A Project Summary

The goal of this project is to improve assessment of student learning in computer science. To this end, this work proposes to develop three *concept inventories* for introductory computer science subjects. Modeled after the successful Force Concept Inventory that was developed to assess student learning of Newtonian physics, the proposed concept inventories will test understanding of key computer science concepts in a manner that enables reliable comparisons between courses at different universities. With a standardized assessment tool, the computer science education community will be able to make meaningful comparisons of the effectiveness of different pedagogical approaches, greatly facilitating computer science education research.

In light of the fact that we are developing a tool to assess student learning, the approach we propose is a student-centric one. Concept inventories are designed to test student comprehension of difficult concepts by forcing a choice between the correct answer and distractors constructed from common student misconceptions. As students remain the primary source of information about which topics are difficult and about what the most common misconceptions are, we will engage students directly, as was done in the development of previous concept inventories. Through student introspections, discussions, interviews, and "think alouds" we will direct our development of questions for the concept inventories. These questions can then be refined and validated through peer review, qualitative, and psychometric analyses until the instrument is ready for large scale distribution. We will assemble an advisory panel — comprised of experienced concept inventory developers — to both advise and annually assess the progress of the project.

Specifically, we propose the following measurable outcomes:

1. **Concept Inventories:** We anticipate developing, testing, and refining concept inventories covering three core topics of computer science: discrete math, programming fundamentals, and digital logic design. These concept inventories will be disseminated to any interested parties.

2. **Insight on Common Misconceptions:** In the process of developing the concept inventories, we anticipate learning a lot about which concepts are most challenging and common misconceptions that we will contribute to the STEM education knowledge base.

Intellectual Merit: The proposed concept inventories in computer science build on the success of the Force Concept Inventory, which has played a significant role in the reform of undergraduate physics education. Through the development of rigorous tools for assessing student learning, the project will facilitate the development and assessment of scholarship of teaching and learning research in computer science, by enabling the comparison of pedagogical techniques within one university and across universities. Our multi-institution partnership will help ensure the validity of the proposed instruments by providing access to diverse student populations and different program objectives. Together, the P.I.s bring to the project a wealth of experience as award-winning teachers and an extensive track record of publishing in education-related forums.

Broader Impact: As the goal of the project is to develop tools for assessing student learning in computer science, the project has the potential to broadly impact computer science teaching and learning. In the development of the proposed concept inventories, we will explore the difficulties students have learning key concepts in computer science and the misconceptions they develop as a result; we will publish the results of these studies for use by other instructors. Also, as a necessary part of their development, a network of colleagues at other universities will be identified to review the concept inventories and test them. Once developed, we will widely disseminate the concept inventories through publications and workshops held at the ACM Special Interest Group on Computer Science Education (SIGCSE) and Frontiers in Education (FIE) conferences. The ability to rigorously compare student learning resulting from different pedagogical approaches will enable a productive dialog within the community and potentially speed the adoption of "best practice" pedagogies. As it has been shown that traditionally under-represented groups are more sensitive to the quality of instruction, improvements in pedagogy are likely to have a disproportionate benefit on such under-represented populations. Furthermore, a concept inventory with broad support will likely affect the computer science advanced placement (AP) exam in significant ways, thereby influencing computer science pedagogy at the high school level.

C Project Description

C.1 Motivation, Objectives, and Expected Impact

The *first* recommendation from a National Academy report entitled Evaluating and Improving Undergraduate Teaching in Science, Technology, Engineering, and Mathematics [23] is that:

> Teaching effectiveness should be judged by the quality and extent of student learning.

We believe that it is not by chance that the report's writers selected this recommendation to be the first, as having a rigorous and reliable tool for assessing student learning enables the systematic improvement of the education of our students. Specifically, a rigorous process for assessing student learning enables:

- **Evaluation of Pedagogical Approaches:** While there is a significant amount of ongoing innovation in pedagogy, it is difficult to evaluate. Is a new method better or just different? Without assessing student learning, it is impossible to tell. Furthermore, the assessment must be done in a standard way so that objective comparisons between pedagogies can be made.
- **Motivation for Improvement:** With no standardized means for assessing student learning, we can become complacent in our local maximas of pedagogy, but such complacency is difficult if we are confronted with how little our students are learning compared with students taught by our peers.
- **Faculty Evaluation:** Currently most departments rely primarily on student evaluations for evaluation of teaching. While potentially reliable for measuring success in achieving affective goals, student evaluations do not directly quantify student learning. Although there is a correlation, it is dangerous to evaluate faculty on one metric (student satisfaction), when it is another (student learning) that we truly desire to optimize.
- **Program Evaluation:** Finally, rigorous tools for demonstrating student achievement would be useful in the accreditation process. Such tools are especially timely in Computer Science because ABET is increasingly emphasizing outcomes assessment and requiring departments to institute continual process monitoring within the discipline.

In light of the potential, the absence of a standardized suite of assessments of student learning across the STEM disciplines is a testament to the difficulty of developing assessment tools. In particular, to address the above applications, an assessment tool must be:

- **Rigorous:** There should be broad agreement within the discipline that the assessment tool accurately reflects the current understanding of the discipline.
- **Reliable:** A student who understands a concept should, with high probability, be rated as knowledgeable, whereas a student that does not understand the concept should, with high probability, be rated as not knowledgeable.
- **Portable:** To facilitate comparisons between programs with substantially different curricula, the assessment tool must not rely on the detailed context in which the material is taught. For example, when testing understanding of programming concepts this must be done without the expectation of knowledge of a particular language.
- **Easy to Administer/Interpret:** To enable widespread use of an assessment tool, it should not require any specialized knowledge or equipment to administer, so that teachers who are not experts in education research can perform the assessment with a small investment of time.

While this is a formidable list of requirements, there is an assessment tool in the Physics community that meets all of them. The Force Concept Inventory (FCI) [35] was developed by David Hestenes and his collaborators to test student understanding of the key concepts of Newtonian mechanics. The FCI is now widely credited with stimulating reform in physics education [59].

The FCI consists of a series of multiple choice questions that present qualitative, concept-oriented problems. The FCI is rigorous, because it has been validated through peer review by other physics educators [34]. It is reliable, because each question's incorrect responses have been selected from common student misconceptions, making them attractive to non-Newtonian thinkers, but easily ignored by Newtonian thinkers; its reliability has been validated through student interviews to confirm that student scores represent student understanding. It is portable, because it tests only the fundamental concepts of force on which Newtonian physics is based and does so without requirement of any particular mathematical sophistication (*e.g.*, Calculus). Lastly, because it is a machine-scorable multiple choice exam, it is easy to administer. The FCI can be interpreted coarsely — focusing on the fraction of correct responses [34] — or at a fine grain — mapping incorrect student responses to common misconceptions that need to be addressed [35].

The quality and portability of the FCI has enabled it to make notable strides in two of the above applications. First, the FCI was used in a study that collected data from 62 distinct course offerings to compare traditional lecture-based pedagogies with those that emphasized interactive engagement; in his study, Hake found that interactive engagement methods achieved an average of a 46 point increase (on a 100 point scale) in FCI scores, while the traditional methods achieved an average of only a 23 point improvement from pre-test [31]. Second, it has significantly motivated many teachers to reform their classes, because (in the words of Richardson, et al. [70]):

> Upon administering the FCI at the end of the semester and reviewing the results, a teacher is frequently brought face-to-face with the ugly truth—his or her students do not understand some of the most basic key concepts. The realization can prompt the teacher to try different teaching methods, sometimes one involving more student participation.

The success of the FCI has led to an effort to replicate its impact in other disciplines. In recent years, the NSF sponsored Foundation Coalition has led an effort to produce concept inventories for other subjects in engineering (*e.g.*, chemistry [44, 67], dynamics [29, 30], fluid mechanics [57], heat transfer [38], material science [45, 46], signals and systems [89–93], statistics [2], strength of materials [70]), creating models for their development [20]. The instruments developed are now seeing use by researchers other than the ones who developed them [22, 39], which supports the expectation that the success of the FCI will be replicated in other disciplines. To date, however, there has been no work on concept inventories for computer science[1]. The goal of this project is to replicate in computer science the impact the FCI has had on physics education, through initiating the development of concept inventories for CS.

C.1.1 Proposed Research

The objective of the proposed research is to develop a number of concept inventories for three "core" computer science subjects [86]: discrete math, programming fundamentals, and digital logic design. Assessment instruments are needed in computer science, where only a small minority of educational research tries to assess the impact of proposed interventions on student learning. For example, a quick survey of the program for the 2005 ACM Technical Symposium on Computer Science Education (SIGCSE) — the largest and most influential forum for computer science-related education research — shows that only 13 of the over 100 regular papers attempted to measure impact on student learning, with fewer than half of those using rigorous approaches. Recently, members of the computer science education community have advocated for a maturing of computer science education [5] and for adaptation of concept inventories to computer science specifically [15].

Because of the virtues of the concept inventory approach (discussed above, but notably the ease with which it can be used by other researchers), we intend to replicate in computer science the success this approach has had in physics. Like physics, computer science has some concepts that are difficult to grasp (*e.g.*, the rigid nature of

[1] There has been some preliminary work on computer engineering concept inventories, but according to a Foundation Coalition coordinator for concept inventories these attempts did not advance far [18]. We have received copies of these works as they stand.

computers [68], variables [76], parameter passing [56], recursion [94], object-orientation [36], pointers and linked data structure manipulation [43]). The primary difference between computer science and most fields where concept inventories are being developed is that the concepts are abstract ones (*e.g.*, the notions of an invariant), for which the students have had little prior exposure [4, 97]. As a result, there may be less focus on eliminating incorrect pre-conceptions the students have upon entering the class and more focus upon misconceptions that develop during the course.

Our decision to focus on introductory subjects is not an arbitrary one. Partly, it is natural to start at the foundational subjects where we are not relying on the students to have prior domain knowledge; specifically, there are no dependencies on the content of prerequisite courses (a possible source of confounding variables). More importantly, however, is that the course objectives of these subjects are relatively stable, despite the continuing evolution of the field of computer science. In particular, discrete math and digital logic design are not fields whose key concepts are in flux. Furthermore, while introductory programming classes have evolved in the past decade to include more object-oriented programming ideas, most of the learning objectives remain unchanged [86]. Thus, we expect our concept inventories to retain their validity for a long time.

We describe our proposed approach for the concept inventory development in Section C.2. The two important features of this methodology are that 1) peer review, among the partner institutions as well as colleagues at other institutions, will play an important role, and 2) students will be involved in the development process because they provide the primary source for learning about student learning. The developed concept inventories will then be validated through peer review, qualitative, and psychometric analyses before being publicly distributed.

To perform this work, we have assembled a team of faculty that both frequently teach the subjects in question and together have a wealth of experience in education research. Furthermore, this team spans a diverse collection of institutions — large, public (University of Illinois at Urbana-Champaign), medium, private (Washington University in St. Louis) and small, undergraduate (Rose-Hulman Institute of Technology) — that provide a diversity of students and programs — as well as a population of hundreds of students per semester in the classes in question — that will help ensure the development of portable instruments. To further increase in the diversity of our student populations, we are partnering with Chicago State University, which is predominantly (87%) African American, and have approached Northeastern Illinois University, which has a large (38%) Hispanic population, as a source for additional student interviews and pilot tests. A letter of support from Chicago State University is included in the proposal's supporting documents. The geographic proximity of these schools enables the team to have periodic face-to-face meetings.

C.1.2 Intellectual Merit and Broader Impact

Intellectual Merit of the Research Activities: The proposed concept inventories in computer science build on the success of the Force Concept Inventory, which has played a significant role in the reform of undergraduate physics education. Through the development of rigorous tools for assessing student learning, the project will facilitate the development and assessment of scholarship of teaching and learning research in computer science by enabling the comparison of pedagogical techniques within one university and across universities. Our multi-institution partnership will help ensure the validity of the proposed instruments by providing access to diverse student populations and different program objectives. Together, the P.I.s bring to the project a wealth of experience as award-winning teachers and an extensive track record of publishing in education-related forums.

Broader Impact: As the goal of the project is to develop tools for assessing student learning in computer science, the project has the potential to broadly impact computer science teaching and learning. In the development of the proposed concept inventories, we will explore the difficulties students have learning key concepts in computer science and the misconceptions they develop as a result; we will publish the results of these studies for use by other instructors. Also, as a necessary part of their development, additional colleagues at other universities will be identified to review the concept inventories and test them. Once developed, we will widely disseminate the concept inventories through publications and workshops held at the ACM Technical Symposium on Computer Science Education (SIGCSE) and Frontiers in Education (FIE) conferences. The ability to rigorously compare student learning resulting from different pedagogical approaches will enable a productive dialog within the community and potentially speed the adoption of

"best practice" pedagogies. As it has been shown that traditionally under-represented groups are more sensitive to the quality of instruction, improvements in pedagogy are likely to have a disproportionate benefit on such under-represented populations [75, 88]. Furthermore, a concept inventory with broad support will likely affect the computer science advanced placement (AP) exam in significant ways, thereby influencing computer science pedagogy at the high school level.

C.2 Developing Concept Inventories

With the ongoing development of concept inventories in a number of (non-CS) disciplines, much has been learned about their development [10, 30, 37, 38, 44, 45, 57, 67, 70, 77, 82]. We plan to base our development of CS concept inventories using best practices identified in this previous work. In this section, we outline our proposed approach to concept inventory development.

For each of the proposed concept inventories, we plan to use the following four-step development process:

(1) identifying the important and difficult concepts,

(2) identifying common misconceptions for those challenging concepts,

(3) designing questions using those misconceptions, and

(4) validating the concept inventory through peer review, qualitative analysis, and psychometric analysis of trial runs.

Before we discuss the details of the proposed process, it is important to highlight two themes that are pervasive throughout the approach. First, ensuring quality necessitates peer review. In addition to the team we have assembled to perform this research, we plan to assemble a community of colleagues at other institutions to provide feedback on the concept inventories (primarily relating to steps 1 and 4). To identify interested faculty for this community beyond our current acquaintances, we are organizing a "Birds of a Feather" session at the SIGCSE conference in March, 2006 [98]. In addition, we will assemble a panel of experienced concept inventory developers to advise and evaluate our work (see Section C.4). Second, as the focus of this exercise is assessing student learning, it is fundamental to involve students in the process. We propose to employ student focus groups, student interviews, and student think-alouds to identify which concepts students find challenging and common misconceptions (step 2). Such focus groups will be selected for a diversity of backgrounds and achievement levels. Finally, while described here as a linear process, we expect the development to be an iterative process.

We discuss each step in more detail:

1. Identifying Important, Difficult Concepts: We will use the IEEE/ACM Computing Curricula 2001 [86] as an aid to set the scope of the proposed inventories. To reach a consensus for which concepts should be included in the inventory, we plan to use a Delphi process [1, 14, 17, 21, 49] with colleagues from other universities, as has been done in previous concept inventory development [30, 82, 84]. The Delphi process has several rounds that involve a large number of instructors who teach the subject/course for which the CI is being developed. The first round has these instructors identify the important concepts in the subject. From this feedback, a list of concepts is created by selecting those identified by multiple instructors. In the second round, participants are asked to rate these concepts for "importance" and "difficulty for students." By collecting input from a large, diverse body of instructors, we increase the likelihood that the instrument will meet the goals of instructors nationwide.

Furthermore, by involving a large population of instructors early in setting the goals for the concept inventory and keeping them informed about our progress, we hope to provide them with a sense of ownership in the concept inventories. Developers of other concept inventories have found this approach facilitates the dissemination of concept inventories and their early adoption [19].

We intend to complement the faculty perceptions of difficult concepts with student perceptions. In particular, we plan to survey students during the period of instruction, having them rate the difficulty of the concepts from the Delphi process.

2. Identifying Student Misconceptions: While a body of literature exists on student misconceptions regarding programming [7,9,36,56,68,69,76], it is not exhaustive, and there has been little work studying misconceptions in other areas of computer science. To isolate the source of student difficulties and identify common student misconceptions, we plan to use the student-centric approach used in many previous concept inventory development (*e.g.*, [38]). Because instructor perceptions of the source and nature of student difficulties may be faulty, it is more reliable to engage students directly. Using the qualitative methods described in Section C.2.2, we propose to use the following techniques:

- **Student Interviews and Think Alouds:** To draw out student misunderstandings, we propose having a dialog with students around broad questions of the form "How does a cache work?" or "When should recursion be employed?". The context of an interview provides the flexibility to explore the extent of a student's understanding. In addition, students can be presented (complex) problems to solve and asked to describe what they are thinking as they solve the problem. Recording the conversations enables further analysis after the fact.
- **Unmoderated Group Discussions:** Groups of (post-instruction) students can be assigned a set of concepts to discuss. For each concept that they believed they correctly understood, they would be charged with preparing a set of statements they would use to demonstrate that they understand the concept. For each concept they were not sure they understood, they would be charged with developing a set of questions, whose answers would help them understand the concept. Successive groups can be asked to come to a consensus about the correctness of the statements and the answers to the questions produced by other groups. In addition, group "think alouds" can be performed where students collaborate to solve problems. By videotaping these discussions, we can analyze the discussions to identify student misconceptions.
- **Interviewing Pre-Instruction Students:** Previous work in developing concept inventories have utilized interviews with pre-instruction students to identify pre-conceptions related to the course material that students bring into our courses. Because students rarely encounter discrete math and digital logic topics in their every day life (unlike Newtonian mechanics or heat transfer), we expect fewer incorrect pre-conceptions in these subjects. Nevertheless, we plan to do a small number of such interviews to verify this hypothesis. Furthermore, there may be some important incorrect pre-conceptions in a programming fundamentals course due to students who have taught themselves how to program.

3. Question Design: In the development of the concept inventories, we expect that creating the questions will be especially challenging for several reasons. First, answering a question correctly should require the understanding of concepts rather than recall of information, or rote execution of standard procedures. This attribute of concept inventory questions distinguishes them from conventional problems on homework assignments and on examinations. Second, a question should require not detailed calculations, but qualitative analysis. For example, the student might choose from among plotted graphs of functions with different shapes. Third, the questions should be portable: they should be clear and unambiguous to students in the relevant computer science course in any institution. Computer science concepts are traditionally taught in the context of particular computing systems, *e.g.*, a particular programming language or a particular instruction set architecture. Our questions should test concepts in a context-insensitive way. Fourth, the questions should avoid testing knowledge of conventions; whereas most concepts in the physical and engineering sciences relate to natural phenomena, most concepts in computer science relate to man-made artifacts whose behaviors are sometimes merely accepted conventions. For example, in an assignment statement, the variable name goes on the left side, and the expression goes on the right side; a flip-flop generally changes state on the rising edge of a clock signal rather than the falling edge; a tree is drawn with the root toward the top of the page. In Section C.3, we provide some examples of concept inventory questions in computer science.

4. Question Validation: In developing the concept inventory, or any test development situation, there are two issues of primary importance: test content validity and test reliability. Content validity addresses whether a test covers the subject matter appropriately with regards to the goals of the assessment; that is, does the test cover the correct knowledge? Test reliability addresses whether the test elicits consistent and reliable responses; that is, would students score equivalently if the exam were repeated or the questions replaced with equivalent alternatives. We plan to use three approaches to validate the quality of the developed questions:

- **Peer Review:** Ensure that colleagues at other institutions agree that the intended correct responses are correct and the intended incorrect responses are incorrect.
- **Think Alouds:** By having students explain why they selected the answers they do, we can ensure that the rate of false positives (students selecting correct answers despite a faulty understanding) and false negatives (students selecting incorrect responses despite a correct understanding) are low.
- **Psychometric Evaluation:** A final validation can be performed during large scale testing of the concept inventory, as has been done in prior work [2, 29, 60, 70, 77, 78, 81, 82]. Psychometric analysis can be performed to ensure that a concept inventory has internal consistency, that is each question is generally effective at discriminating those who score well on the inventory from those who score poorly. We discuss our proposed method of psychometric validation in Section C.2.3.

C.2.1 Human Subjects Methods

As we have taken a student-centric approach to our research, it is necessary to solicit student participants for our study. We desire our subject population to represent the diversity of potential computer science students. In this, we have two challenges: first, many computer science departments are predominantly White and Asian males, and, second, we have found empirically that "high achievement" students volunteer for the interviews at much higher frequencies than their "lower achieving" counterparts. This second challenge is noteworthy because we find that interviews with struggling students are more valuable, because it is from them that we learn why they are struggling.

As a result, we will actively recruit for the study those student populations that are otherwise under-represented. For the post-instruction interviews, we will solicit volunteers who took the course in question in the previous term, offering small monetary compensation to take part in the study as is standard practice. To ensure that our interviews span the achievement distribution, we will use the students' grade for the subject in question to guide student selection. Similarly, we anticipate interviewing all responding women and under-represented minorities to increase representation of these sub-groups. In spite of these efforts, we anticipate a lack of under-represented minorities for our studies, so we are making arrangements to interview students at other institutions (Chicago State University and Northeastern Illinois University) with larger populations of these under-represented minorities. We have already received institutional review board (IRB) permission at the University of Illinois for the initial work described in the Section C.3 involving students at UIUC.

C.2.2 Qualitative Analysis Methods

The goal of the pre-instruction and post-instruction interviews is to explore what preconceptions and misconceptions, if any, the students have about the course material. Because we cannot anticipate what the students will say, we will not use a statistical approach to this phase of the project. In particular, statistical studies are not well suited when the problem is ill-defined and there are no predefined response categories. Rather, an exploratory methodology that can follow up on virtually any response by the students is needed. A qualitative research interview is appropriate for this investigation because we will attempt to understand the world from the interviewee's point of view, prior to scientific explanations [47].

Given these research needs, these exploratory interviews will be conducted using Grounded Theory. Grounded Theory is a qualitative research methodology that is designed to investigate phenomena without pre-determined hypotheses [24]. In a Grounded Theory interview, the researcher listens to what people themselves tell about their world and learns about their views on their situation. Grounded Theory has the flexibility to pursue results in any direction. Questions are phrased to encourage descriptive responses, and additional questions are permitted when they follow up on topics brought up by the interviewee. The results are then used to inform other parts of the study, which may take place at the same time. In this project, the use of Grounded Theory means that the number and length of pre-course interviews conducted will depend upon the richness of the data obtained, and interviews will continue to be conducted as long as greater understanding is desired. Interviews will be recorded for post-interview analysis; audio only recordings are likely to be sufficient for the pre-instruction interviews, but video recordings will be used in the

post-instruction interviews where we will want to correlate the audio with the students progress in the "think aloud" problems. These recording can then be transcribed verbatim and analyzed for emergent themes using well-defined protocols [83]. Although the goals of the current project do not call for psychometric analysis of the interview data, it is worth noting that the transcribed interviews can be quantified and statistically analyzed in the future, if a need arises to do so [11].

In addition to the above interviews, we plan to survey the students during instruction, requesting them to rate their perceptions of the difficulty of the topics identified through the use of the Delphi method. In developing this survey questionnaire we will follow standard procedures for reducing bias in surveys [85].

C.2.3 Psychometric Analysis Methods

We will use standard psychometric methods [16, 51, 87] to ensure the statistical reliability and validity of the concept inventories that we develop. In this effort, we will be assisted by Professor Hua-Hua Chang, one of the senior personnel for this project, who is an expert in educational measurement.

To design each concept inventory, we will write items according to the instructional objectives, that is, the desired learning outcomes that students should achieve. Consequently, each concept inventory will be criterion-referenced, not norm-referenced. Thus we will compute criterion-based reliability and validity measures.

Formative Evaluation of Concept Inventories: During development of each concept inventory, we will conduct item analyses: we will calculate item discrimination, item reliability, and item validity indices. These indices will identify flawed items that we would rewrite. We will also determine whether different items that test the same concept are equally reliable and valid. We envision that eventually we would develop several items for the same concept in order to create different forms, or to have separate pre-tests and post-tests. Finally, we will check for item bias– for example, whether women consistently answer an item correctly more often than men, or whether students from under-represented minorities perform significantly better on an item than non-minority students, even though the true average levels of the two groups on the construct being measured are the same. Eventually, with enough students, we hope to apply item response theory to estimate parameters for a logistic model for each item so that more advanced psychometric analyses can be performed, such as item banking, test equating, and cognitive diagnosis.

Summative Evaluation of Concept Inventories: First, we will conduct a criterion-based reliability study, computing the appropriate stability and generalizability coefficients. Second, we will check the validity of each concept inventory in two ways. We will perform content validation with a panel of experts, who would evaluate the coverage of the concept inventory and its consistency with instructional objectives. We will perform a criterion-referenced validation by correlating the performance of students on each concept inventory with their scores on course examinations. We would expect moderate, but perhaps not perfect, correlation between concept inventories and exam scores, due to the limitations of each type of exam. In addition, correlating individual concept inventory and exam questions will allow us to assess to what degree the former let us predict student achievement on more open-ended and written answer style questions.

C.3 Initial Results

Based on a small amount of funding provided by the Provost's office at the University of Illinois, two of the PI's (Loui and Zilles) have already begun the development of a digital logic concept inventory. With a co-advised graduate student, we have conducted a series of student interviews, developed a small number of candidate concept inventory questions, and pilot tested them with a group of 28 students. We discuss some of our initial results here; this work is currently in submission for publication at the 2006 American Society of Engineering Education annual conference [50].

At UIUC, digital logic design is taught both in the Electrical and Computer Engineering (ECE) and Computer Science (CS) departments because of the large undergraduate enrollment — over 300 students per semester between the two departments — of each department. While slightly tailored for each population, the two courses (ECE290 and CS232) have similar learning objectives, which are well documented (shown in Figure 1), as required by ABET accreditation of Electrical Engineering programs. Given that these objectives correspond closely to those of similar

Representation of information
- Convert between decimal, binary, octal, and hexadecimal representations of integers
- Determine the number of errors that a code can detect or correct
- Understand two's complement representation of integers and determine whether overflow occurs in arithmetic operations
- Distinguish between a variety of decimal and alphanumeric codes

Design and analysis of combinational networks
- Understand the operation of discrete logic gates
- Analyze a combinational network using Boolean expressions
- Convert a verbal specification into a Boolean expression
- Understand basic properties of Boolean algebra: duality, complements, standard forms
- Apply Boolean algebra to prove identities and simplify expressions
- Use Karnaugh maps to find minimal sum-of-products and products-of-sums expressions
- Design combinational networks that use NAND, NOR, and XOR gates
- Design with MSI components such as encoders, decoders, multiplexers, adders, arithmetic-logic units, ROMs, and programmable logic arrays
- Calculate delays in ripple carry adders and combinational arrays

Design and analysis of sequential networks
- Understand the operation of latches; clocked, master-slave, and edge-triggered flip-flops; shift registers; and counters
- Plot and interpret timing diagrams
- Determine the functionality of sequential circuits from state diagrams and timing diagrams
- Translate sequential circuit specifications into state diagrams
- Design sequential circuit components (latches, flip-flops, registers, synchronous counters) using logic gates
- Synthesize general sequential circuits
- Understand tradeoffs in register and counter design

Figure 1: *Learning Objectives for a Digital Logic Design Course.*

offerings at a number of other institutions (*e.g.*, USC, NIU, Oregon State), we used them as the starting point for a list of important concepts in this small scale effort.

To identify which of these concepts were difficult for students, we interviewed eight students — six men, two women — over a period of seven weeks. The spacing of the interviews enabled us to ask questions of each student based on what we had learned from the previous ones. The interviews consisted of both open-ended questions, where we asked students to explain concepts to us, and "think alouds" where students solved problems, verbally explaining their thought process as they were doing it.

From these interviews we have uncovered a collection of observed student difficulties. These include: difficulty relating a state diagram representation of a finite state machine to representing the states in flip flops (discussed further below), an unwillingness to use Karnaugh maps to solve a problem when the problem is provided as a truth table (students seem to associate the solution technique with a particular problem representation), an inability to reason about whether a logic family is complete (also discussed below), difficulty composing medium-scale integration (MSI) components to create other MSI components, and a confusion between the behavior of latches and flip-flops.

From these student difficulties and others, we developed a collection of 11 questions, which were pilot tested as part of one end-of-the-semester review session held for the class. Our intention of this pilot testing was three-fold: 1) to validate that the misconceptions observed in our interviews were common to a larger population of students, 2) to test our formulations of these conceptual questions as multiple-choice questions, and 3) to identify which of a large collection of possible responses the students found to be compelling distractors. Twenty-eight students representing two recitation sections of the class (and a diversity of achievement levels) took the prototype inventory. Highlighting the need for iterative development, some of our questions proved to provide little discrimination between the students, but others reflected our prior observations.

Figure 2 shows a slightly modified version of one of the questions that performed well in the pilot test. In this question, we are testing a fundamental concept of representing information digitally, namely that states of a finite-state machine are represented as numbers and that the range of numbers that can be represented grows exponentially with the number of bits available (*i.e.*, 1 bit $\rightarrow$ 2 choices, 2 bits $\rightarrow$ 4 choices, 3 bits $\rightarrow$ 8 choices, etc.). Specifically, the question is testing whether students recognize that a single additional bit is sufficient to double the number of states

```
If we have a state machine that can be minimally represented in N bits, and I
double the number of states, how many bits are needed to represent the new
state machine.

a) N+2
b) 2N
c) N^2
d) 2^N
e) none of the above
```

Figure 2: *Candidate concept inventory question involving binary encoding of states.*

represented. Note that the correct answer to the question is `(e) none of the above` because the correct value (N+1) stands out from the compelling distractors ($2N$, N^2, and 2^N).

Of the 25 students who completed this question — three students left this question unanswered — slightly less than half (12) answered it correctly. The high number of incorrect responses is surprising because this question is trivial for knowledgeable computer scientists. While all but one of the distractors were selected, many students (7) selected answer `(b)` suggesting a perceived 1-to-1 relationship between the number of states and the number of bits needed to represent them. While further effort is necessary to validate this question, we confirmed in one interview that an incorrect response was not the result of misinterpreting the question.

Another example where our interview conclusions were validated by the pilot test, is shown in Figure 3. This question relates to completeness of logic families, testing the students' ability to perform reductions. For a logic family to be complete, it is necessary to be able to implement AND, OR, and NOT, from which any boolean expression can be implemented. From our interviews, we found that students had typically memorized that NAND, by itself, (answer `c`) was a complete logic family (selected by 24 of 26 respondents). By symmetry, students generally realize that NOR, by itself, (answer `d`) is also likely complete (also selected 24/26). Simple memorization and pattern matching is not sufficient, however, to reason about (a) and (b). The key to verifying that (a) is complete, is realizing that three of these gates can be used to implement a NAND gate, in two steps: 1) by setting the second input to 1, we can implement an inverter, and 2) by inverting the first input and the output we have a NAND gate. Only eight of the 26 students performed this reduction. Similarly, the same approach can be shown that (b) is not complete because implementing a NAND from AND or OR requires at least one inverter. Nevertheless, 10 of the 26 students indicated (incorrectly) that logic family (b) was complete; in one interview we observed a student who convinced themselves that NOT could be implemented from AND gates. Response `(e)` is a poor distractor (only two incorrect responses) that may be removed in the future.

C.4 Research Plan and Time Line

In this section, we discuss the specific outcomes we intend to achieve through this grant, if funded. In particular, we anticipate two specific outcomes: 1) the development, refinement, validation, and dissemination of concept inventories for three core CS subjects: programming fundamentals, discrete math, and digital logic design, and 2) publications on common student misconceptions identified in the process of developing the concept inventories. We describe each of these in detail, and outline a time line on which we expect these outcomes to be achieved.

Concept Inventories: Within the scope of the grant, we anticipate that we will be able to develop three concept inventories. As previously noted, we plan to focus on subjects in the "core" of computer science (*i.e.*, as defined by the ACM/IEEE-CS Computing Curriculum 2001 [86]) as these subjects are the most widely taught and their relative stability facilitates the development of a long-lived instrument. We currently anticipate developing concept inventories for discrete structures (DS1-5), programming fundamentals (PF1-4), and architecture and organization (AR1-6). The concept inventories will be developed using the process described in Section C.2. We will perform the psychometric evaluation of the concept inventories first through trials at the team institutions and then at our partner institutions. We will publish the results of our development and validation of the instruments and provide workshops about the

```
Which of the following are complete logic families (i.e., all possible
combinational logic circuits can be implemented using just these gates
and the constants 0 and 1).  There may be more than one right answer.
```

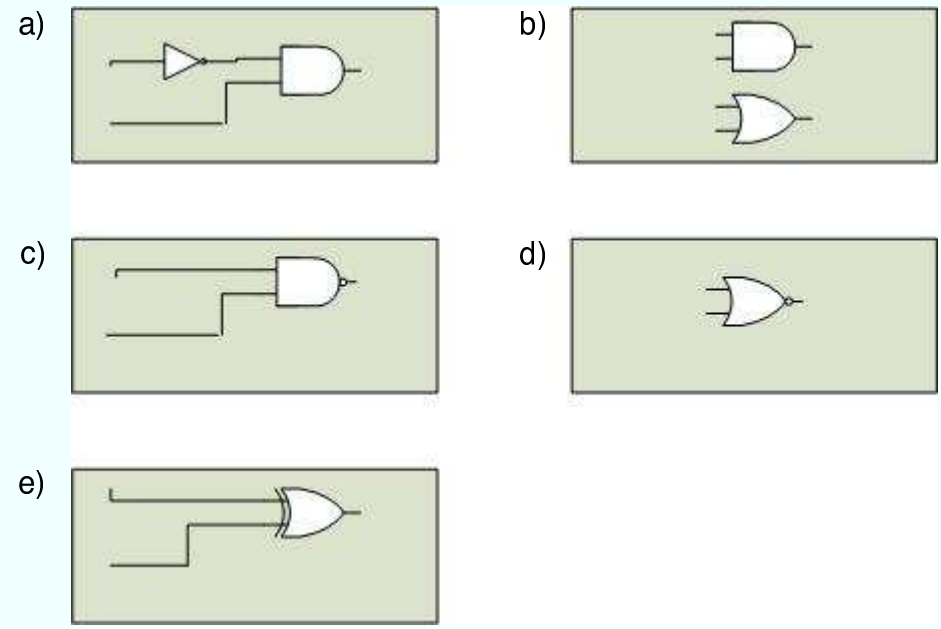

Figure 3: *Candidate concept inventory question involving binary encoding of states.*

concept inventories at SIGCSE and FIE conferences to facilitate their dissemination. We will make the concept inventories available to all interested parties and serve as a clearinghouse for (anonymized) results to facilitate comparisons between different pedagogical approaches.

Insight on Common Misconceptions: In the process of developing the concept inventories, we anticipate learning a lot about student learning challenges and common misconceptions. In addition to using this knowledge in the development of the concept inventories, we plan to distill this knowledge so that it can be published in its own right. In this way, we plan to contribute directly to the STEM education knowledge base, as has been done by previous concept inventory developers in other disciplines [32, 46, 80].

Because of the iterative nature of concept inventory development, we have requested a 3-year period, starting in August 2006, to do the proposed work. As previously mentioned, we are organizing a "Birds of a Feather" session at the SIGCSE conference (March, 2006), to identify additional partner institutions. The rest of the proposed time line is shown in Figure 4. Based on the partner institutions identified at SIGCSE and others, we intend to begin the Delphi process to settle on a list of included concepts before the term of the project, so that it will largely be in place before the Fall term.

At the beginning of the Fall term of years 1 and 2, we plan to interview incoming freshmen (before instruction) to identify their incoming knowledge and any preconceptions they have about the content covered by the proposed inventories. In parallel, we will begin surveying students who recently took and are currently enrolled in the courses in question, to identify which concepts they perceived as difficult; we expect to complete these student perceptions of difficulty in the first term. We will complement these student perceptions with faculty perceptions (part of the Delphi process) and with student interviews to probe student understanding. We expect these interviews to begin during the Fall term of the first year and to continue at least into the second year. Through these interviews we intend to identify the sources of student difficulty, which we can collect into publications (perhaps in the second year of the grant) and use to begin question development.

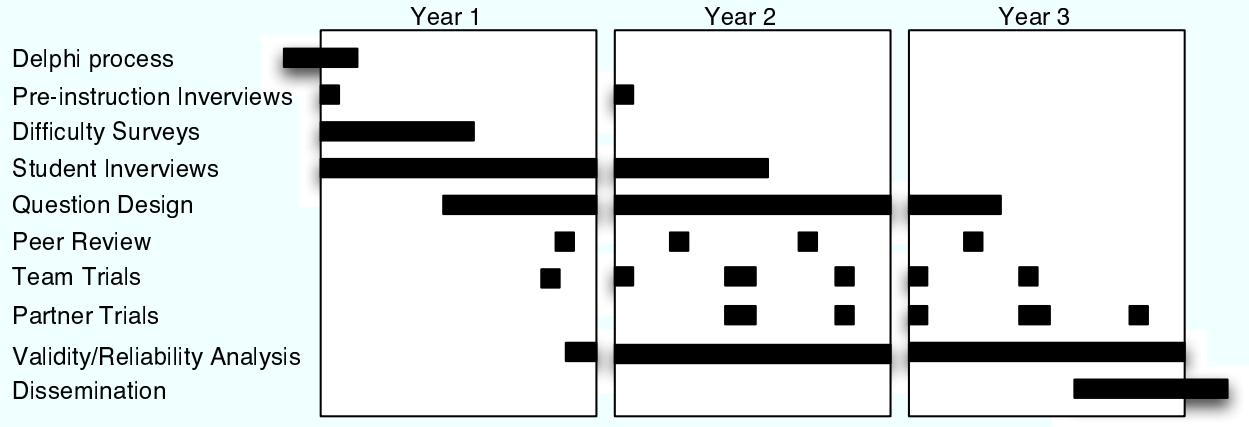

Figure 4: *Proposed project time line, starting August, 2006.*

Question development is expected to begin in the Spring of 2007 and question refinement is expected to last throughout most of the grant period as we receive feedback from peer, qualitative, and quantitative analysis. We intend to test draft concept inventories at the end of the Spring 2007 term. At this point, we will also begin peer analysis (to ensure that the intended correct answers appear unambiguously correct to experts) and qualitative analysis (to ensure that the questions are correctly interpreted by the students). After refining the questions over the summer, we plan to perform both a pre-test and post-test in the Fall 2007 term. Based on these results we will begin quantitative analysis in the Spring 2008 term. We expect the iterative cycle of testing, analysis, and refinement to continue through the third year of the grant, adding partner institutions to broaden the spectrum of student populations under test.

Advisory Panel: We will assemble an advisory panel to both advise and annually assess the progress of the project. We have been provided, by the project manager of the Foundation Coalition's concept inventory development, recommendations of experienced concept inventory developers who would make a good advisory panel [19]. We plan to convene the panel annually, in the middle of each year of funding. In addition to providing feedback to the project, they will be charged with issuing a report that will be included in progress reports to NSF.

C.5 Results from Prior NSF Funding and Qualifications of the PIs

Ken Goldman is an Associate Professor of Computer Science at Washington University in St. Louis. He has been teaching introductory computer science courses since 1995, when he began redesigning the introductory sequence. He pioneered a move to Java in 1997, making Washington University one of the first universities to recognize the value of Java as a language for teaching introductory computer science. The PI has a strong history of transferring research results to the classroom, creating both an introductory course for non-majors and an upper-level course on distributed applications. Goldman was named "Professor of the Year" in 2005 by the graduating senior class of the Washington University School of Engineering and Applied Science.

Dr. Kenneth J. Goldman, PI, *An Interactive Learning Environment for Introductory Computer Science*, (EIA-0305954), $514,996.00, 8/15/03-7/31/06.

This project involves the development of JPie, an interactive programming environment designed to make object-oriented software development accessible to a wider audience. JPie achieves this by providing live software, in which a program's behavior can be modified while the program runs. Programs are constructed by graphical manipulation of functional components [6, 25] so inexperienced programmers can achieve early success in a concepts-first curriculum [26, 27], without the steep learning curve that precedes development in a traditional textual language. JPie maintains detailed information about the software being developed and uses this to provide a more fluid software development process. For example, JPie's integrated thread-oriented debugger supports fine-grained inspection of the execution, run-time code modification, and on-the-fly exception handling. JPie transparently exposes the Java

programming model and provides access to compiled classes in the underlying language, enabling development of sophisticated object-oriented applications and establishing a pathway for programmers who want to transition into more traditional textual programming. JPie's functionality is provided within a user-friendly environment that streamlines the software development process. For example, its user interface builder supports property connections and automatic event handling.

Fine-grained *dynamic classes* [8, 28], whose signature and implementation can be modified at run time, were developed to support live programming in JPie. Dynamic classes provide full interoperability with compiled classes, including polymorphism, without modification of the Java virtual machine. Changes to dynamic classes, such as the declaration of instance variables and methods, as well as the modification of statements and expressions within methods, take immediate affect on existing instances of those classes. This makes JPie particularly suitable for experimentation in the educational setting. Recent JPie extensions support object-oriented access to relational databases [61] and live client/server development, including dynamic server interface changes, with support for both SOAP and CORBA [65, 66].

This project has involved 20 undergraduate and graduate students, including six minority and underrepresented students. JPie has been used for several semesters in our introductory computer science course for non-majors. In cooperation with ACM, Goldman organized and conducted a Java Engagement for Teacher Training (JETT) workshop for high school computer science teachers in the region. The workshop ran two full days in May 2004 and included hands-on sessions with JPie. Goldman also presented a JPie workshop for college educators at the ACM Special Interest Group on Computer Science Education conference in February 2005. Goldman will be an invited panelist at SIGCSE 2006, for a discussion on the effectiveness of tools for introductory programming courses.

Cinda Heeren is a lecturer and the visiting assistant director of diversity programs in the Computer Science department at the University of Illinois at Urbana-Champaign. She is the course coordinator and primary instructor for the department's entry-level theory of computation course, Discrete Mathematical Structures (CS173). Through support from Hewlett Packard and campus teaching innovation grants, she has reformed the class to reflect best practices in teaching and learning, as follows: 1) through the introduction of radio frequency polling devices, she has transformed her large (typically 150-200 students) classrooms into an interactive environment where participation is expected, 2) she has instituted optional (but attended by 90% of students) small, undergraduate-led active problem solving sections, 3) she has instituted "Proofs of the Day" providing students with a daily reasoning exercise whose topic is consistent with present course material, 4) she has effectively used electronic media (wikis and online chat) to create a forum where students can communicate with each other and TA (both synchronously and asynchronously). The concept inventory in Discrete Mathematics will serve as a critical evaluative tool for these and future course pedagogical innovations.

Dr. Heeren has a strong interest in computer science outreach, education, and diversity. To that end, she regularly presents popularized versions of fundamental lessons in computer science to students of all ages, from third graders through high school teachers, in a variety of venues. In addition to teaching middle school math at Countryside School (1997-2004), she has presented at numerous workshops including UIUC/ROE Novice Teacher Support Program, Expanding Your Horizons in Science and Mathematics conference for middle school girls, UIUC Engineering Open House, AIMS Drive-In Workshop for middle school teachers, MathManiaCS summer teacher training workshops, and the UIUC and College Board sponsored Java Engagement for Teacher Training (JETT) workshop.

Dr. Heeren served as director (Spring 2004-Fall 2005) of the Building Communities project, an NSF supported collaborative project between six central Illinois colleges and universities, whose focus is recruiting and retention of under-represented groups in computing. As part of that project she created and organized the first UIUC Regional Celebration of Women in computing in Spring, 2005, administered the Games for Girls and Technical Ambassadors Competitions, and facilitated high school recruiting visits by UIUC undergraduate women. She coordinated a panel discussion on recruiting and retention of women at the Grace Hopper Celebration of Women in Computing in October, 2004, and will present (with others) a workshop on hosting regional celebrations of women in computing at ACM SIGCSE Technical Symposium on Computer Science Education, in March, 2006. She is also the faculty advisor to the women in computer science organization at UIUC.

Dr. Heeren has had no prior NSF support.

Lisa C. Kaczmarczyk is an Assistant Professor in the Computer Science and Software Engineering Department at the Rose-Hulman Institute of Technology. Currently, Prof. Kaczmarczyk is teaching introductory computer science, both Honors and Standard, using innovative practices in pair programming and project development. Kaczmarczyk is developing a new course for the spring quarter that will introduce upper-division computer science students to highly influential writings in computer science. This course will require significant writing and a large project customized to the students' academic and career goals.

Prof. Kaczmarczyk's interdisciplinary research brings together the fields of computer science, psychology and education. Her research has used an artificial neural network to model the effect of different pedagogical delivery methods on student learning. After results showing that an Incremental Learning delivery method produced the best performance [42], Kaczmarczyk conducted a human subject study to further explore the results. This study not only compared performance between several popular delivery methods, but also compared differences in strategy development, cognitive development and affect. Kaczmarczyk's analysis was both statistical and qualitative, using well-established methodological protocols for each type of research. The results demonstrate that Incremental Learners develop the most effective study and test-taking strategies, have the best conceptual development, and have the most positive reactions to learning. Results show a significant improvement for Incremental Learners in all developmental areas, above that seen for other types of learners [41].

While working toward her Ph.D. at the University of Texas at Austin, Kaczmarczyk was an Assistant Instructor in the Computer Sciences department, teaching one class per semester for 6 years. She created Technical Writing for Computer Science Majors, using her industry experience and academic background to create a course that taught students to express themselves in written form as professional computer scientists. In this course, she conducted a study that investigated student perceptions of their learning. The study investigated pre- and post-instruction perceptions of skill mastery, self-efficacy, and motivation [40, 62], which was used to refine the course. Prior to her time at Rose-Hulman and UT-Austin, Kaczmarczyk spent 7 years as a full-time Instructor of Computer Science at Chemeketa Community College in Salem, Oregon. At Chemeketa, Kaczmarczyk designed the computer science transfer curriculum, coordinated course content with regional universities, and taught the majority of the curriculum. Kaczmarczyk is fluent in Spanish, has taught a computer literacy class in Spanish, and has worked extensively with a wide range of non-traditional students. She also taught as a visiting Instructor at the University of Oregon.

Prof. Kaczmarczyk is an active member of the ACM Special Interest Group on Computer Science Education (SIGCSE) and of the Cognitive Science Society. Prof. Kaczmarczyk has served on the organizing committee for the SIGCSE conference twice and reviews for the SIGCSE, ITiCSE, FIE and Cognitive Society conferences. The first three conferences are computer science and engineering conferences; the Cognitive Society conference is the primary international forum for presenting interdisciplinary research in human cognition. Kaczmarczyk won scholarships to attend the Grace Hopper Celebration of Women in Computing twice. Prof. Kaczmarczyk will be the keynote speaker at the Indiana Women in Computing conference in February 2006.

Prof. Kaczmarczyk has had no prior NSF support.

Michael C. Loui is Professor of Electrical and Computer Engineering and University Distinguished Teacher/Scholar at the University of Illinois at Urbana-Champaign. He will contribute to the discrete math inventory as an authority on the theory of computing [3] and will be teaching introductory digital logic design (ECE290) for the seventh time this Spring. He was an associate dean of the Graduate College at Illinois from 1996 to 2000. In 1995, he won the campus's Luckman Undergraduate Distinguished Teaching Award. In 2006, he was elected Fellow of the IEEE. He currently serves on the editorial boards of four scholarly journals, including *College Teaching* and *Teaching Ethics*.

In 2003, Professor Loui was named a Carnegie Scholar by the Carnegie Foundation for the Advancement of Teaching, to contribute to an emerging scholarship of teaching and learning. For his Carnegie project, he investigated how engineering students develop professional identities and how ethics instruction can affect this development. He found that after completing an engineering ethics course, some students articulate a capacious notion of professional responsibility that encompasses stewardship for society and the environment [54].

Professor Loui has collaborated with both undergraduate and graduate students in educational research projects, using both qualitative and quantitative methods. With graduate student Ryan Chmiel, he showed that in an assembly language programming course, students who completed debugging exercises spent significantly less time testing and debugging their programs [12, 13]. With undergraduate Golnaz Hashemian, Professor Loui created an interview protocol with multiple variation scenarios. In each scenario, the interviewed student was placed in the position of an engineer who faces a difficult ethical problem. Analysis of the interviews showed that compared with students who have not studied professional ethics, students who completed an engineering ethics course express greater confidence about taking action even when they have no assigned responsibility for a problem; they understand that professional responsibilities extend beyond completing assigned tasks conscientiously [33]. With graduate student I-Ju Liao, Professor Loui collected examination data in two large, required computer engineering courses to determine whether women earn lower scores than men. They found no statistically significant differences, even after controlling for ACT-Math scores [48].

J. H. Smith, W. D. Lawson, **M. C. Loui**, S. P. Nichols, P. E. Ulmer, V. Weil, PIs, *National Institute for Engineering Ethics Video Project: A Sequel to Gilbane Gold* (SES-0138309), $183,577, March 1, 2002 to February 28, 2005.

We developed a new video *Incident at Morales*, which dramatizes a fictional but realistic case study in engineering ethics [55]. The new video is directed to a broad audience, including engineering students, practicing engineers, and others who work with engineers. The video emphasizes everyday concerns rather than whistle-blowing situations, and it shows engineering in an international context. The video shows positive and negative role models of engineers who strive to reconcile conflicting ethical, technical, and economic constraints. We wrote a detailed study guide that accompanies the video, and we sent one free copy of the video and study guide to the dean of each engineering school in the United States. We showed the video at a conference on Ethics and Social Responsibility in Engineering and Technology, and at meetings of the Association for Practical and Professional Ethics and the National Academy of Engineering. In 2003, we demonstrated the pedagogical use of the video with two cooperative learning techniques in a special session at the Frontiers in Education Conference. We assessed the educational effectiveness of the video, using multiple instruments, with both student and professional audiences. A single showing of the video produced statistically significant positive changes in viewers' opinions about engineering practices, and statistically significant improvements in the sophistication of viewers' moral reasoning skills [52]. Both VHS and DVD versions of the video are now available for purchase through the National Institute for Engineering Ethics (www.niee.org). Several publications were supported by this grant [12, 13, 33, 48, 52–55, 64].

Craig Zilles is an Assistant Professor in the Department of Computer Science at the University of Illinois at Urbana-Champaign, who frequently teaches undergraduate computer architecture subjects. His most relevant prior NSF award is an NSF CAREER award. The award *CAREER: A Framework for Dynamic Self-Tuning of General Purpose Programs* (CCR-0347260, $410,000, Feb 1, 2004 - Jan. 1 2009), supports the development of an intelligent framework to control a run-time code optimization process, resulting in the following publications [71–74, 99, 100]. The educational component of this grant (now only in its second year) has already led to two education-oriented publications, as described below.

The first paper, entitled "SPIMbot: An Engaging, Problem-based Approach to Teaching Assembly Language Programming", describes a tool that the P.I. developed for teaching assembly language programming in the context of large enrollment classes. The tool allows students to program virtual robots (including performing I/O and handling interrupts) to perform certain tasks in a virtual environment. Because the tool can support multiple independently programmed robots simultaneously, it can be used for programming contests. The use of robots and programming contests is very motivating for students to learn what can otherwise be a very dry subject (assembly language). The paper [96] was presented at this year's SIGCSE conference and the software is available for download [95] for use by other institutions. It was the P.I.'s desire to quantify whether SPIMbot had a positive impact on student learning that has led to his interest in developing concept inventories.

The second paper discusses how some of the central concepts of computer science can be presented in the context of a computer architecture class. This paper [97] represents his initial step toward concept inventories in CS, in that it explored common misconceptions in learning ideas like abstraction layers, indirection, and pipelining. As

previously mentioned, unlike in Newtonian mechanics, our students do not have strongly held beliefs about much of the key concepts of computer science that need to be dislodged, but, interestingly, many do have preconceptions about technical terms they have developed from reading the popular computing literature (*e.g.*, slashdot).

Beyond actively publishing in the computer science education literature, the P.I. is widely recognized as being one of the best teachers in the Computer Science department at the University of Illinois at Urbana-Champaign (UIUC). He has frequently been elected to the university's "Incomplete Lists of Teachers Ranked as Excellent" [63], chairs the department's committee on Teaching Evaluation and Improvement, and is a member of the campus's community on the Scholarship of Teaching and Learning (SoTL).

References

[1] M. Adler and E. Ziglio, editors. *Gazing into the oracle: The Delphi Method and its application to social policy and public health*. Jessica Kingsley Publishers, London, 1996.

[2] K. Allen, A. Stone, T. R. Rhoads, and T. J. Murphy. The Statistics Concepts Inventory: Developing a Valid and Reliable Instrument. In *American Society of Engineering Education, Annual Conference*, June 2004.

[3] E. Allender, M. C. Loui, and K. W. Regan. Complexity theory. In A. Tucker, editor, *The Computer Science Handbook*, pages 5–1 to 5–30. CRC Press, Boca Raton, Fla., 2004.

[4] M. Ben-Ari. Constructivism in computer science education. In *the Technical Symposium on Computer Science Education (SIGCSE)*, pages 257–261, Mar 1998.

[5] M. Ben-Ari. The Concorde Doesnt Fly Anymore. In *the Technical Symposium on Computer Science Education (SIGCSE), keynote address*, Feb 2005.

[6] B. E. Birnbaum and K. J. Goldman. Achieving Flexibility in Direct-Manipulation Programming Environments by Relaxing the Edit-Time Grammar. In *VLHCC '05: Proceedings of the 2005 IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC'05)*, pages 259–266, Dallas, TX, 2005. IEEE Computer Society.

[7] B. Boulay. Some difficulties of learning to program. *Journal of Educational Computing Research*, 2(1):57–73, 1986.

[8] J. R. Brandt and K. J. Goldman. Run-time Modification of the Class Hierarchy in a Live Java Development Environment. Technical Report TR-2004-77, Washington University Department of Computer Science and Engineering, September 2004.

[9] P. Brna, B. du Boulay, and H. Pain. *Learning to Build and Comprehend Complex Information Structures: Prolog as a Case Study. Contemporary Studies in Cognitive Science and Technology.* Ablex, Stamford, CT, 1999.

[10] E. Brunsell and J. Marcks. Identifying A Baseline for Teachers Astronomy Content Knowledge. *Astronomy Education Review*, 3(2), 2005.

[11] M. Chi. Quantifying qualitative analyses of verbal data: A practical quide. *Journal of the Learning Sciences*, 6:271–315, 1997.

[12] R. Chmiel and M. C. Loui. An integrated approach to instruction in debugging computer programs. *IEEE Transactions on Education, to appear*. Preliminary version: In Proceedings of the Thirty-Third ASEE/IEEE Frontiers in Education Conference, Westminster, Colo., November 5-8, 2003. (pp. S4C-1 to S4C-6).

[13] R. Chmiel and M. C. Loui. Debugging: from novice to expert. In *the Thirty-Fifth ACM Technical Symposium on Computer Science Education (SIGCSE)*, pages 17–21, Mar 2004.

[14] M. J. Clayton. Delphi: A Technique to Harness Expert Opinion for Critical Decision-Making Task in Education. *Educational Psychology*, 17:373–386, 1997.

[15] J. Clement. A call for action (research): Applying science education research to computer science instruction. *Computer Science Education*, 14(4):343–364, 2004.

[16] L. Crocker and J. Algina. *Introduction to Classical and Modern Test Theory*. Holt, Rinehart, and Winston, New York, 1986.

[17] N. Dalkey and O. Helmer. An experimental application of the delphi method to the use of experts. *Management Science*, 9:458–467, 1963.

[18] D. Evans. Personal communication, March 2005.

[19] D. Evans. Personal communication, January 2006.

[20] D. Evans et al. Progress on Concept Inventory Assessment Tools. In *the Thirty-Third ASEE/IEEE Frontiers in Education*, Nov 2003.

[21] L. S. Fish and D. Busby. The delphi method. In D. Prenkl and S. Moon, editors, *Research methods in family therapy*, pages 469–482. Guilford Press., New York, 1996.

[22] B. D. Flinn and R. J. Caspe. Knowledge Retention Between Sequential Classes. In *the Thirty-Fifth ASEE/IEEE Frontiers in Education: poster presentation*, Oct 2005.

[23] M. A. Fox and E. Norman Hackerman. *Evaluating and Improving Undergraduate Teaching in Science, Technology, Engineering, and Mathematics*. National Academies Press, Washington D.C., 2003.

[24] B. Glaser and A. Strauss. *The Discovery of Grounded Theory; Strategies for qualitative research.* Aldine, Chicago, IL, 1967.

[25] K. J. Goldman. A demonstration of JPie: an environment for live software construction in Java. In *Companion of the 18th annual ACM SIGPLAN conference on Object-oriented programming, systems, languages, and applications*, pages 74–75. ACM Press, 2003.

[26] K. J. Goldman. A concepts-first introduction to computer science. In *Proceedings of the 35th SIGCSE technical symposium on Computer science education*, pages 432–436. ACM Press, 2004.

[27] K. J. Goldman. An Interactive Environment for Beginning Java Programmers. Science of Computer Programming, 2004.

[28] K. J. Goldman. Live Software Development with Dynamic Classes. Technical Report TR-2004-81, Washington University Department of Computer Science and Engineering, August 2004.

[29] G. L. Gray, F. Costanzo, D. Evans, P. Cornwell, B. Self, and J. L. Lane. The Dynamics Concept Inventory Assessment Test: A Progress Report and Some Results. In *American Society of Engineering Education, Annual Conference*, June 2005.

[30] G. L. Gray, D. Evans, P. Cornwell, F. Costanzo, and B. Self. Toward a Nationwide Dynamics Concept Inventory Assessment Test. In *American Society of Engineering Education, Annual Conference*, June 2003.

[31] R. Hake. Interactive-engagement vs traditional methods: A six-thousand-student survey of mechanics test data for introductory physics courses. *Am. J. Physics*, 66, 1998.

[32] I. A. Halloun and D. Hestenes. The Initial Knowledge State of College Physics Students. *Am. J. Physics*, 53(11), Nov 1985.

[33] G. Hashemian and M. C. Loui. Work-in-progress: Engineering courage: from "not my business" to positive responsibility. In *the Thirty-Fifth ASEE/IEEE Frontiers in Education*, pages S3D–17 to S3D–18, Oct 2005.

[34] D. Hestenes and I. Halloun. Interpreting the FCI. *The Physics Teacher*, 33:502–506, 1995.

[35] D. Hestenes, M. Wells, and G. Swackhamer. Force Concept Inventory. *The Physics Teacher*, 30, Mar 1992.

[36] S. Holland, R. Griffiths, and M. Woodman. Avoiding object misconceptions. In *the Technical Symposium on Computer Science Education (SIGCSE)*, pages 131 – 134, Mar 1997.

[37] B. Hufnagel. Development of the Astronomy Diagnostic Test by Beth Hufnagel . *Astronomy Education Review*, 1(1), 2002.

[38] A. Jacobi, J. Martin, J. Mitchell, and T. Newell. A Concept Inventory for Heat Transfer. In *the Thirty-Third ASEE/IEEE Frontiers in Education*, Nov 2003.

[39] W. Jordan, H. Cardenas, and C. B. O'Neal. Using a Materials Concept Inventory to Assess an Introductory Materials Class: Potential and Problems. In *American Society of Engineering Education, Annual Conference*, June 2005.

[40] L. Kaczmarczyk. A Technical Writing Class for Computer Science Majors: Measuring Student Perceptions of Learning. In *the Technical Symposium on Computer Science Education (SIGCSE), Panel*, Mar 2003.

[41] L. Kaczmarczyk, M. Last, and R. Miikkulainen. The Effect of Delivery Method on Strategy and Conceptual Development. Technical Report AI04-318, The University of Texas at Austin, Department of Computer Science, December 2004.

[42] L. Kaczmarczyk and R. Miikkulainen. The Acquisition of Intellectual Expertise: A Computational Model. In *Proceedings of the 26th Annual Conference of the Cognitive Science Society*, April 2004.

[43] K. N. King. The Case for Java as a First Language . In *the 35th Annual ACM Southeast Conference*, pages 124–131, April 1997.

[44] S. Krause, J. Birk, R. Bauer, B. Jenkins, and M. J. Pavelich. Development, Testing, and Application of a Chemistry Concept Inventory. In *the Thirty-Fourth ASEE/IEEE Frontiers in Education*, Oct 2004.

[45] S. Krause, J. C. Decker, and R. Griffin. Using a Materials Concept Inventory to Assess Conceptual Gain in Introductory Materials Engineering Courses. In *the Thirty-Third ASEE/IEEE Frontiers in Education*, Nov 2003.

[46] S. Krause, A. Tasooji, and R. Griffin. Origins of Misconceptions in a Materials Concept Inventory From Student Focus Groups. In *American Society of Engineering Education, Annual Conference*, June 2004.

[47] S. Kvale. *Interviews: An Introduction to Qualitative Research Inquiry*. Sage, CA, 1996.

[48] I. Liao and M. C. Loui. Work-in-progress: Do women score lower than men on computer engineering exams? In *the Thirty-Fifth ASEE/IEEE Frontiers in Education*, pages T3D–7 to T3D–8, Oct 2005.

[49] H. Linstone and M. Turoff. *The Delphi Method: Techniques and Applications*. Addison-Wesley, Reading, MA, 1975.

[50] J. T. Longino, M. C. Loui, and C. Zilles. Student Misconceptions in an Introductory Digital Logic Design Course. In submission to *the American Society of Engineering Education, Annual Conference*, June 2006.

[51] F. Lord. *Applications of item response theory to practical testing problems*. Lawrence Erlbaum Associates, Hillsdale, NJ, 1980.

[52] M. C. Loui. Assessment of an engineering ethics video: Incident at morales. *Journal of Engineering Education*, to appear. Preliminary version: Proceedings of the Thirty-Fifth ASEE/IEEE Frontiers in Education Conference, Indianapolis, Ind., October 1922, 2005, pp. S3D-19 to S3D-20.

[53] M. C. Loui. Educational technologies and the teaching of ethics in science and engineering. *Science and Engineering Ethics*, 11(3):435–446, Jul 2005.

[54] M. C. Loui. Ethics and the development of professional identities of engineering students. *Journal of Engineering Education*, 94(4):383–390, October 2005. Preliminary version: Proceedings of the Thirty-Fourth ASEE/IEEE Frontiers in Education Conference, Savannah, Ga., October 2023, 2004, pp. T2E-11 to T2E-12.

[55] M. C. Loui, E. W. LeFevre, S. P. Nichols, C. M. Skooglund, J. H. Smith, F. Suppe, P. E. Ulmer, and V. Weil. Incident at Morales: an engineering ethics video. In *the Thirty-Third ASEE/IEEE Frontiers in Education*, pages S1H–1 to S1H–2, Nov 2003.

[56] S. K. Madison. *A Study of College Students' Construct of Parameter Passing: Implications for Instruction*. PhD thesis, Department of Computer Science, The University of Wisconsin - Milwaukee, 1995.

[57] J. Martin, J. Mitchell, and T. Newell. Development of a Concept Inventory for Fluid Mechanics. In *the Thirty-Third ASEE/IEEE Frontiers in Education*, Nov 2003.

[58] J. Martin, J. Mitchell, and F. Pfefferkorn. Work in Progress - Concept Inventories for the Thermal Stem of Mechanical Engineering. In *the Thirty-Fifth ASEE/IEEE Frontiers in Education*, Oct 2005.

[59] J. P. Mestre. Facts and myths about pedagogies of engagement in science learning. *Peer Review*, 7(2):24–27, Winter 2005.

[60] R. L. Miller, R. A. Streveler, M. A. Nelson, M. R. Geist, and B. M. Olds. Concept Inventories Meet Cognitive Psychology: Using Beta Testing as a Mechanism for Identifying Engineering Student Misconceptions . In *American Society of Engineering Education, Annual Conference*, June 2005.

[61] A. H. Mitz. The Design and Implementation of Database-Access Middleware for Live Object-Oriented Programming, 2004. Masters Thesis, Washington University in St. Louis, Department of Computer Science and Engineering, TR-2004-21.

[62] L. K. (Moderator). Incorporating Writing into the CS Curriculum. In *the Technical Symposium on Computer Science Education (SIGCSE), Panel*, Mar 2004.

[63] U. of Illinois at Urbana-Champaign. Directory of Incomplete Lists of Teachers Ranked as Excellent. http://www.oir.uiuc.edu/dme/Ices/incldir.html.

[64] S.-I. Pae and M. C. Loui. Optimal random number generation from a biased coin. In *the Sixteenth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 1079–1088, Jan 2005.

[65] S. L. Pallemulle, V. H. Clark, and K. J. Goldman. Supporting Live Development of SOAP and CORBA Clients. Technical Report TR-2004-56, Washington University Department of Computer Science and Engineering, September 2004.

[66] S. L. Pallemulle, K. J. Goldman, and B. E. Morgan. Supporting Live Development of SOAP and CORBA Servers. In *ICDCS '05: Proceedings of the 25th IEEE International Conference on Distributed Computing Systems (ICDCS'05)*, pages 553–562, Washington, DC, USA, 2005. IEEE Computer Society.

[67] M. Pavelich, B. Jenkins, J. Birk, R. Bauer, and S. Krause. Development of a Chemistry Concept Inventory for Use in Chemistry, Materials and other Engineering Courses. In *American Society of Engineering Education, Annual Conference*, June 2004.

[68] R. Pea. Language independent conceptual bugs in novice programming. *Journal of Educational Computing Research*, 2(1):25–36, 1986.

[69] R. Putname, D. Sleeman, J. Baxter, and L. Kuspa. A summary of misconceptions of high school basic programmers. *Journal of Educational Computing Research*, 2(4):459–472, 1986.

[70] J. Richardson, P. Steif, J. Morgan, and J. Dantzler. Development of a Concept Inventory for Strength of Materials. In *the Thirty-Third ASEE/IEEE Frontiers in Education*, Nov 2003.

[71] N. Riley and C. Zilles. Probabilistic counter updates for predictor hysteresis and bias. *Computer Architecture Letters*, 4, august 2005.

[72] N. Riley and C. Zilles. Probabilistic counter updates for predictor hysteresis and stratification. In *Proceedings of the 10th International Symposium on High-Performance Computer Architecture (HPCA'06)*, 2006.

[73] P. Salverda, G. Rosu, and C. B. Zilles. Formally defining and verifying master/slave speculative parallelization. In *Proceedings of the International Symposium of Formal Methods Europe (FM'05)*, pages 123–138, 2005.

[74] P. Salverda and C. Zilles. A criticality analysis of clustering in superscalar processors. In *Proceedings of the 38th annual IEEE/ACM International Symposium on Microarchitecture (MICRO-38)*, pages 55–66. IEEE Computer Society, 2005.

[75] E. Seymour and N. M. Hewitt. *Talking about Leaving: Why Undergraduates Leave the Science*. Westview Press/Harper Collins, Boulder, CO, 1997.

[76] D. Sleeman, R. Putname, J. Baxter, and L. Kuspa. Pascal and high school students: A study of errors. *Journal of Educational Computing Research*, 2(1):5–23, 1986.

[77] P. Steif. Comparison between Performance on a Concept Inventory and Solving of Multifaceted Problems. In *the Thirty-Third ASEE/IEEE Frontiers in Education*, Nov 2003.

[78] P. Steif. Initial Data from a Statics Concept Inventory. In *American Society of Engineering Education, Annual Conference*, June 2004.

[79] P. Steif, A. Dollar, and J. A. Dantzler. Results from a Statics Concept Inventory and their Relationship to Other Measures of Performance in Statics. In *the Thirty-Fifth ASEE/IEEE Frontiers in Education*, Oct 2005.

[80] P. S. Steif. An Articulation of the Concepts and Skills Which Underlie Engineering Statics. In *the Thirty-Fourth ASEE/IEEE Frontiers in Education*, Oct 2004.

[81] P. S. Steif and J. A. Dantzler. A Statics Concept Inventory: Development and Psychometric Analysis. *Journal of Engineering Education*, Oct 2005.

[82] A. Stone, K. Allen, T. R. Rhoads, T. J. Murphy, R. L. Shehab, and C. Saha. The Statistics Concept Inventory: A Pilot Study. In *the Thirty-Third ASEE/IEEE Frontiers in Education*, Nov 2003.

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

[84] R. Streveler, B. M. Olds, R. L. Miller, and M. A. Nelson. Using a Delphi study to identify the most difficult concepts for students to master in thermal and transport science. In *American Society of Engineering Education, Annual Conference*, June 2003.

[85] S. Sudman and N. Bradburn. *Asking Questions : A Practical Guide to Questionnaire Design*. Jossey-Bass, NJ, 1982.

[86] The Computer Society of the Institute for Electrical and Electronic Engineers and Association for Computing Machinery. Computing Curricula 2001, Computer Science Volume. http://www.sigcse.org/cc2001/index.html.

[87] D. Thissen and H. Wainer. *Test scoring*. Lawrence Erlbaum Associates, Mahwah, NJ, 2001.

[88] P. U. Treisman. Studying students studying calculus: A look at the lives of minority mathematics students in college. *The College Mathematics Journal*, 23(5):362–372, 1992.

[89] K. E. Wage and J. R. Buck. Development of the Signals and Systems Concept Inventory (SSCI) Assessment Instrument. In *the Thirty-First ASEE/IEEE Frontiers in Education (FIE)*, Oct 2001.

[90] K. E. Wage, J. R. Buck, T. B. Welch, and C. H. G. Wright. Testing and Validation of the Signals and Systems Concept Inventory. In *the 2nd IEEE Signal Processing Education Workshop*, pages 1–6, Oct 2002.

[91] K. E. Wage, J. R. Buck, T. B. Welch, and C. H. G. Wright. The Continuous-Time Signals and Systems Concept Inventory. In *the 2002 International Conference on Acoustics, Speech, and Signal Processing (ICASSP), vol. IV*, pages 4112–4115, May 2002.

[92] K. E. Wage, J. R. Buck, T. B. Welch, and C. H. G. Wright. The Signals and Systems Concept Inventory. In *the 2002 ASEE Annual Conference*, pages 1–29, Jun 2002.

[93] K. E. Wage, J. R. Buck, and C. H. G. Wright. Obstacles in Signals and Systems Conceptual Learning. In *the 3rd IEEE Signal Processing Education Workshop*, pages 58–62, Aug 2004.

[94] C.-C. Wu, N. B. Dale, and L. J. Bethel. Conceptual models and cognitive learning styles in teaching recursion. In *the Technical Symposium on Computer Science Education (SIGCSE)*, pages 292 – 296, Mar 1998.

[95] C. Zilles. SPIMbot. http://www-faculty.cs.uiuc.edu/∼zilles/spimbot.

[96] C. Zilles. SPIMbot: An Engaging, Problem-based Approach to Teaching Assembly Language Programming. In *the Technical Symposium on Computer Science Education (SIGCSE)*, Feb 2005.

[97] C. Zilles. "What does a CPU have in common with a fast food restaurant?" A Reflection on Emphasizing the Big Ideas of Computer Science in a Computer Organization Class. In *the Thirty-Fifth ASEE/IEEE Frontiers in Education*, Oct 2005.

[98] C. Zilles. Toward CS concept inventories: Assessing learning in Computer Science. In *the Technical Symposium on Computer Science Education (SIGCSE), Birds of a Feather session*, Mar 2006.

[99] C. Zilles and D. Flint. Challenges to Providing Performance Isolation in Transactional Memories. In *Proceedings of the Fourth Workshop on Duplicating, Deconstructing, and Debunking*, pages 48–55, June 2005.

[100] C. Zilles and N. Neelakantam. Reactive techniques for controlling software speculation. In *Proceedings of the international symposium on Code generation and optimization (CGO'05)*, pages 305–316. IEEE Computer Society, 2005.