Cooperative Expertise For Multidisciplinary Computing

Ursula Wolz
The College of New Jersey
Dept. of Computer Science
Ewing, NJ 08528
+1 609 771 2766
wolz@tcnj.edu

Lillian (Boots) Cassel
Villanova University
Dept. of Computing Sciences
Villanova, PA 19085
+1 610 519 7341
cassel@acm.org

Thomas Way
Villanova University
Dept of Computing Sciences
Villanova, PA 19085
+1 610 519 5033
Thomas.Way@villanova.edu

Kim Pearson
The College of New Jersey
Dept of English
Ewing, NJ 08528
+1 609 771 2692
kpearson@tcnj.edu

ABSTRACT
As the need for multidisciplinary computing education continues to increase, consideration for distributed expertise will become critical to implementing a successful curriculum. A model of *cooperative expertise* is presented in which faculty maintain responsibility for their own course, creating and evaluating assignments for their students that support learning in their colleagues' courses as well. We present outcomes of an experiment to implement this model at two geographically separated institutions through three courses (two at one institution, one at the other), by faculty in computer science, media and English. Results reported include faculty analysis of student achievement in each course and student surveys of attitudes toward multidisciplinary collaboration. Overall, student learning and attitudes are enhanced by the collaborative experience.

Categories and Subject Descriptors
K.3.2 [**Computing Milieux**] Computer and Information Science Education
K. 4.0 Computers and Society

General Terms
Human Factors

Keywords
K-12 CS Education, Multidisciplinary Computing, Writing and Computing, Distributed Expertise, Cooperative Expertise.

1. INTRODUCTION
This paper describes an experience undertaken as part of our NSF CPATH funded project on Distributed Expertise. That concept evolved from an earlier project, funded through NSF CCLI, to investigate the potential for collaboration in bringing enhanced expertise to teaching computer science [1,2]. The premise, confirmed by our research was that many computer science departments do not have the human resources sufficient to adequately teach every area of computer science, unrealistically stretching faculty expertise and resources. We found that instructors cope by emphasizing the part of the subject with which they are most comfortable for them and skipping those for which they lacked confidence. We received enthusiastic support by CS faculty for potential collaboration with subject area experts who could augment local resources. However, there were challenges involved that made direct collaboration impractical at the time. We were able to explore resource sharing in CITIDEL [3,4] and Ensemble [5,6] but were unable to fully address collaborative teaching models.

In the ensuing decade, the computing education landscape has changed, yet these needs remain the same. While the number of students who pursue a computing major has declined, the breadth of computing has increased, as had the need to expose a larger constituency to computing concepts and skills. Perhaps most importantly, we in the computing disciplines have begun to recognize the contributions other disciplines offer to us.

The project reported on here explores the potential for collaborative, cooperative teaching. It has a number of interesting components. Faculty participants come from different educational institutions and from different disciplines. Students include computer science majors, non-computing majors taking a computing course, and students in an interactive storytelling course. How these disparate groups interacted and built on each other's abilities gives us insights into creating collaborative efforts among faculty and among students. The results presented suggest that disciplinary boundaries do not have to limit learning and workforce potential.

2. COOPERATIVE EXPERTISE
Cooperative Expertise is a collaborative teaching model from a set of three Distributed Expertise models in which faculty and students in distinct courses identify and share expertise with faculty and students in other courses where that expertise can be beneficial. Ideally, collaborative expertise is a two-way collaborative model, where all involved benefit both from learning from the expertise of others and from the experience of sharing expertise with others.

A key concern to facilitating mutually supportive collaboration is balancing individual responsibility with dependence on resources. Ultimately, within the constraints of mainstream higher education, accountability for instruction and assessment must rest with the instructor of record. This is particularly critical for classroom instruction where outside forces (e.g. institutional expectations) may not completely embrace a collaborative cooperative model. An instruction takes responsibility for the course syllabus. A student takes responsibility for completing assignments. Cooperative class expectations for both the instructor and student must provide "fault tolerance," that is, fall-back mechanisms for resource failure – when an instructor or student on whom your work relies, fails to deliver in the expected manner.

To address this in the collaborative model, based on experience from the previous year, we designed curriculum for three co-dependent courses, where each faculty member was fully responsible for just one course. Furthermore, assignments were designed to provide enrichment for the other courses, but no assignment was created that depended upon deliverables from students outside the immediate course. Two face-to-face meetings of the collaborating faculty members, prior to the start of the semester, provided an opportunity to negotiate where, when and how student assignments could support learning between courses.

Three design deliverables provided the focus for the courses. To varying degrees all three courses contributed to design documents and prototype implementations of the following:

- A sprite editor for producing 2-D animation strips.
- A story engine for interactive storytelling.
- A two player game engine for a mobile device.

Cooperative Expertise is a collaborative teaching model from a set of three Distributed Expertise models where faculty and students in distinct courses identify and share expertise with faculty and students in other courses where that expertise can be beneficial. Ideally, collaborative expertise is a two-way collaborative model, where all involved benefit both from learning from the expertise of others and from the experience of sharing expertise with others.

2.1 Models of Distributed Expertise

We have identified three models of Distributed Expertise that define the center and endpoints of a spectrum of collaborative approaches to sharing expertise in education [7]. The models are: Remote Expert with Local Facilitator (RE), where faculty expertise resides at one institution and benefits faculty and students at another institution; the Cooperative Experts (CE), where faculty at two collaborating institutions exchange mutually beneficial expertise with the other; and, the Special Resource (SR) model, where expertise is shared in more of a one-time or guest instructor format.

2.2 Multidisciplinary Computing

The TCNJ faculty members have been active participants in multi-disciplinary computing experiences for almost a decade[8,9] A two semester upper level experience in game design and implementation continues to provide the anchor. A firm principle of this experience is that each student brings personal expertise to the enterprise as well as highly personalized learning goals within the framework of broad course objectives. One of those objectives is for students to demonstrate through deliverables and reflective essays how diverse expertise contributes to the software enterprise. In past years, we have drawn on assignments from other courses including animation, artificial intelligence and interaction design. The deliverables, however, were defined entirely within the games course, with individual students choosing to do major projects based on specifications defined by the games students to augment the game implementation. The work done outside the course was never mission critical. This was not by instructor design, but has emerged as a significant component of the course.

Last year, the authors created a cooperative experience between our two institutions between the second semester games class at TCNJ and an introductory game design class at Villanova which was open to all majors. Our primary focus was to "outsource" technical development from the games course at TCNJ to the more experienced computer science students in the introductory games course. This was a very different model from prior experiences at TCNJ because the outsourced programming work was expected to be mission critical. Results reported in detail in [7] confirm that students learned to appreciate multidisciplinary cooperation. However, the TCNJ games class deliverables were completed without any significant contribution from the Villanova students. As a means of self-protection (in order to deliver prototypes) the games students gradually moved the outsourced contributions to the background and augmentation status we had experienced in prior semesters.

2.3 Responsibility and Accountability

The mixed success of the prior year led the instructors to analyze how to support better cooperation and collaboration between students. This is a problem in any partnered approach to technology skills development. Because the US undergraduate experience is entrenched in an individualistic accountability tradition (e.g. grades), there is an inherent tension between the goals of collaboration and accountability. While industry is searching for computing expertise that includes "plays well with others", the pedagogy of the traditional classroom, especially across institutions is antithetical to this goal.

Our proposed solution was to compromise on sharing in the interest of student self-protection. The structure of our cooperative experiment held to the following:

1) Each assignment would be assessed by the faculty of record for the course in which the student was enrolled, using rubrics and criteria solely the responsibility of that instructor.

2) Assignments in a particular class, whether to the group or an individual had to have a clearly defined objective related to the course goals of that course (e.g. no busy work, much less to support another course.)

3) Deliverables assigned in one course should not be mission critical to deliverables in another course, that is, a student's ability to complete an assignment should not be dependent upon the work of a student in another course.

In retrospect these principles seem obvious, and can be applied in a single cooperative classroom as well. However, despite our own extensive experience with multidisciplinary computing classrooms, we did not fully appreciate this until we were immersed in the cross-institutional experience.

3. Cooperative Curriculum

The Game Implementation course at TCNJ is a spring semester experience in video game development that focuses on project management and software implementation. It follows a fall course in which game engine architecture and game design are emphasized through the development of a robust design document that includes a game story, interface design, media prototyping (e.g. graphics, sound and music.) This upper level course with permission of instructor prerequisite is offered as a cross-listed course in Computer Science and Interactive Multimedia attracting students from majors across campus

The Software Engineering course offered at Villanova University is taken in the second semester of the junior year, and is a software specification and design course that provides students with a theoretical and practical background in the topic while preparing students to apply their educational experiences to their career.

Interactive Storytelling (IS) is an advanced class in the Professional Writing focus area of the Interactive Multimedia major at The College of New Jersey. The class requires students to consider the ways in which the use of interactive technologies and media create new possibilities and challenges for storytellers. Students create and critique stories that aspire to blend the principles of narrative theory with elegant software design.

3.1 Course Objectives

The principal Cooperative Expertise objective for students in Software Engineering simulate real-world software engineering practices of software design, specification, consultation, evaluation and testing through practical and collaborative exercises that benefit students in the other two courses. A secondary objective is to expose students to faculty and student expertise in game design and interactive storytelling, to place software engineering concepts within the context of concrete applications.

As a self-contained unit, the Games Implementation course at TCNJ focuses on multidisciplinary cooperation and highly individualized learning to accomplish a technical task. Students learn project management via an immersive experience. A major focus of this class is developing an appreciation for the complex skill set required to produce a highly interactive piece of computing software. The principle objective for with regard to cooperative expertise was to provide students with insight into distance communication in the project cycle.

The learning goals for the Interactive Storytelling course expect students to identify and classify interactive stories, both in terms of their interactive features and their narrative structures. Students are expected to analyze and critique the technical and design features of interactive narratives on the basis of their contributions to storytelling, and examine and apply existing interactive multimedia tools to the creation of interactive narratives. Students are also expected to discuss and hypothesize ways in which this genre of storytelling might evolve.. They should be able to understand and make reasoned arguments about the social, cultural and ethical value of interactive storytelling techniques and artifacts. As such, this course was expected to provide realistic "clients" for the software engineering course, and game designer expertise for the games course.

3.2 Enrichment Assignments

Specific enrichment activities were assigned to students in the collaborating courses to provide all students involved with opportunities to support the learning of students in other course while applying the skills and experience learned in their own courses. These enrichment assignments were broken down as follows.

In support of Games by Software Engineering:

- Design evaluation of a sprite editor software specification by the Software Engineering class, with an offer to a student team to take on implementation as a semester project.
- Specific SE student team project to design and implement a prototype sprite editor tool, which included a software design and implementation phase, in-class demonstration of prototypes, and comparison with needs of Games course.
- Consulting on iPhone programming by SE students, primarily identifying student expertise, although this available resourse was not used by Games students.
- In-class discussion of distributed collaboration on game design projects, how the process would work, and how it works in industry.

In Support of Games by Interactive Storytelling:

- Provide literary analysis of the game story, highlighting concepts of genre, narrative structure and story consistency.
- Via sample stories developed in the Wolz story engine, articulate relationship between story engines and game engine logic system. Provide insight into data structures to support complex story.
- Contribute to discussions of level elaboration and overall story consistency.

In support of Interactive Storytelling by Software Engineering:

- Interactive story evaluation of four stories from usability and software design perspectives, include code review when available, students gained experience with Flash, Scratch and Processing.
- Evaluation of existing free and open source interactive storytelling engines, which was done by one student, though outcome was weak.
- Collaboration on a software requirements and design specification of an interactive storytelling engine.
- In-class discussion of distributed collaboration on interactive storytelling projects, to motivate active learning activities.

In support of Interactive Storytelling by Games:

- Provide interactive story for a two person game for literary analysis.
- Provide consulting services via code review with regard to logic implementation of assigned interactive stories (e.g. provide technical expertise in implementing stories within

the Wolz story engine prototype.) Analyze shortcomings of Wolz story engine based on experience with game story.

- Provide real-world opportunities for interactive storytelling students to elaborate on level concepts (e.g. flesh out interactivity and narrative in the game levels.)

In support of Software Engineering by Games:

- Act as client for development of sprite editor, research existing technology and what software needs to be adapted and enhances.
- Contribute design perspective to story engine development based on implementation of game story to both console and mobile device.
- Create design specifications for data communication needs for two player cross-platform game (web-based Java applet and iPhone), soliciting technical expertise from SE class.

In support of Software Engineering by Interactive Storytelling

- Provide client perspective for storytelling engine.
- Provide opportunity for code critique on engine.
- Provide feedback on sprite editor usability for storytelling.

4. RESEARCH METHODOLOGY

This is what we set out to do:
***Objective**: To demonstrate how four courses, each with its own learning outcomes can be coordinated to enhance learning and support faculty expertise in a multidisciplinary manner through the development of three deliverables: an iPhone game (based on a design developed in Games I at TCNJ), a Sprite editor, and a Storytelling Engine.*

Please note that fairly early on we abandoned including one course taught at Villanova, a non-major computing course. Discussion of this occurs in Section 5.

***Methodology:** Three faculty from two institutions will coordinate course activities among four classes: Wolz, Games II at TCNJ; Pearson, Interactive Storytelling at TCNJ, Way, Software Engineering at Villanova; Way, Digital Media at Villanova. Course activities will require individual students to coordinate with peers in other classes to complete development of the three deliverables. Expertise sharing will be documented, with the expectation that it will be reciprocal, that is, no class will act solely as client, consultant or developer.*

4.1 Report on Deliverables

Outcomes were both expected and surprising. The cross-collaboration took the responsibility for directing deliverables development away from the instructors as expected, however the anticipated path was influenced by the “real world” roles each class took on as client.

A sprite editor design was produced, however, the focus of the ultimate design, influenced heavily by the animators in the games course, was significantly different from that initially conceived by the software engineering students. Whereas the Villanova students wanted to focus on cutting up a sprite sheet (a document that holds all of the possible positions of a sprite) into individual animation cells, the games animators needed the inverse system: a way to take individual drawings created in Photoshop and merge them into a Sprite sheet. Low level specification of layout on the sheet, agreement on number of drawings per animation, granularity of animation and a host of other details with which animators are intimately acquainted were communicated to the design team at Villanova.

A prototype story engine, created by Wolz was adopted by the Interactive Story telling students, some of whom developed programming expertise as they personally enhanced the engine at code level. The games class did considerable analysis of how to take a story narrative and translate it into a data structure and logic. A student in the games course led an effort by the software engineering students to encompass exploitation of the engine to support game design. Thus students in all three classes were engaged in the conversations about the boundaries between interactive story and game design. While this project was not a major focus in any of the three classes with regard to explicit deliverables, it became a point of departure for curricular issues in all three classes: in Software Engineering regarding assumptions about design specifications, in Interactive Storytelling regarding linear versus interactive storytelling, and in Games regarding the implentatnion of a story concept as a logic and data structure.

Contrary to expectations for “outsourcing” and “expert consulting” the game course students were unable to derive much technical expertise on iPhone programming from the software engineering class. Instead they were very actively dependent upon the interactive storytelling students to contribute to developing a plausible game. In retrospect this reaffirms some of the social dynamics of the previous year when the SE class was considered an outsourcing site. Both years the SE students did not have ownership of the implementation development – they were not responsible for a particular component or block of code. The lag time in contacting them, as well as the lack of establishing trust over distance meant that the Games students relied on each other in face-to-face rather than try to distance communication technology. We posit that the heavy influence of undergraduate culture came into play here. Despite herculean efforts on the part of the instructors to forestall it, the students all still did major work at the last moment, typically through the night prior to the next class meeting. The reality of the undergraduate experience trumped our best efforts.

The major lesson learned by the instructors was to support a pedagogy akin to agile design: exploit resources as need emerges. Don’t over-commit to teacher-driven concept learning, but focus on the holistic themes of the course goals. Survey results reported in section 5 confirm the success of this approach.

4.2 Impact on Student Content Learning

Based on student final project work, final exam results and observations in the classes, all three instructors are confident in saying that student met the learning goals for each course. As summarized above, in a number of instances, the collaboration provided a vehicle for “real world” experience.

We posit that this sort of collaboration is perhaps more realistic than connecting with an industrial or non-profit client. Under those circumstances there is very little wiggle room if the design and implementation process goes wrong. The artificial timeframe of a semester course does not provide the wiggle room of delayed roll-out that can occur in an industrial setting. Furthermore, the students are learning skills while implementing them, complicating milestone predictions.

We further assert that the collaborative approach improves upon the industrial client model because students in all three classes have an opportunity to experience both the client and developer side of the process. This was particularly important in the interactive storytelling course. These students programmed in a version of Java called Processing. They delivered product that was critiqued by the technical classes. Such experiences will become increasingly important in a 21st century liberal education.

4.3 Impact on Student Soft Skills Learning

Surveys were administered at the end of the semester. Twenty two Villanova students responded, twenty five at TCNJ (19 in Games, 6 in Storytelling.) Table 1 confirms our objectives that students will gain from the cross-collaborative experience. Table 2 provides detail how students at each institution felt about their learning. Overall students were positive about the experience.

5. ANALYSIS AND SUMMARY

The surveys in particular indicate that the Villanova students definitely gained from this experience. The TCNJ students, many of whom had experienced multidisciplinary perspectives via courses in the interactive multimedia major were still positive (> 50%) but less inclined to view the contributions of the Villanova experience as value added. With regard to soft skills, the students' perspective on learning is primarily positive. The complexity of collaboration provided invaluable experience to all of the students and the faculty.

Traditional undergraduate education is lecture-based, and especially in STEM fields student often work on projects with well-known solutions. Major universities can provide more collaborative exposure to the "real world" of collaborative product development and research. As we focus on multidisciplinary studies we have an opportunity to create models of pedagogy that exploit local and remote resources in a manner that provides a less hierarchical, teacher-centered learning experience. Especially in a field as dynamic as computer science this is critical.

Our collaboration this past year has been a pleasure for the instructors and a true learning experience for our students.

•

6. ACKNOWLEDGMENTS

This work is supported by NSF CISE IS-0829616 and CPATH 0935942 Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the National Science Foundation.

7. REFERENCES

[1] Cassel, L.N. and D. Kumar. A state of the course report: computer organization & architecture. in 7th annual conference on innovation and technology in computer science education. 2002. Aarhus, Denmark: ACM Press.

[2] Cassel, L.N., G. Davies, and D. Kumar, The Shape of an Evolving Discipline, in Informatics Curricula and Teaching Methods, L.N. Cassel and R. Reis, Editors. 2002, Kluwer Academic Publishers: Florianópolis Brazil,. p. 131-138.

[3] Cassel, L.N., et al. CITIDEL: The Computer and Information Technology Interactive Digital Education Library. 2002 [cited 2004 February 8]; Available from: http://www.citidel.org

[4] Cassel, L.N., et al. Report on the NSF major educational funding initiative for a National Science, Technology, Engineering, and Mathematics Education Digital Library (NSDL) with special emphasis on the Computing Education component. in 34th SIGCSE technical symposium on computer science education. 2003. Reno, Nevada: ACM Press, New York, NY.

[5] Carpenter, B.S., et al. Multiple sources with multiple portals: a demonstration of the ensemble computing portal in second life. in 10th Annual Joint Conference on Digital Libraries (JCDL). 2010. Gold Coast, Queensland, Australia.

[6] Fox, E.A., et al. Ensemble PDP-8: eight principles for distributed portals. in 10th Annual Joint Conference on Digital Libraries (JCDL). 2010. Gold Coast, Queensland, Australia: ACM.

[7] Way, T., et al., A Distributed Expertise Model for Teaching Computing Across Disciplines and Institutions., in The 2010 International Conference on Frontiers in Education: Computer Science and Computer Engineering 2010: Las Vegas, Nevada.

[8] Wolz U, C. Ault and T. M. Nakra, "Teaching Game Design through Cross-Disciplinary Content and Individualized Student Deliverables," The Journal of Game Development, 2(1), Feb. 2007.

[9] Wolz U, and S. M. Pulimood, "An integrated approach to project management through classic CS III and video game development," SIGCSE Technical Symposium (SIGCSE 2007), pp. 322-326, 2007