (Spr. 2003), 217-229.

[16] Mayer, R. E. 1981. The Psychology of How Novices Learn Computer Programming. *ACM Comput. Surv.* 13, 1 (Mar. 1981), 121-141.

[17] Miles, M.B. and Huberman, A.M. 1994. *Qualitative Data Analysis: An Expanded Sourcebook, 2nd Edition.* Sage Publications, Thousand Oaks, CA.

[18] Pea, R. D. 1986. Language-independent conceptual "bugs" in novice programming. *Journal of Educational Computing Research*, 2, 1 (1986), 25-36.

[19] Sanders, K. and Thomas, L. 2007. Checklists for grading Object-oriented CS1 programs: concepts and misconceptions. In *Proceedings of the Twelfth Annual Conference on Innovation and Technology in Computer Science Education* (Dundee, Scotland, June 25 - 27, 2007).

[20] Spohrer, J. C. and Soloway, E. 1986. Alternatives to construct-based programming misconceptions. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (Boston, MA, United States, April 13 - 17, 1986).

[21] Stanovich, K. E. 2003. The Fundamental Computational Biases of Human Cognition: Heuristics That (Sometimes) Impair Decision Making and Problem Solving. In *The Psychology of Problem Solving*, J. E. Davidson and R. J. Sternberg, Eds. Cambridge University Press, Cambridge, UK, 291-342.

[22] Strauss, A. and Corbin, J. 1998. *Basics of Qualitative Research*. Sage Publications, Thousand Oaks, CA