Online curricula for monitored, closed-lab first-year CS courses

We propose to improve computer science (CS) instruction so students gain a firm foundation in programming and instructors have more effective ways to support learning. Current courses often deter students from underrepresented groups and produce students who have fragmented and fragile knowledge rather than integrated understanding. New technologies and recent research on learning demonstrate the benefit of reducing lecture and increasing interactive laboratory experience in CS courses to promote coherent understanding.

Building on these recent advances, we have developed a new approach to undergraduate computer science instruction at the University of California, Berkeley (Clancy *et al*., 2003). We propose a three-year project to extend this approach through the development, evaluation, and dissemination of two courses: an introductory Java programming course ("CS 1") and a Java-based course in data structures and programming methodology ("CS 2"). We will collaborate with faculty across the country.

Our courses have several features that promote coherent understanding:

- A diverse set of activities, including case studies, collaborative problems, comparison of solutions, video explanations from different instructors, brainstorms, simulations, and projects provide more and timelier opportunities to learn.
- Collaboration and group work enhance understanding and engage students as both producers of solutions and critics of alternatives (Hoadley, 2004).
- A variety of support mechanisms help students keep up with course assignments and sustain consistent participation.
- Activities refined in classroom trials help underrepresented students, including women, have more success.
- Embedded assessments and data on student interactions enable instructors to use a fine-grained knowledge of student conceptions and progress to customize instruction.
- A comprehensive learning environment enables instructors to allocate limited time where it can be of best use: efficiently diagnosing student needs, iteratively customizing the course materials, and improving TA training.

Prior Work

The proposed work is informed by substantial prior research, including work on curriculum and assessment design in computer science courses, learning environments and collaborative tool use and design, and our experiences with courses taught at U.C. Berkeley and Merced Community College using the new format.

Research on teaching and learning computer science

Over the past twenty years, the PIs have provided research, leadership, and innovation in the teaching of computer programming. From their NSF-supported work in the 1980's, the case study approach to teaching programming emerged[1], and two collections of case studies were published (Clancy and Linn, 1992; Clancy and Linn, 1996). Today, the Advanced Placement CS examinations are partly based on a case study, and most introductory programming textbooks contain at least one "case study" chapter. Later work, also supported by NSF[2], identified compelling issues relating to programming education:

- the origin and application of student conceptions regarding what gets evaluated and what gets taken literally (Davis *et al*., 1993);

[1] MDR 84-70514: *Computers and Problem Solving* [$325,000] 1985-1988. Selected publications arising directly from this project include Linn and Clancy (1990, 1992); Clancy and Linn (1990, 1992a, 1992b, 1996); Clancy, Astrachan, and Matsuoka (2000), Bell, Linn, Clancy (1994).

[2] MDR 89-54753: *Designing Case Studies to Teach Complex Problem Solving: Capitalizing on Advanced Technologies* [$790,000], 1989-1993. Selected publications arising directly from this project include Linn, Katz, Clancy and Recker, (1992); Davis, Linn, Mann and Clancy (1993); Mann, Linn and Clancy (1994).

- the acquisition of attitudes and understanding that facilitate code reuse (Hoadley *et al.*, 1996);
- programming tools that facilitate understanding of the evaluation process and the critique and debugging of solutions (Bell *et al.*, 1994; Mann *et al.*, 1994).

The PIs have also been active in publicizing fundamental work to a larger audience of CS educators, e.g. Clancy and Linn (1999), Clancy (2004). The instructional framework emerging from this work, called scaffolded knowledge integration (SKI), has extended the findings to science education in general (Linn, 1995; Linn and Hsi, 2000).

Research on learning environments

The courses in our current project are supported by a learning environment that extends the SKI framework and integrates findings from related investigations (e.g., Bruer, 1993; Bransford *et al.*, 1999; Means, 1994; Reiser *et al.*, 2001; Roschelle *et al.*, 2000; Scardamalia and Bereiter, 1996; Koschmann, 1996; Koschmann *et al.*, 2002; Slavin, 1995).

One important learning environment has contributed directly towards the current project, in that it serves as the foundation for our current system. The Web-based Inquiry Science Environment (WISE) was developed by Linn and her colleagues in the School of Education at U.C. Berkeley (Linn and Slotta, 2000; Slotta and Linn, 2000; Linn *et al.*, 2004) with support from NSF[3]. The WISE project researched an Internet-based platform for K-12 science activities where students work collaboratively on inquiry projects, making use of evidence from the Web. Five years of classroom trials and refinement have resulted in a stable technology platform and curriculum library that has been used by more than 50,000 students. WISE provides a number of tools for authoring online activities: assessment tools, including tools that provide interactivity and that share student answers after they are made; collaboration tools, with threading and gating options; data analysis, including entry, reporting, and organization; presentation; and reflection, including journaling tools.

University of California Web-based Instruction for Science and Engineering (UC-WISE)

The UC-WISE project aims to provide laboratory-based higher-education courses that incorporate online technology for collaboration, inquiry learning, and assessment, as well as to allow instructors to customize courses, prototype new course elements, and collect review comments from experienced course developers. The UC-WISE system includes a database of annotated learning objects, served by linking to the WISE system, plus portals into the database for students, instructors, and master curriculum developers.

One course titled "Introduction to Symbolic Programming", CS 3, has been fully implemented in the UC-WISE system. First offered in summer 2002, this course has been successful (Clancy *et al.*, 2003). We are constructing UC-WISE curricula for experimental sections of CS 61B ("Data Structures") and CS 4 ("Introduction to Programming for Engineers") to be offered in fall 2004. A recent version of CS 3 can be seen at http://spring04.ucwise.org/.[4]

Table 1 compares UC-WISE and traditionally formatted courses from a student's view.

Traditional	UC-WISE
The student attends 2-3 hours per week of lecture.	The student attends 0-1 hours per week of lecture.
The student works through a set of short exercises, perhaps with a partner, in 2 hours per week of lab.	The student works through a wide variety of exercises in 4-6 hours per week of lab.

[3] REC-98-73180: *The Web-based Inquiry Science Environment (WISE)* [$1,200,000], 1998-present. See http://wise.berkeley.edu. Selected publications arising directly from this project include Slotta (2002), Linn and Slotta (2000), Linn (in press).

[4] To view content, log in as a guest, then move the calendar at the bottom of the initial course portal view to an early date in the term. Clicking on an activity listed in a day will open a window containing the content of that activity. We recommend the use of a recent version of the Mozilla browser. Guest access does not allow viewing of or participation in students' collaborative work, or access to instructor tools to design, create, and organize materials.

The student attends 1 hour per week of "discussion", in which examples similar to upcoming homework exercises are covered.	Students engage in online collaboration; occasional impromptu discussions may arise in lab with a neighboring student, or between TA and a group of students.
The student works on homework, not always clear about skills and techniques that need to be applied. Help, when available, is found through a class newsgroup, fortuitously scheduled teaching assistant office hours, or communication with fellow students.	Most exercises formerly done as homework are now completed in lab. Relevant context is clear from earlier exercises. Help from instructors is always available in the lab section
The student is not always aware of misunderstandings, and may wait weeks to address them.	Staff review quizzes and gated collaboration submissions in lab, and engage a student in timely tutoring if necessary.

Table 1: Comparison between UC-WISE and traditionally formatted courses, from a student's perspective. Students experience a more diverse set of learning opportunities while receiving more consistent support.

Designing a UC-WISE curriculum is a major challenge. The UC-WISE CS 3 course contains all the lab and homework assignments from the traditional version, plus all the programs formerly demoed in lecture, plus other material. Thus the work students do in this course includes everything they did in the former CS 3, plus numerous extra exercises.

UC-WISE benefits for students

Replacement of lecture and homeworks with lab: Deficiencies of lecture-based courses in computer science are generally recognized (Kay, 1998). While computer science instruction generally includes a lab component (Austing, 1979; Tucker, 1991; IEEE Task Force on Computing Curricula, 2001; Parker *et al*, 1990), we seek to expand and take better advantage of lab-based activities relative to their traditional use as a supplement to a lecture and homework-based course. In lab settings, students can engage in their learning rather than passively receive it (Urban-Lurain and Weinshank, 2000). In traditional courses, relatively simple lab exercises are most likely followed by substantially more complex homework problems or projects. With the variety and breadth of materials in a UC-WISE course, the increase in difficulty as students move between activities is small.

In a good lab session, as contrasted with the best possible lecture, students are actively engaged the entire time, forming hypotheses and doing real work. More lab time means more work and, we hypothesize, more learning. Homework and lab sessions offer similar levels of student engagement; a crucial difference is that students are more closely supervised and can more easily receive support in lab sessions.

Frequent embedded assessments: In traditionally formatted courses, students may receive no feedback—even pertaining to possible misconceptions of core concepts—until they perform badly on the first midterm exam. This is usually too late for instructors to respond effectively. Educational research has shown that students have difficulties monitoring their own progress in computer science courses (Davis *et al.*, 1995; Linn and Clancy, 1992; Tucker, 1991; Soloway and Spohrer, 1988; Linn and Kessel, 2003). Many students report success, believing that they are doing fine in courses, when they are actually near failure.

In a UC-WISE class, the typical day begins with a quiz that tests material covered in the preceding lab section. Instructors and assistants can view student answers immediately after they are submitted. Subsequent student activities include self-assessments, gated collaborations (which provide students opportunities to check their understanding of a given concept against that of their classmates), and scripted assessments that provide varying degrees of help for incorrect answers. These frequent assessments keep students apprised of their progress in the course, provide opportunities for staff intervention when students are confused, and motivate students to seek help if necessary (Tanimoto *et al*. 2002).

Just-in-time targeted tutoring: The lab format and the frequent embedded assessments allow instructors to engage in one-to-one and group tutoring sessions. Tutoring has long been known to

be the best method of instruction (Cohen *et al.*, 1982; Bloom, 1984), and has been shown to work without extensive training on the part of the instructor (e.g., Chi *et al.*, 2001). The main drawback of tutoring—that it is very time-intensive and, therefore, costly—is reduced by the UC-WISE format. Students spend most of their time learning from the materials and their collaborators, while the tutoring is efficiently used when confusion occurs. This is often typical in the lab component of a course; it shouldn't be surprising that converting the entire course into a lab should maintain these benefits.

UC-WISE creates additional benefits by providing many tools for the instructor to monitor student progress and, more importantly, understanding. With such information, instructors can initiate tutoring sessions *at the moment* a misconception or confusion is occurring. This crucial advance not only makes tutoring sessions more efficient, but also effects deep student learning (Linn, in press; Bransford *et al.*, 1999; diSessa and Minstrell, 1998; Linn and Hsi, 2000).

Collaboration among students: Research has shown that collaborative activities have a myriad of benefits in all phases of learning and coursework (Slavin, 1995; Koschmann *et al.*, 2002). Still, collaboration in computer science courses is commonly limited to project work and the solving of laboratory activities. That is, students collaborate during the act of programming, but rarely while engaging in higher-order thinking and meta-cognitive reflection on the practice of programming.

In contrast, a UC-WISE course engages students in a *variety* of collaborative activities. A□gated collaboration" presents students with a prompt; after answering, they can review the responses of their classmates. (Contrast this with a question posed to students in a recitation section. Ideally, all students consider the question and contribute their answers to discussion; in practice, only a few consider the question seriously, and only a few—always the same few—contribute answers.) Focused discussion activities provide opportunities for sharing strategies for debugging or avoiding particular kinds of errors, for comparing design or development approaches, and for criticizing and defending alternatives.

Engagement of traditionally underserved groups: Computer science instruction has traditionally failed to engage some populations of students, including women, minority students, and students from community colleges (Linn, 2003; AAUW, 2000; Snyder *et al.*, 1999; Vollman and Eck, 2001; Margolis and Fisher, 2001). The typical lecture and homework-based computer science instruction proves impenetrable and unwelcoming to some students.

In contrast, a UC-WISE course provides a much wider variety of activities, allowing more students to shine. Shy students are encouraged to contribute in collaborative exercises, and receive help without seeking it out. Students who might otherwise drop out are kept in the course with the support provided by staff and the learning environment: UC-WISE succeeded with groups of students typically at risk for success in computer science. Students who scored in the lowest quartile of the course were a full grade better, on average, than those of the previous year's traditionally instructed course (Clancy *et al.*, 2003). In the spring 2004 CS 3, female students outscored males by one-half grade, and seven of the top nine students were female.

Qualitative analysis (from surveys and interviews) suggests that UC-WISE technology and pedagogical approaches promote positive views towards computer science by women and non-CS majors. This success was due in part to the support provided to students in monitoring their progress on a daily basis, rather than the previous midterm-based approach. In addition, the course pace and materials have been pilot-tested and refined to ensure that all topics are communicated clearly and that course demands and grading rubrics are unambiguous. As a result, students know where they are and where they are going and can use this information to develop their own ability to monitor progress.

More efficient interaction with staff: Consider the obstacles presented to a student seeking help in a traditional course. Only short focused questions are appropriate for the class news-group. Staff office hours may not coincide with the student's free time. A staff member, presented with the student's question, will lack context—why is the student asking this question?—and history—what does this student already know?—on which to base a response. In contrast, students in a UC-WISE course ask questions in the context of the day's lab activities, consulting a staff member who most likely has been observing the student's work throughout the

term. As anecdotal support, tutoring sessions for CS 3 at Berkeley's campus tutoring center have been cancelled because of a lack of students seeking help. This coincides with the course's use of UC-WISE, and this is the only computer science course for which this has happened.

UC-WISE benefits for instructors

The fine-grained activities in a UC-WISE course expose a substantial amount of information about student conceptions and misconceptions. We have learned in two years of UC-WISE CS 3 of a variety of previously unrecognized misunderstandings (Ryan, 2004), and have modified or invented activities to address them.

A UC-WISE course typically involves less lecturing than its traditional counterpart, and correspondingly frees up instructor time to be spent flexibly. Example activities include teaching assistant training, review of student work to diagnose problems, and inventing new pieces of curriculum. In this way, instructors can concentrate on their strengths, devoting efforts to the aspects of teaching that they do best and enjoy most.

A UC-WISE course also provides more subtle benefits for a lecturer. Lecture preparation in a traditional course is often done without much awareness of student understanding; a lecturer is likely to overprepare as a result. Student performance on UC-WISE activities, in contrast, provides data on *actual* misconceptions that a lecturer might target. (Techniques such as peer instruction have the same goal, but in our opinion achieve it less effectively.)

Teaching assistants in a UC-WISE course work essentially the same number of hours as in the traditional version. Extra time needed to supervise lab sections is offset by reducing or eliminating office hours; reducing or eliminating preparation for recitation section; and, because most questions are answered in lab, a drop in class-related electronic mail and newsgroup traffic.

Summary: benefits of UC-WISE

Our experience thus far with UC-WISE courses suggests that we can improve three aspects of introductory programming education.

- We can provide more opportunities and more timely opportunities for student learning. Our online lab activities include all those formerly used in a traditionally-organized counterpart course plus a lot more. Embedded assessments and targeted tutoring ensure that no student falls behind. This pedagogical support, coupled with the variety of course activities, increase motivation and successful performance of underrepresented students.
- We can increase instructor productivity. Less time is spent on ineffective preparation for lectures; more time is spent on diagnosing student needs, improving course activities, and helping course staff to help students better. As a result, the instructor acquires a deeper understanding of student conceptions.
- We can provide customizable materials that are readily adapted to new courses. More and more universities are offering introductory programming specific to disciplines such as engineering, biology, physics, and social science.

Project Plan

We propose to design, implement, evaluate, and disseminate two courses and supporting materials. The two first-year courses are:

1. "Introductory Programming in Java"
2. "Data Structures and Programming Methodology using Java"

We propose a three-year project that extends these courses from U.C. Berkeley to collaborating campuses and finally to a wider audience; the *Timeline* section below offers more detail. We will also focus on instructor training, devoting significant energy to both the creation of workshops and training modules as well as evaluation of the instructors and teaching assistants, how well they learned, and how well our pedagogical approach fared across diverse settings.

We have assembled a team of collaborating faculty across the country to attend and help refine our instructor workshops, teach the new CS courses in their initial dissemination phase, customize the courses for their specific needs, and help support the evaluation of student learning and instructor use. In year three, we will more widely disseminate the materials to institutions we

identify after the project is underway. In this second round of dissemination, we will focus on sustainability issues. The *Dissemination* section below offers more detail.

At present, there are no "standard" versions of these courses. The ACM/IEEE Computing Curriculum 2001 (http://www.acm.org/sigcse/cc2001/) lists three main organizations of the introductory sequence: the procedural approach, an object-oriented approach, and an approach that starts with functional programming and then switches to objects. Even teaching object-oriented programming, which Java supports most directly, instructors may choose various foci. Similarly, data structures content may be covered in a variety of ways and sequences. We have experience with several of these approaches at Berkeley.

Our modular approach will accommodate all of these organizations. For instance, the sequence will be reorderable and easily able to accommodate courses of differing lengths (e.g. both semester and quarter courses), the projects will be able to grow and shrink flexibly, and the materials will be useful in contexts ranging from second courses in computer science in high school through advanced discipline-based courses that desire a computing emphasis. Customization by instructors will be easy to integrate and track.

Course designs

In this section, we describe the educational materials that we plan to create. Before getting into details, we discuss three major themes in our approach to instructional design. First, consistent with the UC-WISE pedagogical philosophy, we will emphasize learning by doing and knowledge integration. In the CS 1 and CS 2 courses, learning by doing means experimenting with language constructs and with components of various class libraries (see, for instance, Reed *et al.*, 2002). Knowledge integration activities include practicing patterns of design and development, learning from experience and linking and integrating programming knowledge, and building accurate models and internal representations of programming processes.

Second, consistent with the emphasis on larger project assignments in many offerings of the CS 2 course, we will include numerous project-related activities supported by *case studies*. Case studies are worked-out problem solutions accompanied by detailed narratives of the solution process and questions that involve the reader in prediction and reflection, analyzing solution features, evaluating design alternatives, and applying solution techniques to new problems (Linn and Clancy, 1992; Clancy and Linn, 1992a; Clancy and Linn, 1992b; Clancy *et al.* 2000).

Third, we will provide opportunities for mentorship between instructors and students. Examples include design/development reviews and project planning meetings (Davis and Linn, 2000). Another example is the targeted tutoring mentioned above. Gated collaborations, to which all students must respond, provide more subtle opportunities for instructor mentorship. Careful design of these activities results not in students having the feeling of being "quizzed", but in a natural curricular flow (Carlson, 2003). By including activities that involve reflection and interaction, we provide for student self-monitoring as well as instructor monitoring. Examples of such activities include journal maintenance and self-assessments.

CS 1 and CS 2

Collectively, the material for the two Java courses will address Java features, data structuring techniques, and fundamentals of programming methodology:

- Creation and use of programs and applets.
- Objects and classes: use and definition of classes, methods, and parameters; use of inheritance; definition and analysis of an inheritance hierarchy.
- Flow of control: coding and analysis of conditionals and loops; use and analysis of recursion; catching, defining, and throwing exceptions; event handling.
- Standard data structures and data types: definition and use of arrays and linked structures; creation and use of linked lists, binary trees, heaps, hash tables, and graphs; use of java.util classes; design using search (set, dictionary) and scheduling (stack, queue, priority queue) types; implementation of these types; analysis of alternative implementations.
- Design with interfaces: supplying and understanding pre- and post-conditions in specifications; using a specification expressed as a set of procedure headers with comments;

providing suitable comments for modules, data types, and methods.

- Management of relatively large programs: incremental development; design of thorough test suites; efficient debugging.

Activities will include online quizzes, programming assignments, analysis and modification of existing code, collaborative discussions, and reflection.

A review team of outside experts will be assembled to evaluate our curricular materials with respect to coverage and depth of treatment of core concepts in CS 1 and CS 2, as detailed by ACM/IEEE Computing Curriculum 2001 (http://www.acm.org/sigcse/cc2001/). The *Management plan* section contains more information on the review team.

Table 2 details a sample day of curriculum on testing and debugging. The activities in it are based on a procedure named contains1MoreThan that is intended to determine whether its first string argument is the result of inserting exactly one character into its second argument. They address problems encountered by Zweben *et al.* (1989) and Leventhal *et al.* (1994) on positive test bias.

Step type	Step contents

Gated collaboration	A student proposes to test contains1MoreThan with the following calls:
	`System.out.println (contains1MoreThan ("ABC", "BC"));`
	`System.out.println (contains1MoreThan ("ABC", "AC"));`
	`System.out.println (contains1MoreThan ("ABC", "AB"));`
	Give the shortest implementation of contains1MoreThan that would produce the right answers for those calls.

	Describe the deficiencies of this set of test calls.
	Provide a missing test case, along with a description of a bug that it would reveal.
Scripted assessment	The class buggy1.java contains a program with a buggy version of contains1MoreThan. Identify a test suite that reveals the bug.
	The class buggy2.class contains a program with a buggy version of contains1MoreThan. Find a pair of arguments that reveal the bug and are as short as possible.
Exercise	The class buggy3.class contains a program with a buggy version of contains1MoreThan. Find the bug. Keep track of the test cases you used.
Reflection	Put an entry in your journal describing test cases that you needed to reveal the bugs of the previous steps, especially those you hadn't previously thought of.
Design/defend discussion	Based on your experiences in the previous steps, design and defend a thorough black-box test suite for contains1MoreThan.
Gated collaboration	Suppose a solution to the function is based on the following pseudocode {…} Design and defend a thorough glass-box test suite for contains1MoreThan.
Brainstorm/ reflection	List boundary conditions for the following complex problem {…}. Describe your solution.
Discussion (online or face-to-face)	(for half the class) Explain why it would be easier to test your own version of than a version that someone else has written. (for the other half) Explain why it would be easier to test someone else's version than your own. Then try to persuade someone in the other group to agree with you.

Table 2: A day of curriculum on testing and debugging in CS 1.

Table 3 details a day of curriculum on hashing. Students have prepared for the day's activity by reading a textbook section on hashing. Some of the steps use an InstrumentedHashTable class that collects and displays statistics on collisions, and a TestString class that's used for experiments.

Step type	Step contents
Quiz	Explain how an object's hashCode and equals methods are used in the call of a put method.
Gated collaboration	A call to put results in calls to your hashCode and equals methods. Give another example of a builtin method that, when called, results in calls to methods you have written.

Exercise	Run the InstrumentedHashTable program using a data file. Describe what's wrong with its output.
	Add println statements to the equals and hashCode methods to verify your answer to today's quiz.
Gated collaboration	The equals method in the TestString class has a bug. Explain how you might deduce that from the output in the previous two steps.
Scripted assessment	Fix the bug in the equals method.
Gated collaboration	The hashCode method supplied in the TestString class is a terrible one. Run the InstrumentedHashTable program using the words file and plot the graph from the produced information to verify this. Then explain what the problem with the hashCode function is and how the graph shows that the hash function is bad.
	Replace the TestString hashCode method by one that returns the sum of the ASCII values of the characters in the string being hashed, then run it using the words file. Is this hashCode function a good one for this data, or not? Briefly explain.
	The given hashCode method turns out to work well with collections of longer strings, for example, book titles. Give a reason for this behavior.
Exercise	Fix your class to use the String.hashCode method, and test it.
Discussion (on-line or face to face)	Invent and defend a hashCode function for Tic-Tac-Toe boards.
	Students learning to use Java hash tables often mistakenly use the statement `put (s.hashCode ( ));` when trying to add an object `s` to a hash table. What misunderstanding would produce this error?
Gated collaboration	What's the difference between handling collisions by chaining and by open addressing?
Homework	Design an experiment to determine if java.util.HashMap uses chaining or open addressing.

Table 3: A day of curriculum on hashing in CS 2.

Metadata

All of the materials (learning objects) developed in this project will be extensively annotated with metadata. For materials we create at U.C. Berkeley, we will start with relevant fields from the Dublin Core and IEEE LOM specifications. We have been researching areas to extend these standards where they are weak, especially with fields specific to the domain of computer science. For example, we have a detailed list of the topics covered in typical first- and second-semester programming courses, structured to capture prerequisite relationships. Topics can be associated with typical misconceptions, and learning objects tagged with how well they address specific misconceptions.

We are also developing mechanisms for tracking learning object "references": specific examples, code pieces, problem contexts, or the like that are introduced in one learning object and referred to in another learning object (in this situation, the second object references the first).

As UC-WISE courses are adopted at other institutions, instructors will add to the materials that we developed. We will simplify the tagging of activities with appropriate metadata while still making the customization process easy and efficient for busy instructors.

Collaborating faculty and customizations

We have assembled a diverse team of collaborating faculty that will help our evaluation efforts through the collection baseline and learning data early, provide feedback on our professional development materials, refine and customize the curriculum with attention to their local needs, and spark our dissemination efforts in years two and three.

Our faculty collaborators include the following: Dan Garcia and Kathy Yelick (U.C. Berkeley), who will have developed a pilot version of our CS 1 course in fall 2004, targeted towards engineers, and will teach it in 2005; David Kay (U.C. Irvine); Paul Kube (U.C. San Diego); Owen Astrachan (Duke); Elizabeth Edmiston and Sung-Sik Kwon (North Carolina Central University, a minority serving institution); Paramsothy Thananjey (Vista Community College). They will assist us closely as the curriculum is initially developed in 2005. Several of our collaborators have substantial experience developing innovative programming curricula.

Collaborating faculty will attend a multi-day workshop in summer 2005, hosted by North Carolina Central University, and will teach their first course in spring 2006 or fall 2005. Faculty will attend another workshop in summer 2006, to review their courses and plan customizations of curricular material. They will teach again in fall 2006 or spring 2007. Our collaborators will receive a stipend and reimbursement for expenses.

The collaborating faculty will be encouraged to modify the materials developed at Berkeley as they use them at their local institutions. We will support successive refinement: the second time that they teach this course, in the fall 2006/spring 2007 year, the faculty will target some portion of the curriculum that did not serve their students well. Interviews with faculty, as discussed in the *Evaluation* section below, will help determine what prompted the customization, how easy it was to perform, and how well it succeeded.

Another anticipated advantage of this collaboration is the identification of *curriculum design patterns*. Two relevant projects include the major focus on software design patterns in software engineering (Gamma *et al*. 1995) and pedagogical patterns being developed by Bergin and colleagues (see http://csis.pace.edu/~bergin/). Proto-patterns of combinations of WISE components are already emerging from our experience with UC-WISE courses. One example that shows promise for student learning is a daily organization that starts with a quiz based on assigned reading, continues through synthesis, analysis, and experimenting activities intermingled with short scripted assessments and gated collaborations, and ends with a discussion focusing on how to deal with difficulties in understanding the day's topic. Reuse of these patterns will enable us to improve them.

New and modified objects will over time begin to accumulate in our learning object repository. By making management of these additions easy, and persuading instructors to use the metadata tagging facilities described above, we hope to have a living resource that will maintain currency and quality through sharing and peer review. By sharing our materials with larger digital libraries—see the *Dissemination* section below—we will further support this growth.

Professional development materials and workshops

Paramount to the success of this course in diverse settings is an effective process for developing faculty course leaders, graduate instructors, and, when present, student assistants. We will develop a series of workshops, modules, and mentorship approaches for all three of these populations. Our materials will cover instructional design principles and goals, the process of customization, tool design and usage, monitoring and tutoring guidelines, and term- and session-length course management techniques.

Our work on professional development materials and workshops will start in the first semester, as summarized in the *Timeline* section below. In the opening workshop (summer 2005), collaborating faculty will first work through the materials for faculty course leaders, the role that all of them will be taking on within the project. These materials will include an overview of the curriculum materials; introduction to the UC-WISE learning environment and instructor portal, where learning objects are created, modified, and sequenced; best practices on staff development and training, including exposure to the materials designed for teaching assistants and student assistants; and more. Similar activities and material will be provided for graduate teaching assistants. Later in this workshop, the collaborating faculty will critique and help refine all of the professional development materials.

These meetings will emphasize the features within our learning environment for monitoring students' online programming and collaborative activity, and for effectively using this information to engage students in tutoring sessions. Challenges of initiating and conducting a productive tutoring session go significantly beyond mastery of the technology and curricular materials (Chi, 1996; Graesser, Person, and Magliano, 1995):

- The student is typically unaware of his or her misunderstanding. How should the instructor approach the student in an unthreatening way?
- How should the instructor choose examples or activities to address misconceptions?
- How should the instructor pose these examples and activities so as to maximize active learning by the student and avoid "giving away the answers"?

Our professional development materials will include role-playing scenarios in order to enable instructional staff to practice specific responses to student dilemmas. The staff will enact these scenarios and then discuss possible resolutions in order to develop the capability of responding in real time during a laboratory session.

Many problems arise when student pacing within a course diverges, with some students substantially farther ahead in the material than others, as pilot studies in U.C. Berkeley have shown (see also Carrasquel, 1999). When pacing is not synchronous, collaborative activities lose effectiveness, student-to-student assistance is diminished, and instructor monitoring become more difficult. To avoid these problems, we will alert instructors to ways to keep the class in general synchrony, via directed tutoring of students that are falling behind, implementing grading policies that encourage timely completion of work, including "optional" materials for advanced students at strategic points, and forcing synchronization by announcements or by simple encouragement. Small adjustments made early should reduce divergence later in the course.

We will develop both an online and a face-to-face approach to train student course assistants[5]. Student assistants generally receive credit for their work with the lab sessions, and, during a course, will be required to attend a one-hour weekly meeting with the graduate student instructor and a monthly meeting with the faculty leader as well as regular meetings with students. We will develop modules for student assistants that prepare them to deal with student difficulties, foster an inquiry atmosphere in the classroom, and develop an understanding of the types of misunderstandings that arise as student attempt to learn the relevant programming skills and concepts. In the first few terms at an institution initiating UC-WISE courses, there will be course assistants who are unfamiliar with the system. We thus will also tailor some of the regular curricular materials into training modules.

Two other workshops will be developed in this project. A second workshop will take place in summer 2006 at U.C. Irvine or U.C. San Diego. This workshop will focus on curricular customizations, refinement of the professional development materials, and in review of the project as a whole. The third workshop will be designed for delivery at professional meetings such as SIGCSE, and will consist of selected portions of the opening workshop targeted for computer science instructors interested in making use of UC-WISE materials or courses. This workshop will be proposed for SIGCSE 2007, as detailed in the *Dissemination* section below.

Tailoring of the technology platform

We will tailor our existing learning environment in minor ways for this project. Our current system can support the proposed courses. We will upgrade to the existing production server, to handle the increased dissemination load, as detailed in the budget notes.

We will develop a "scripted assessment" tool for Java, similar to a tool we developed for our Scheme-based course. This tool will allow instructors to create different interfaces and functionality to serve different purposes, from heavily constrained activities early in the course to more open-ended activities as students gain programming expertise. We will adapt Eclipse [http://eclipse.org], an open-source interactive development environment (IDE) with powerful components that can be used for building specialized IDEs. Through the integration of a Java source interpreter (e.g., Dynamic Java [http://koala.ilog.fr/djava/], BeanShell [http://beanshell.org/]) a student's code can be run, which eliminates or drastically reduces the need for pattern matching to determine correctness and provide useful hints.

The scripted assessment tool for Java will take the place of an IDE in many instances, especially in the early materials. That is, the student will explore and program in constrained settings, rather than in an open-ended IDE. Additionally, our tool will be embedded in the

[5] Student assistants are generally undergraduates majoring in the discipline (computer science) that receive units for assisting during closed-lab sessions. They are most often unpaid but receive units. They are a useful presence, although not a required one, and generally very positive about the experience. Some institutions don't have mechanisms in place to offer credit for volunteer work in this way. We have never encountered a two-year institution that does, for instance.

curriculum, so that students won't have to switch contexts to explore programming concepts. Students will use standalone Eclipse for their project work later in the course.

The development will take place in the first eighteen months of this project. We will leverage existing work for the bulk of our system. The scripted assessment tool's macro language, for instance, will be our current Scheme-based one. The authoring interface will also be heavily based on the tool used in the Scheme course. Additionally, our tool will be released as an open source project, which should help support its continued development.

Timeline and milestones

We have targeted a January 1, 2005 start date for this project, although organizational work will begin earlier. Some material for both courses will be pilot-tested in experimental Berkeley sections in fall of 2004, before funding begins. We are fortunate to have these pilot tests funded by U.C. Berkeley; such matching funds will help ensure that we can have three revisions of our curriculum.

Time	Milestones
Fall 2004 (unfunded)	U.C. Berkeley pilot tests are conducted for Course 2 (Clancy) and Course 1 (Garcia and Yelick). Assessment and evaluation in these courses begins.
Spring 2005	Student learning and instructor measures from the fall courses are evaluated. Proposal of permanent installation of the UC-WISE CS 61B is made at Berkeley with support of evaluation. Full development of curriculum begins. Refinement based on evaluation of pilot tests identifies some successful materials. New collaborative materials are developed. Server is installed; development of Java scripted assessment tool begins. Gathering of baseline data at participant sites. Opening workshop and professional development materials are designed.
Summer 2005	Opening workshop is held at NCCU. Professional development materials are refined with feedback from opening workshop. Prototype Java scripted assessment tool is evaluated in a traditionally-formatted CS 2.
Fall 2005	Clancy teaches a full-scale Course 2 at Berkeley. Java tools are revised during semester; new functionality is completed. Collaborating faculty unable to teach in following spring give courses now.
Spring 2006	Collaborating faculty teach course. Each participant site visited and observed, with instructor interviews. Refinement of course materials, especially those using new tools, is undertaken. Secondary workshop is designed, focusing on customization and local strategies.
Summer 2006	Evaluation of customization and adaptation proceeds. Student learning is analyzed. Second workshop is held in Southern California; customizations and experiences are reviewed.
Fall 2006	Courses are taught at U.C. Berkeley and other campuses, supporting participant customization. SIGCSE workshop is developed. Additional dissemination targets are identified, supported.
Spring 2007	Courses are taught at participant campuses, and additional sites as discovered. SIGCSE workshop is delivered. Instructor interviews are again conducted, with focus on customization.
Fall 2007	Summative evaluation is completed

Table 4: Timeline for material development, implementation, evaluation, refinement, and dissemination.

Evaluation Plan

Our plan to evaluate the impact of the new courses involves documenting student learning, instructor curriculum customization, and instructor workload. Each offering of the new courses will be accompanied by comprehensive review of course materials, student performance, instructor revisions, and instructor activities. Over the proposed three-year research program, we

will study how instructors improve on their course designs and instructional practices in response to student achievement. We will document student achievement across diverse sites, with attention to demographics, course sequencing, and instructional setting. At some sites, we will track students longitudinally through CS 1, CS 2 and the courses they take in the following year.

Preparation. By the end of fall 2004, we will have pilot versions of the Berkeley CS 4 and CS 61B courses. At this point, we will evaluate the impact of the various course components relative to Berkeley baseline data. We will also submit the curriculum, our evaluation of course elements, and sample student work, for critique by a panel of outside experts; this panel will include all the instructors who have agreed to use the courses as well as advisory board members. With our collaborating faculty, we will identify curriculum modifications necessary to accommodate course offerings at their institutions.

We will evaluate the effectiveness of these course elements using student work as well instructor feedback. We will use this information to redesign the course, to identify places where instructors might effectively customize the course, and to tailor our authoring environment so that it supports the customization process.

Course trials. Each subsequent course offering will begin with a workshop to introduce the materials and support the process of customization. At the workshop, instructors will be provided with illustrative samples of student work from the pilot trials, and guided in reflections about their own students' conceptualizations. Instructors will identify questions to investigate, perhaps with "compelling comparisons" that contrast two different forms of instruction (e.g. with and without lecture), and will jointly design comprehensive assessments to document student learning and impact of instructor customization. These comparison studies will illuminate the range of applications for various instructional patterns and shed light on their sensitivity to customization.

Each course offering will include pretests, embedded assessments, quizzes, midterm and final exams, observations, student interviews at regular intervals, and instructor interviews. We are examining tools to allow randomization of materials presented to individual students that would enable randomized experimentation within the course (Titterton, 2000).

Assessment of student learning. UC-WISE employs embedded assessments that document student learning, making it visible for instructors and researchers alike. For example, our prior work has employed these measures to track the frequency of late assignment submissions, the use of tutors, and the dropout rate for various groups. (Instructors have occasionally been surprised by misconceptions these assessments revealed.) We will refine the assessments and other outcome measures after each course trial.

In addition, we will employ standardized items drawn from a battery of tested items. We will correlate student performance on these measures with records of their performance on specific course elements (e.g., projects, quizzes, midterm and final exams). Using the correlation matrix as a baseline reference, we will be able to determine whether revisions to course elements have helped students to learn more or become more proficient in their coursework.

New assessment items will be administered to advanced students and experts, to be sure they measure skills that have enduring value. We will also use think-aloud interviews with exemplary and more typical students in order to determine whether or not students draw on relevant aspects of the course in order to solve class assignments.

To compare populations within and across institutions, we will use online surveys that we have developed over the past five years. These materials will document students' prior programming experience, gender, major, self-reported learning practices, predicted grade, and beliefs about programming. Items measuring reactions to our learning environment will be drawn from the Flashlight program (see http://www.tltgroup.org/programs/flashlight.html). We will devise specific approaches to determine the relative success of students from underrepresented groups.

Student attendance will be taken at every lab and lecture session, and related to student usage that is off-site but during the session in question (i.e., watching a live stream of the lecture, or working through the material at home during lab hours to take advantage of real-time collaborative materials) and usage that is both off-site and off-hour.

Customization. The UC-WISE system allows instructor customizations to be captured automatically; instructors will be interviewed about the rationale underlying their choices. We

will examine the variation between university, community college and other groups in terms of how they tailor materials to address different student populations or instructional settings. These interviews, together with similar interviews of graduate lab instructors and analysis of student coursework captured by UC-WISE, should reveal productive curriculum design patterns and their sensitivity to application in diverse instructional situations.

Instructor workload and effectiveness. Observations, course logs, and interviews will allow us to evaluate the impact of UC-WISE on the workload and effectiveness of instructors. We are especially interested in the instructors' pedagogical practices, since the UC-WISE course format allows for greater levels of student-instructor interactions than are possible in traditional lecture courses. We will establish baseline capabilities using scenario-based professional development activities that engage instructors in scenarios where simulated students display a particular misunderstanding. We will videotape responses and discuss alternative approaches. By repeating these scenarios after the course has been taught, we will determine the new skills and capabilities instructors and TAs have developed.

Both site-based and external evaluations will be used. Berkeley staff will travel to each institution in the second and third years to make classroom observations. The collaborating faculty will provide evidence through the professional development meetings and customizations, and will support our collection of student responses from the UC-WISE database. Instructional staff will log the time they spent working on the course, and comparisons will be made with instructor estimates from previous or concurrent courses. In year two, when curricular customizations are performed, instructor load will again be measured and compared within categories (e.g., time spent customizing, monitoring student performance, mentoring other instructional staff, etc.).

Summary evaluation. In year three of this project, we will evaluate the comprehensiveness of the learning object repository, determine how faculty use the shared resources, and study the reuse of materials. We will develop methods for assessing the repository for the dissemination part of the project. To determine how the UC-WISE courses become integrated within the computer science departments of the partner institutions, we will interview instructors concerning their enthusiasm about the system, comparing those teaching UC-WISE courses with their colleagues who are not teaching UC-WISE courses.

Evaluation responsibilities of project participants

Outside evaluation will be undertaken by Heller Research Associates, directed by Joan I. Heller. They will be responsible for overseeing all evaluation aspects of this project. This oversight will involve several large tasks: contributing to and approving the evaluation plan as outlined above; evaluating professional development materials through interviews with instructional staff and SIGCSE workshop participants; interviewing collaborating faculty on the success of their customizations; certifying the analyses of the development team and collaborating faculty on student learning; and assisting with yearly report generation with formative and summative findings.

The collaborating faculty will be responsible for sharing exam questions and data from earlier exams given at their institution, running and videotaping aspects of the professional development activities, performing some basic analyses of surveys and collaborative discussion entries, and observing the lab-sessions. All levels of instructors—faculty, graduate student, and student assistant—will participate in surveys and interviews.

The development team will generate analyses of student learning and student pacing using the breadth of data generated by the learning environment. The developers will support investigations of the evaluator that require customized access to our databases and environments (e.g., tracking customizations of each curriculum).

Management Plan

Michael Clancy (PI) is Senior Lecturer in the Computer Science Division at U.C. Berkeley. He has taught and coordinated lower-division courses and trained teaching assistants at Berkeley

for more than 25 years. Clancy has contributed widely to the body of research on computer science education, including work on case studies, tool design and use, and curricular activities. He has developed many of the materials used in the lower-division CS courses at Berkeley, and he designed and taught the first UC-WISE course. Clancy will take the lead in designing and refining the curricular materials for both courses, although his focus will be on CS 2.

Dr. Marcia C. Linn (co-PI) is Professor of Education in the Graduate School of Education. A pioneer in the research of learning technologies, Linn's scholarly work includes two decades of research in science classrooms and undergraduate courses, investigating the most effective designs for technology-enhanced projects. Linn will participate in development of case studies and project-based activities, assist in the creation of the professional development materials, and head dissemination efforts, including production of reports and future solicitations for support.

Dr. James Slotta (Senior personnel) is a researcher and lecturer in the Graduate School of Education. Slotta led the development of a precollege learning environment and currently coordindates the technology development for an NSF Center for Learning and Teaching called Technology-Enhanced Learning in Science (TELS). He will coordinate the research findings between TELS and UC-WISE.

Dr. Nathaniel Titterton (Project Director) is a professional researcher in the Computer Science Division at U.C. Berkeley. He has a background in computer science, statistics (M.A., Berkeley) and in instruction, and has contributed to the design of the UC-WISE system and the introductory course (which he will teach in the 2004/2005 academic year at Berkeley). Dr. Titterton will direct the work in this project, including the design of technology components, supervision of programmers, coordination of partnerships with collaborating faculty, investigations into student learning, and design of the professional development modules and workshops.

Dr. Joan I. Heller (evaluation), director of Heller Research Associates, will lead the outside evaluation on this project. She has expertise on analysis of children's and adults' performance and expertise, thinking, problem solving, information processing, and misconceptions. Her design and evaluation of self-instructional manuals to guide student learning of the LOGO programming language is especially relevant for the proposed work. Relevant publications include Heller (2004, 2003a, 2003b, 2001). Dr. Heller will oversee the summative evaluation of the project, instrument design and validation, data collection, data analysis, and report writing.

Collaborating faculty. We have assembled a diverse team of collaborating faculty (see letters of support). Dan Garcia and Kathy Yelick (U.C. Berkeley) will have developed a pilot version of our CS 1 course in fall 2004, targeted towards engineers, and will most likely teach it in 2005. Dr. Garcia will present his experiences at the first workshop and will contribute with additional customizations. David Kay (U.C. Irvine), Paul Kube (U.C. San Diego), and Owen Astrachan (Duke) are coordinators for introductory computer science curricula at their institutions, and have engaged in substantial work on innovative CS materials. PI Clancy has collaborated with all of them.

Sung-Sik Kwon and Elizabeth Edmiston are professors at North Carolina Central University, a minority serving institution with a diverse student body. Dr. Kwon is in CS, and will teach CS 2; Dr. Edmiston teaches in the business school, and will teach CS 1. Having perspectives from both departments will be valuable. NCCU will host our summer 2005 workshop.

Paramsothy Thananjey is an instructor with Vista community college, having taught for 8 years. He will interact more closely with the project team at U.C. Berkeley than the rest of our collaborators, attending our curriculum development meetings on a bi-weekly basis. We feel that his input on the needs and nature of community college instruction will be invaluable.

Jeff Wright, Dean of U.C. Merced College of Engineering, will continue collaboration with us after our successful UC-WISE-based course in Merced Community College in spring 2003. As Merced's funding status becomes clearer, we will either work with newly hired computer science faculty or an instructor at a community college with an accreditation relationship.

Review Team. A team of experts in computer science instruction and Java will review our curricular materials for quality, completeness, and accuracy. Review team members include Richard Pattis (Carnegie Mellon), Mark Weiss (Florida International), and Julie Zelenski

(Stanford), all experienced teachers of introductory programming courses and innovative curriculum developers (see letters of support).

Dissemination

The audience for this project includes teachers of introductory computer science in settings ranging from two-year institutions through universities. (Our materials will also be appropriate for AP CS high school teachers.) We have targeted a range of instructors to participate directly in this project, and have plans to disseminate our UC-WISE courses, materials and findings more widely in year three and beyond.

Availability of materials. The materials to be disseminated include all curricular materials, including multiple versions of individual activities geared for different student populations; sequences of those materials for different populations and terms (e.g., semester and quarter, and perhaps shorter courses); professional development materials, workshops, and activities for all levels of instructional staff, including usage notes for particular curricular materials; and scoring rubrics and sample answers for student assessments.

All materials developed under this grant will be made available for free to the public through our Web accessible repository, excepting rare cases of assessment items that need to be kept private. We will publish our efforts widely, maintaining listings on CITADEL (a repository of the NSDL), CSTC (another NSDL repository), the ACM Special Interest Group on Computer Science Education website (http://www.acm.org/sigcse/topics/), SMETE.ORG, and other digital repositories. All materials produced in this project will be tagged with standard metadata. We will assemble sequences of our materials that the College Board can link to, targeting high-school teachers across the country who prepare students for the AP CS exams.

Materials can be exported from the UC-WISE format (e.g., to HTML), but doing so will lose the server tools and services (i.e., storage of student interactions and tools to view them).

Dissemination of research findings. Our research findings will be disseminated through journal articles and conference presentations. We plan on SIGCSE presentations in years two and three, and a workshop at professional meetings for faculty (e.g., at SIGCSE) for professional development in year three. We will continue to run these workshops, assuming reasonable interest, and will also assist users interested in offering workshops and tailoring professional development materials for their region.

Self-sustaining national distribution. The UC-WISE technology platform includes features that can aid dissemination, such as a scalable Web-based architecture that can support many courses and can be replicated on local servers (Slotta, 2002). Instructor tools allow course designers to search for and import curricular materials from other instructors; further integration with digital libraries will expand this process. The UC-WISE source code is an open source project, enabling continued development and installation by interested parties. It is designed to allow easy integration of new technologies (see the NSF funded TELS center: http://www.telscenter.org/research/technology.html).

One long-term goal of this work is to establish a dynamic community of computer science instructors. This community will enable participants to find relevant materials, develop new assessments, contribute their efforts to the community, and participate in relevant discussions with a wide audience. If successful, this model will ensure sustainability and growth. In particular, we don't expect to need a commercial publisher.

As UC-WISE course use grows, institutions will need to provide local servers. Hardware and support shouldn't represent an unreasonable expense for computer science departments.

Evaluation of dissemination effort. In fall 2007 (year three), evaluation of our dissemination efforts will be completed. This will include interview data from participating faculty on their interest in continuing UC-WISE formatted courses, as well as interest from colleagues that will have been exposed to the ideas; year three workshop enrollment and follow-up data; contact with and use of UC-WISE by outside faculty during the project; and growth of contact with materials stored on our servers.