Renaissance Computing: An Initiative for Promoting Student Participation in Computing

Leen-Kiat Soh, Ashok Samal, Stephen Scott
Department of Computer Science and Engineering, Univesity of Nebraska
Lincoln, NE 68588
1-402-472-3200
{lksoh,samal,sscott}@cse.unl.edu

Stephen Ramsay
Department of English
University of Nebraska
Lincoln, NE 68588
1-402-472-3301
sramsay2@unl.edu

Etsuko Moriyama
School of Biological Sciences
University of Nebraska
Lincoln, NE 68588
1-402-472-4979
emoriyama2@unl.edu

George Meyer
Department of Biological Systems Engineering
Univesity of Nebraska
Lincoln, NE 68588
1-402-472-3377
gmeyer1@unl.edu

Brian Moore
Department of Music Education
School of Music and Fine Arts
Univesity of Nebraska
Lincoln, NE 68588
1-402-472-2537
bmoore1@unl.edu

William G. Thomas
Department of History
Univesity of Nebraska
Lincoln, NE 68588
1-402-472-8318
wthomas4@unl.edu

Duane Shell
Department of Education Psychology
Univesity of Nebraska
Lincoln, NE 68588
1-402-472-6981
dshell2@unl.edu

ABSTRACT

In this paper, we report on a recently funded project called Renaissance Computing, an initiative for promoting student participation in computing. We propose what we consider a radical rethinking not only of our core curriculum in computer science, but of the role of computer science at the university level. In our conception, "computational thinking" is neither easily separated from other endeavors nor easily balkanized into a single department. We thus imagine a CS program that is inextricably linked to other domains. Our proposed initiative covers introductory, depth, and capstone courses, targeting both CS majors and minors. It is also aimed to develop interdisciplinary courses in sciences, engineering, arts, and humanities. Furthermore, the framework embraces collaborative learning to help improve learning. This paper discusses proposed framework, its intellectual basis and planned activities.

Categories and Subject Descriptors

K.3.2 [**Computers and Education**]: Computer and Information Science Education - c*omputer science education, curriculum.*

General Terms Design, Human Factors, Measurement

Keywords Interdisciplinary, Initiative, Introductory, Capstone

1. INTRODUCTION

In the narrowest sense, "computational thinking" is the mindset that students need to acquire in order to work effectively with computational systems. More broadly, however, it is a way of understanding the world—one that transcends mere methodology and which is likewise transportable across a wide variety of human endeavors. Exploring the implications of this notion, and of its practical and theoretical applications, has always been the core subject matter of computer science. Our computational systems have changed over the decades, but the fundamental habit of mind has not. If anything, it has become more and more apparent that computer science is not about the machine.

The synopsis for the National Science Foundation's CISE Pathways to Revitalized Undergraduate Computing Education (CPATH) program notes that "despite the deep and pervasive impact of computing and the creative efforts of individuals in a small number of institutions, undergraduate computing education today often looks much as it did several decades ago." On the one hand, we regard such stasis as unproblematic, to the extent that computational thinking, as outlined above, has remained largely invariant throughout its history. However, we take very much to

heart a later statement, which declares that computing has broadened and now "require[s] integration of multidisciplinary domains." In our view, computer science curricula that do not address this broadening—which seek change through minor adjustments to the languages, systems, and engineering paradigms taught to undergraduate students—do not address the ways in which computational thinking pervades life in the modern world.

We therefore propose what we consider a radical re-thinking not only of our core curriculum in computer science, but of the role of computer science at the university level. In our conception, "computational thinking" is neither easily separated from other endeavors nor easily balkanized into a single department. We therefore imagine a computer science program that is always inextricably linked to other domains. We further understand these domains to include not only the subjects ordinarily thought of as cognate with computational thinking (like bioinformatics or computational physics), but with such notions as "humanities computing," "arts computing," and "music computing."

Our term for this new approach, "**Renaissance Computing**," is intended to evoke that period of history in which computational thinking arguably first began—a period marked by the single lack of hard and fast lines between subjects that has now overtaken university curricula. We envision an undergraduate program in which students are prepared for the challenges of any one computational domain by virtue of their exposure to multiple domains in which computational thinking plays prominent a role.

2. Intellectual Basis

2.1 How Renaissance Computing Benefits CS Majors

CS majors will have the opportunity to apply computational thinking skills to a plethora of disciplines that are traditionally close to CS such as sciences and engineering as well as those that have already become prevalent, such as arts and humanities. CS majors will be motivated to understand the challenges in those disciplines, and in turn, be "forced" to truly drill down on problem solving using their computational thinking skills in order to find a good solution. Interdisciplinary problem solving and teamwork also allow CS to bring in students of diverse backgrounds and interests, helping our CS majors to better prepare for the real world.

Further, we argue that exposing CS majors to topics in digital arts and humanities also trains them on another front. Creativity and imagination are highly valued in arts and humanities. Unlike science or engineering-oriented applications that usually adhere to strict specifications, problems in digital arts and humanities are much more open-ended and at times with moving targets. Therefore, having interdisciplinary courses such as those proposed in Section 3 will further train our CS majors with their creative thinking skills.

Finally, as the world becomes increasingly digital, CS graduates will likely encounter interdisciplinary projects in sciences, engineering, arts, and humanities in their careers. With our emphasis in interdisciplinary problem solving and collaborative learning, Renaissance Computing will better prepare our CS graduates with the skills they need to succeed in industry. Take for example the areas such as bioinformatics and computational biology, which are now providing a huge playing ground for computer scientists, where they can apply their skills to manage and exploit data. It also provides challenges to computer scientists where new algorithms need to be developed. However, collaboration between computer scientists and biologists are often hampered due to the lack of common language and background knowledge. Interdisciplinary training of CS students in early stages of their education would facilitate the removal of such barriers and broaden their career paths.

2.2 How Renaissance Computing Benefits Other Disciplines

Understanding how the proposed Renaissance Computing program can benefit other disciplines is critical. First, this understanding will allow us to build meaningful and engaging real-world applications that provide interesting and yet challenging assignments and discussions for CS majors. Second, this understanding will allow us to identify the specific computational thinking and computing skills that non-CS majors need, which in turn help us motivate them to take more CS courses and to minor in CS. To illustrate, let us look at today's work in humanities.

Today's literary scholars, historians, linguists, philosophers, and their students have access to vast text and image resources reflecting the primary objects of study in their respective fields. It is obvious that these kinds of digital resources—distinguished not merely by their comprehensiveness, but by their tractability—allow one to ask questions that have literally never been asked before, and so digital scholars in the humanities are increasingly turning toward analysis of this material, which involves subjects that are quite solidly in the realm of computer science and engineering—including data mining, ontology-based storage, agent-based simulation, statistical computing, and natural language processing. Yet, few students have the skills necessary for undertaking comprehensive analysis of these corpora as corpora, by using them to form new kinds of questions and create new kinds of tools. And to make matters worse, their inability to do so means that the potentiality of these resources—the new tools, interfaces, and modes of critical thinking they suggest—will undoubtedly be left in the hands of engineers who, understandably, may have no domain knowledge of the material in question. Our proposed framework endeavors to create specialist engineers that can be a part of the digital revolution by giving engineers a sense of the problems unique to the humanities, and by giving humanists a deeper sense of the computational. The goal is to educate a new generation of students who are adept not merely at using tools, but at creating them and understanding the nature and implication of that creation.

2.3 Why Collaborative Learning

"Collaborative learning" is learning stemming from collaboration that involves the construction of a solution that otherwise could not be produced [9]. In Renaissance Computing, we emphasize collaborative learning to yield even more significant impact in student learning of their respective discipline-specific topics and of their interdisciplinary problem solving and teamwork skills. This is especially true as students of diverse backgrounds, interests, and disciplines will encounter conflicts during the joint construction of understanding process. These conflicts could lead to improved learning [11].

Increasingly popular is the use of computer-supported collaborative learning (CSCL) systems [15] to increase the likelihood of improved learning in participants through more guided or structured collaborative learning, fueled by the advent of Web 2.0 and the renewed opportunities for innovations in teaching and learning [2] and that today's students learn in groups and in a social setting more often, and they solve problems from gathering information online more often than students of yesterday who more likely solved problems individually from information they memorized [10]. Research has shown that CSCL can provide reorganized social contexts that promote active and on-task learning, especially when increasingly more instruction and learning are performed online [16].

However, there are reported pitfalls, such as taking social interaction for granted and restricting social interaction to cognitive processes with these CSCL systems [14] and the risk of overscripting CSCL systems such that they disturb the natural collaborative interactions and problem solving processes and increase cognitive load on the use of the CSCL itself [6]. Thus, for our Renaissance Computing paradigm, the primary use of CSCL is to support and *not* to replace actual collaborative activities, to improve student learning of their discipline-specific topics, and of teamwork and interdisciplinary problem solving skills, and to systematically track and support classroom management across different courses.

3. The Basic Framework

Our basic framework of the proposed Renaissance Computing curriculum is one that revolutionizes undergraduate computing education for CS majors and minors, emphasizing interdisciplinary course contents for introductory, middle, and capstone CS courses, and collaborative learning activities. Figure 1 shows the framework for CS majors, CS minors, and non-CS majors. It includes a set of CS1 courses and a unifying CS2 course, sets of technical electives (so-called CS depth courses), and a culminating, integrative capstone course. This design also provides us with longitudinal stewardship over the evolution of these courses, including tracking and assessment of student performances and instructional content. The framework is flexible: it accommodates regular CS majors, as well as CS minors from other departments, and fulfils other majors' technical elective requirements. The capstone course also allows students of other disciplines to collaborate with our CS majors and minors in interdisciplinary group projects, further exposing the students to exciting aspects of solving interdisciplinary problems. Finally, the proposed framework is customizable by individual students, providing different pathways to suit student needs and interests, either as a CS major or a CS minor.

The framework will also incorporate collaborative learning activities with interdisciplinary projects or assignments for groups of students from different courses, allowing students of different CS1 courses to interact, and students of different levels (from freshmen to seniors) to interact. The collaborative learning environment will support student social networking and facilitate teamwork using a computer-supported collaborative learning (CSCL) system called I-MINDS. The system will also help teachers monitor and manage student groups.

3.1 The Renaissance Computing CS Courses

The CS1 "Funnel". The proposed framework has a set of CS1 courses, each tied to a different non-CS area. Each course may use a different programming language and have a unique emphasis. For example, CS1-Engineering may use C or C++ as the programming language and with assignments and examples in engineering applications, while CS1-Arts may use Python or Java as the programming language, and with assignments and examples in multimedia applications pertaining to, say, music, dance, and digital arts. CS1-Sciences will be relevant to life sciences (e.g., physics, chemistry, and biology) and natural resource sciences (e.g., geosciences and hydrology); CS1-Humanities will be relevant to history, English, and so forth. Each of these CS1 courses will contain the same basic core of CS topics (a subset of those in

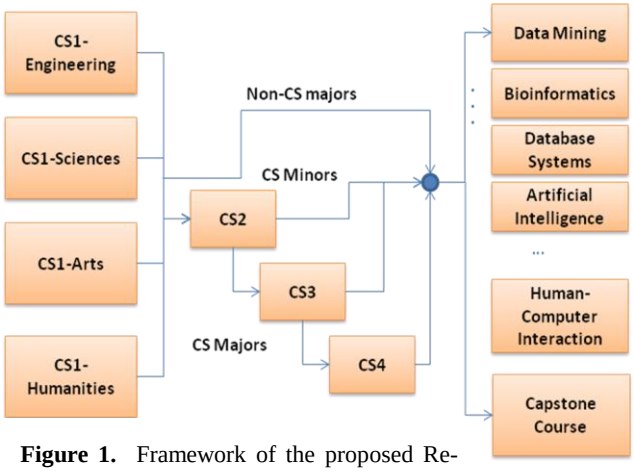

Figure 1. Framework of the proposed Renaissance Computing curriculum

IEEE/ACM Computing Curricula 2001), including the problem-solving paradigm of object-oriented programming, at least covering the concepts of abstraction, encapsulation, and the "object" view when constructing a solution. Each CS1 course will also introduce students to database design. CS1 will be required of all CS majors and minors.

CS2. The framework has one CS2 course, to cover data structures (e.g., stacks, queues, linked lists), other traditional CS2 search and sorting algorithms, and the concepts of object-oriented programming further: polymorphism and inheritance. Further, OO concepts such as event-driven programming and exception handling will be covered. CS2 will be required of all CS majors and minors.

Depth Courses. In addition to the set of depth courses already being offered (e.g., Bioinformatics, Data Mining, Artificial Intelligence, Database Systems, and Human-Computer Interaction), new topics such as simulation, computer visualization, embedded systems, and autonomic computing will be considered. These depth courses will serve as technical electives for CS majors and minors.

Capstone Course. The proposed framework requires all CS majors and minors to take a capstone, project-based course. All projects will be interdisciplinary in terms of the problem to be solved and also the team members. This capstone course will also meet an elective requirement of all majors of participating departments.

3.2 Discipline Non-CS Majors

We envision the CS1 "Funnel" courses as establishing a foundation for broader implementation of computational thinking within the academic disciplines. Within the Renaissance Computing Framework in Figure 1, we anticipate that these courses will ultimately be required of all students in their respective major fields. We also anticipate that the capstone course will become, if not required, at least a broadly-taken senior elective. As a result, Renaissance Computing will become an integral component of the University's general core academic requirements. While all students will take CS1 and most capstone, we anticipate that many of these students will take additional CS2 and depth courses to gain additional computational knowledge and skills to apply in their field, with perhaps many pursuing a CS minor.

3.3 Technology-Based Learning Platforms

We see two specific technology-based learning platforms as critical to the implementation of Renaissance Computing courses and the incorporation of collaborative learning activities. These technology platforms could facilitate course delivery at the university and provide potentially cost-effective ways to disseminate and replicate the Renaissance Computing approach at other institutions.

Learning Objects. Learning objects (LOs) are small, stand-alone, mediated, content "chunks" that can be reused in multiple instructional contexts and serve as building blocks for lessons. The value of the learning object approach has been recommended by the Department of Defense [1], business and industry [3], public schools [8,18], and higher education [7,13]. Major strengths cited were reusability, ease of updates and content management, customization, interoperability, and overall flexibility. Research on LOs has also verified their instructional value [4,5,17].

Because the CS1 "Funnel" courses share common CS core components, LOs would be a cost-effective and efficient way to share this common core curriculum across the courses. As part of this framework, we will create LOs for *selected common CS core components* that will be used across the courses to test the feasibility of this approach. We will also create LOs for *selected discipline-specific content* to determine student reactions to the LO approach for curriculum within their discipline. The flexibility of LOs makes them a potentially viable approach to supplement the delivery of Renaissance Computing courses and the ultimate dissemination of CS1 courses to other institutions.

Computer-Supported Collaborative Learning: I-MINDS. Collaborative learning is a foundational principle of Renaissance Computing. To support collaborative learning, across the courses as well as the disciplines, we will utilize a computer-supported collaborative learning (CSCL) tool: the Intelligent Multiagent Infrastructure for Distributed Systems in Education (I-MINDS) [20]. I-MINDS employs a system of intelligent software agents, representing individual students and the instructor (or teaching resource in the case of an asynchronous course or lesson). In I-MINDS, each student agent serves a student, profiling the student's behavior in his or her structured and non-structured collaborative activities [12]. The student agents exchange information to form peer groups that are compatible to help students collaborate [21]. For each group, there is a group agent that provides scaffolding and monitors the progress on accomplishing tasks assigned by the instructor. The instructor is supported by a teacher agent that displays statistics of students and groups, identifies problems in group activities (for example, a student being too dominating or too shy), and manages the Q&A sessions by ranking questions posed by the students. The teacher agent also administers quizzes and supports multithreaded forum discussions. Our educational studies show that agent-mediation can help improve student performance [19] and form more effective and efficient groups [12,21].

For this framework, we will examine how to incorporate collaborative learning into our CS1 "Funnel" courses so that students of various disciplines could interact without having to be in the same class and students of different levels (from freshmen to seniors, and CS majors to minors) can interact. To promote teamwork and collaborative learning, I-MINDS also provides a unique function: it is one of only a handful computer-supported collaborative learning (CSCL) systems that tracks and records all student activities among themselves and with the teacher through its graphical user interface. I-MINDS enables instructors to track and capture student participation in collaborative activities, allowing, for example, researchers to study student pedagogy of students of diverse backgrounds. Since student participation is highly accountable through I-MINDS, instructors would be able to better motivate students to collaborate, which is important for our Renaissance Computing framework.

4. PLANNED ACTIVITIES

4.1 Year 1: Planning and Development Phase

We will initiate planning activities with a one-day "Renaissance Computing: Curriculum Planning" (RC-CP) Workshop. The objective of this workshop will be to develop a planning framework for identifying course content. In addition to the project PIs, other faculty from Computer Science and the participating disciplines will be invited. At the workshop, work groups will be formed for each of the disciplines: sciences, engineering, arts, and humanities). At the workshop, initial planning discussions in the work groups will focus on three specific threads: (1) aspects of computational thinking that expand the problem solving skills of students in non-CS majors, (b) aspects of interdisciplinary problem solving that better engage potential CS majors and prepare CS majors for a career in computing, and (c) aspects of collaborative learning that affect student motivation, aptitudes, and attitudes towards working with students in other disciplines.

Following the RC-CP workshop, each work group will regularly meet to develop the specific content of its respective CS1 course. This includes identification of computational topics, core computational knowledge and tools, interdisciplinary problems and collaborative problem-based learning (PBL) activities, and additional content.

We will fully develop several CS1 courses (e.g., CS1 Humanities and CS1 Sciences) for implementation in the pilot study. For these courses, we will complete full implementation of content topics, PBL activities, and collaborative activities. During this phase, we will employ a graduate student to assist with the creation of the specific LOs for these courses. These will be SCORM-compliant to allow maximum portability. We will develop common LOs for the two courses where possible. We will set up I-MINDS for collaborative learning activities in these pilot courses.

4.2 Year 2: Pilot Study Phase

The pilot courses will be offered during the first semester of Year 2, with students recruited primarily from our freshmen CS majors and minors and students from the School of Biological Sciences, the Departments of History and English, and secondarily from other participating academic units. Students participating in these courses and the instructor will be asked to consent to participation in research and evaluation activities in accordance with Institutional Review Board (IRB) guidelines. We will collect data on student outcomes and technical aspects of course implementation. For student outcomes we will use pre- and post-tests to assess students' knowledge and skills in computational thinking and tool use, attitudes and motivation about computational thinking and CS, attitudes and motivation about interdisciplinary collaboration, impact of the course on students' self-regulation, and students' perception of the classroom environment. In addition, we will embed specific assessments within the LOs. These will capture both student learning and immediate student attitudinal and motivational reactions to the LO. At the end of the course, we will conduct a short on-line interview with students to gather their self-assessment of their learning, motivation, and experience in the course.

Feedback on technical aspects of course implementation will be obtained from students and the instructor using an end-of-the-course on-line interview. Students and the instructor will be asked to provide feedback on satisfaction, course effectiveness, and suggestions for course improvements. They will also be asked to rate the utility and interestingness of specific course components. We will also capture data from interaction with the LOs and I-MINDS to identify how students were interacting with these learning technologies. Instructors also will be provided with an on-line diary to record immediate reactions and comments concerning the course.

5. ACKNOWLEDGMENTS

This material is based upon work supported by the National Science Foundation under Grants No. 0632642 and 0829647.

6. REFERENCES

[1] ADL, Advanced Distributed Learning. (2003). *DoD affirms SCORM's role in training transformation.* Retrieved July 18, 2005, from www.adlnet.org

[2] Alexander, B. (2006). Web 2.0: A New Wave of Innovation for Teaching and Learning?, *EDUCAUSE Review*, 41(2), 32-44.

[3] ASTD. (2000). A Primer on Learning Objects. *Learning circuits: ASTD's online magazine about E-learning*. http://learning circuits.org/mar2000/primer.html

[4] Boster, F. J., Meyer, G. S., Roberto, A. J., & Inge, C. C. (2002). *A report on the effect of the UnitedStreaming Application on educational performance.* Cometrika/United Learning.

[5] Bradley, C., & Boyle, T. (2003). *The development and deployment of multimedia learning objects.* Available at http://www.cs.kuleuven.ac.be/~erikd/PRES/2003/LO2003/

[6] Dillenbourg, P. (2002). Over-Scripting CSCL: The Risks of Blending Collaborative Learning with Instructional Design. In P. A. Kirschner (Ed.), *Three worlds of CSCL: Can we support CSCL* (pp. 61–91). Heerlen: Open Universiteit Nederland.

[7] Francia, G. (2003). A Tale of Two Learning Objects. *J. of Educational Technology Systems*, 3, 177-190.

[8] Grunwald Associates. (2002). *Video and television use among K-12 teachers*. Survey results in Power Point format prepared for the Corporation for Public Broadcasting (CPB).

[9] Hansen, T., L. Dirckinck-Holmfeld, R. Lewis, and J. Rogelj (1999). Using Telematics for Collaborative Knowledge Construction, Chapter 9, in P. Dillenbourg (ed.) *Collaborative Learning: Cognitive and Computational Approaches*, Oxford: Elsevier, pp. 169-195.

[10] Hartman, J. L., C. Dziuban, and J. Brophy-Ellison (2007). Faculty 2.0, *EDUCAUSE Review*, 42(5), 62-77.

[11] Howe, C., A. Tolmie, A. Anderson, and M. Mackenzie (1992). Conceptual Knowledge in Physics: The Role of Group Interaction in Computer-Supported Teaching, *Learning and Instruction*, 2(3), 161-183.

[12] Khandaker, N., L.-K. Soh, and H. Jiang (2006). Student Learning and Team Formation in a Structured CSCL Environment, *Proc. ICCE'2006* (pp. 185-192), Beijing, China.

[13] Koppi, T., & Lavitt, N. (2003). *Institutional use of learning objects three years on: Lessons learned and future direction.* Retrieved October 10, 2004, from http://www.cs.kuleuven.ac.be/~erikd/PRES/2003/LO2003/

[14] Kreijns, K., P. A. Kirschner, and W. Jochyems (2003). Identifying the Pitfalls for Social Interaction in Computer-Supported Collaborative Learning Environments: A Review of the Research, *Computers in Human Behavior*, 19(3), 335-353.

[15] Lehtinen, E., K. Harkkarainen, L. Lipponen, M. Rahikainen, and H. Muukkonen (2001). Computer Supported Collaborative Learning: A Review. Retrieved March 5, 2008, from http://www.comlab.hut.fi/opetus/205/etatehtava1.pdf

[16] MacDonald, J. (2003). Assessing Online Collaborative Learning: Process and Product, *Computers and Education*, 40(4), 377-391.

[17] Nugent, G., L.-K. Soh, and A. Samal (2006a). Design, Development and Validation of Learning Objects, *J. Educational Technology Systems*, 34(3), 271-281.

[18] Pasnik, S., & Keisch, D. (2003). *Teachers' Domain Evaluation Report, NY: Center for Children and Technology*. Retrieved October 10, 2004, from http://www2.edc.org/CCT/publications_report_summary.asp?numPubId=148

[19] Soh, L.-K., H. Jiang, and C. Ansorge (2004). Agent-Based Cooperative Learning: A Proof-of-Concept Experiment. *Proc. SIGCSE'2004* (pp. 368-372), Norfolk, VA.

[20] Soh, L.-K., N. Khandaker, and H. Jiang (2008). I-MINDS: A Multiagent System for Intelligent Computer-Supported Cooperative Learning and Classroom Management, *Int. J. AI in Education*, 18(2), 119-151.

[21] Soh, L.-K., N. Khandaker, and H. Jiang (2006). Multiagent Coalition Formation for Computer-Supported Cooperative Learning *Proc. IAAI'2006* (pp.1844-1851), Boston, MA.