Innovative Uses of Classroom Tools & Technologies to Foster Students’ Learning

Dr. MD B. Sarder, University of Southern Mississippi

Dr. Sarder is an associate professor and program coordinator of the industrial engineering technology program at the University of Southern Mississippi (USM). He is also an assistant director of the center for logistics, trade and transportation. At the USM, he revamped his program by developing as many as fourteen new courses, implementing hands on experience in courses, and delivering online courses for distant students. Dr. Sarder is very active in engineering and technology education research. He has published a book and more than fifty articles in various areas of industrial engineering including K-12 research. He is actively involved with professional society activities including IIE and ASEE. He is the editor in chief of the International Journal of Logistics & Transportation Research and serving in the editorial board for several other journals.
Innovative Uses of Classroom Tools & Technologies to Foster Students’ Learning

Abstract

According to many researchers, students learn best when they are actively engaged in the learning process. The more opportunities provided for students to participate and be involved, the better they ought to achieve from the course. In many cases, professors need to go beyond the institutional and technological barriers to deliver the best of class education. Many professors frequently use multimedia application such as PowerPoint, graphics, audio, video & animation in courses to promote student involvement in their learning. PowerPoint supports hyperlinks and sound features that can easily create an interactive presentation that provides opportunities for students to employ higher cognitive strategies. Digital video and animation are invaluable tool for demonstrating processes such as automation. This paper highlights some uses of classroom tools and technology in a particular institution which were proven effective.

Nowadays online course delivery is a popular concept, but it has some issues with engaging students. There are some very effective common tools that can be used in online classes to improve students’ engagement. It is very unlikely that a single tool or technique is enough for online learning. Creative uses of a combination of tools serve better in this case. Some of these tools are: email, discussion boards, application sharing, selective release, podcasting, vodcasting, and mass texting. Email is a teacher's best friend. Students should be encouraged to use email and discussion board extensively. They need to be monitored for their discussions on course management site continually and guided constructively. Discussions generate ideas, help create a learning community in online classes, provide discussion transcripts, provide a means of online conference and collaboration, and get students thinking in writing as they write. It is shown that archived lectures with the PowerPoint slides helps student the most. Competitive group projects are common course requirements. Technology such as selective release can create work environment so that students within the group use their specific room as virtual communication platform. It makes easy for students to do brain storming and share files with their group members without revealing information to other groups.
Introduction

As students learning is widely accepted as key metric of student success, increased attention is being paid to the tools and techniques best suited to its successful adoption in classroom environment. It is very evident that technology seems to offer a natural and accessible way to advance students learning [1]. ECAR has surveyed undergraduate students annually since 2004 about technology in higher education and found that more and more students are using latest education technology in their education [2]. On an average student to computer ratio of 4:1 and a teacher and student population ready, willing and able to use technology (Figure 1). Yet despite its availability, technology is not widely integrated into the learning experience. A recent survey of ECAR shows that they wish that their professors more often use classroom technology in online or face to face teaching (Figure 2).

Figure 1: Students’ ownership of education related tech devices (a) and their importance rank to do with mobile devices (b) [2]
Figure 2: Student wished that their professors communicated more using these means [2]

Classroom technology is both highly customizable and intrinsically motivating to students, it is particularly well-suited to expand the learning experience [3]. Besides innovative uses of technology in classroom, organized teaching & rich course content is very important to students’ learning. In addition to rich course content & organized teaching, a good teacher needs to personalize the needs and problems of the students. As a teacher, it is our goal to inspire students & enhance their learning. Our objective as a teacher is to motivate students toward a level of independence where they develop a desire to learn and think for themselves. Professors should try to maintain a very lively and interactive environment by using a variety of teaching methods. They should encourage students to voice their ideas about the subject matter and to ask questions. They also need to challenge students to broaden their minds and go beyond specific course objectives.

If instructors are not properly trained in online delivery, methodologies and effective use of e-Learning tools, the success of the online program will be compromised. An online program will be weakened if its facilitators are not adequately prepared to function in the Virtual Classroom.
To keep these in mind, professors at our institution use variety of tools & technologies in an effective way to enhance student learning outcomes. Some of these tools and techniques are Adobe, PowerPoint, Podcast, Vodcast, Live lecture, mass emails, mass texting, Microsoft office live workspace, Wiggio, WebCT, and Wimba. Professors at our institution strive very hard to keep their knowledge up to date and explore new ways of effective and efficient teaching by learning.

Technological advancement in classroom equipment provides an edge to teaching. This brings flexibility to both teachers and the students. Audio-visual aids like using power point slides, laser technology, video clips to emphasize important points, and using WebCT, Wimba are effective learning tools with proven impacts. Due to the availability of these technologies, more students are able to take courses via online. In our online courses, we made the course curriculum suitable for online students. Students were allowed to view the lectures, performed group projects, appeared quizzes and exams, joined group discussions, etc. via chat, teleconferences, video conferences and other online means. Following sections described some effective & efficient use of e-learning tools specific to our institution.

Need for improved engagement in online classes

Classroom tools and technologies are means to engage students in online classes and hence improve students’ learning. In 2008, a survey conducted by the U.S. Department of Education showed that 97% of 2-year and 89% of 4-year public institutions offer distance-learning courses [4]. Also, according to new research recently released by the University of Wisconsin-Madison involving about 7,500 undergraduate and graduate students, an overwhelming 82% of students said they would prefer courses that utilize online lectures over traditional classes that do not include an online lecture component [5]. As more courses in higher education move to an online format [6, 7], a major concern is a potential loss of personal interaction between the professor and student [8]. There is evidence that a growing number of courses delivered in an online format tend to be configured and delivered in an asynchronous manner, more often associated with traditional independent study and correspondence work (i.e., students work independently to complete posted assignments at their own pace) [9]. While this format serves the purpose of meeting the needs of the non-traditional learner in regard to delimiting issues of time and
distance, and in many instances is a viable option, it leaves a "missing link" in the learning curve for students because they lack the opportunity to benefit from the experience of structured dialogue, interaction with faculty and peers, and the sense of community that can be created in a traditional on-site classroom environment. As Berge states, "…learning involves two types of interaction: interaction with content and interpersonal interaction (i.e., interaction with other people)" (p. 22[10]). Kearsley and Lynch contend that online courses must adopt a pedagogical framework more closely aligned with social learning theory for students to maximize the benefits of online instruction [6].

Design Methodology

One of the most important elements of planning and managing online courses is that there are lot of tools and techniques available, but not all of these technologies are appropriate matches to the subject taught and the teacher’s pedagogical style and strategies. As such, it is very important that instructors ensure that pedagogical principles drive the use of technology rather than the other way around. Instructors must strive to achieve certain learning standards, regardless of the medium through which they are teaching [11]. Because of this, course planning should take place before instructors select the technology and course management system that will be used for the course.

It is also important to note, that although there is tremendous diversity in the educational technologies available to online instructors, the field of distance learning technology is changing quickly, and it is therefore necessary for instructors and administrators to keep a close eye on emerging trends and associated best practices [12]. The first step in the planning process involves the development of learning objectives. The importance of learning objective development and communication is highlighted throughout the literature, including Park University’s guidelines for the creation of learning objectives [13].

In our effort to increase student interaction and students’ learning within an online course delivery system, whether the course is entirely online or being offered face-to-face augmented by online support, we planned to incorporate various tools and techniques. We found some tools
and techniques are more effective than others. The following sections highlights some implementations. We are still evaluating the effectiveness of these tools and hence a complete assessment is not presented in this paper.

**Innovations in Curriculum Design**

We implemented a multi-tiered online delivery (MOD) system. The first tier consists of in-lecture activities (Lecture Recording Tier). The second is an offline modification of lectures by importing quizzes into the lecture as well as extracting events (Offline Lecture Splicing Tier). The third tier of the application is available to the students and it increases their interaction with the instructors (Student Interaction Tier). The MOD has two features. The first feature was based on a question/answer repository (database) related to material covered in specific lectures. This repository works in conjunction with the recorded lectures to serve as an interactive feedback mechanism to ensure proper viewing as well as improve understanding of the lecture material. The second feature was to extract specific events from each recorded lecture. These events correspond to interactions between students and instructor in a live lecture setting. The benefits of extracting these interactions will reflect on current and future students. Furthermore, it will positively impact the training of future teachers of the subject matter.

**Educating Educators**

If instructors are not properly trained in alternate delivery, methodologies and effective use of e-learning tools, the success of the online program will be compromised. An online program will be weakened if its facilitators are not adequately prepared to function in the Virtual Classroom. Keeping this in mind, we developed an “Educating Educators” program at USM for faculty members of IET and CS programs. Following are the components of the program:

1. **Summer Workshops**

A summer workshop was designed for online faculty development that will show faculty how to use MOD architecture and other online teaching tools in teaching alternate delivery classes. Our primary goal here is to encourage faculty in any discipline to engage students in the class
discussion. We outline the structure of our ten-day workshops (1 day per week for 10 weeks), specify the video conferencing software (Blackboard Collaborate or similar ones) the workshop participants use to connect remotely, discuss the technology to be implemented, and explain how the new methods works pedagogically. During each session we demonstrated and discuss specific online tools and their uses to improve interactivity in the class, and at the same time, we incorporated participant feedback and contributions. Our Learning Enhancement Center (LEC) was used to coordinate and organize this virtual workshop.

ii. Peer Mentoring

Peer mentoring at a university typically involves support and guidance for junior faculty or faculty who are new to alternate delivery from more experienced faculty, often within the same discipline. It is an important strategy for assisting relatively inexperienced faculty during their transition to alternate teaching. It is an effective program for faculty development and retention. Many institutions such as Florida State University, University of Iowa and Monash University have implemented peer-mentoring programs to increase faculty competencies. We formed a group of volunteer peer mentors who are experienced and effective in alternate delivery and willing to provide the mentoring service to their fellow faculty.

Innovation and Uniqueness of our Approach

i. Real Time Communication with Students

An online instructor must be able to compensate for the lack of a physical presence by creating a supportive environment in the Virtual Classroom where all students feel comfortable participating. Professors at our institution use as many interfaces as possible to provide flexibility and accommodate individual needs. They use almost all Blackboard functions such as discussions, chat, assignments, mail, assignment drop box, selective release, podcast, vodcast, e-board, web access, breakout rooms, application share, tracking, etc. along with emails, telephone calls, and meetings to manage online courses. For competitive group projects, students are given access to their specific groups where they can upload documents and share within the group. Students are allowed to present their projects and attend all quizzes and exams online.
Most professors at our institution go beyond the institutional and technological barriers to deliver interactive and on time information. For instance, WebCT linkage with Wimba does not support mass texting or captioning. On time student interaction is critical for online learning. The project will implement a “Virtual Office Hours” to connect students in real time using Wiggio. Wiggio supports mass emailing, mass texting, voice notes, poling, hosting group conference calls and others to communicate with students. Accommodating disabled students was also possible in the online environment. In some cases, online media provides a better interface to decipher information for disabled students. We create transcripts of online lectures, caption video clips, and make them available to those students.

ii. Interactive Use of Multimedia

According to many researchers [14, 15], students learn best when they are actively engaged in the learning process. The more opportunities provided for students to participate and be involved, the more likely they will benefit from the course. We use multimedia applications such as PowerPoint, graphics, audio clips, video clips & animation in online courses to promote student involvement in their learning. PowerPoint supports hyperlinks and sound features that can easily create an interactive presentation that provides opportunities for students to employ higher cognitive strategies. Audio, digital video and animation are invaluable tools for demonstrating processes such as industrial automation.

iii. Effective Uses of e-Learning Tools

As mentioned earlier that we have found some very effective common tools in our online classes. It is very unlikely that a single tool or technique is enough for online learning. Creative uses of a combination of tools serve better in this case. Some of these tools are: email, discussion boards, application sharing, selective release, podcasting, vodcasting, and mass texting. Email is a teacher’s best friend. I encourage students to use email and discussion board extensively. We monitor discussion board continually, check our email several times a day and reply students’ email instantly. We have found that archived lectures with the PowerPoint slides helps student the most. Most of our classes have competitive group projects. We create separate rooms using
selective release so that students within the group use their specific room as virtual communication platform. It makes easy for students to do brain storming and share files with their group members without revealing information to other groups. Discussions generate ideas, help create a learning community in online classes, provide discussion transcripts, provide a means of online conference and collaboration, and get students thinking in writing as they write. We also found mass texting and voice messaging to student’s cell phones as convenient and on time communication tool to notify updates. WebCT announcements or emails are not able to do the same as texting.

**Promoting Critical Thinking**

Certainly, one of the greatest gifts teachers can give to students is the ability to think critically on a variety of topics. We hold our students accountable for their actions and set their expectations high. Professors need to value students’ input and opinions by serving as a facilitator. In our institution, we use various techniques such as Class Assessment, Cooperative Learning, Case Study, Open Ended Questions, Conference Style Learning, and Dialogue to promote critical thinking. Class Assessment Techniques monitor and facilitate students' critical thinking. Cooperative Learning puts students in a group learning situation which is the best way to foster critical thinking. Case Study methods present a case to the class without a conclusion. Professors then lead students through a discussion, allowing students to construct a conclusion for the case. Use of Dialogue and Open Ended Questions obliges students to think critically. In a Conference Style Learning environment, professors act as facilitators to maximize the organic flow of discussion and allow students to develop their critical thinking skills.

**Technology Impact on Student Learning**

Through the use of online tools and technology, learning can also be qualitatively different. The process of learning in online classes can become significantly richer as students have access to new and different types of information, can manipulate it on the computer through graphic displays or controlled experiments in ways never before possible, and can communicate their results and conclusions in a variety of media to their teacher, students in the next classroom, or students around the world. Online learning has its most promising potential in the high synergy
represented by active dialog among the participants, one of the most important sources of learning in a Virtual Classroom. Online teaching provides convenience, wider student access, and efficiency in student learning.

**Student Centered Learning**

Student centered learning is about helping students to discover their own learning styles, to understand their motivation and to acquire effective study skills that will be valuable throughout their lives. Our online classes are much more student centered than the typical face to face classes due to uses of various online tools. The students are much freer to move in individual directions in the online environment. Online teaching drove the “center” of the classroom from professor’s podium to students’ desktop. Our courses are designed as truly a student-centered learning episode. Most of our student not only meet the specific course objectives but go beyond that. They learn interpersonal skills, better communication skills, conduct researches, and publish articles. They learned to be competent in the challenging job market.

**Other techniques of engagement**

1. **Uses of WebCT Functions**

Most of our online professors use almost all of WebCT functions such as discussions, chat, assignments, mail, assignment drop box, selective release, podcast, vodcast, e-board, web access, breakout rooms, application share, tracking, etc. along with emails, telephone calls, and meetings to manage our online courses. Those professors who teach face to face format can also use online shell as a supplement for their face to face courses. Students can access course materials or view announcement from the online shell in case they miss the class or lost their handouts.

2. **Contengency Approach to Handle USM Server Downtime**

In the last year our WebCT server was down for atleast four times for more than several hours. This kind of situation puts students in a difficult position to prepare their quizes and exams and to submit their assignments ontime. We use Microsoft Office Live Workspace as a backup tool to handle that situation. Following figure shows an example of using Microsoft Office Live Workspace. Figure 3 shows the use of Microsoft Office Live Workspace in IET 409 course.
3. Use of Live Classroom

Access to Live Classroom gives face to face feelings to online students. Professors discuss lecture materials and solve representative mathematical problems while student can see and listen lectures in real time. They can ask questions using VOIP or chat functions. Sessions can be recorded for students to review at a later time. We use web, e-board, and share functions to make our online lectures interactive. Following figure show some example of Live Classroom use.
4. Administering Virtual Group Projects

Nowadays, collaboration is a necessity for graduates to work in real-world jobs. Most often engineers and technologists work with multi-disciplinary teams and require people management skills. These students must participate in group projects and case studies, which provide vital opportunities to effectively work as a team. The virtual projects are useful learning tools where the students assimilate and implement all the concepts they have learned in the classroom and get the feedback from teachers to pinpoint their roles in the group. In our classes, online students learn how to perform a group project despite they are physically separated. They need to organize the project and coordinate with their project partners using various communication tools. To make that easy, we created individual live classroom for each group where they can discuss their projects, share documents, and even present their project report to all students. Since it is a competitive project, only group members of each group have the access to their specific live classroom. We found this tool very effective for our students to conduct their group projects.

5. On Time Student Interaction

Many of our online professors go beyond the institutional and technological barriers to deliver interactive and on-time information. For instance, WebCT linkage with Wimba does not support mass texting or captioning. On-time student interaction is very critical for online learning. Many professors at our institution have implemented a “Virtual Office Hours” to connect students in real-time using Wiggio. Wiggio supports mass emailing, mass texting, voice notes, polling, hosting group conference calls and others to communicate with students. Following is an example of mass texting to IET 400 students to notify an impromptu meeting.
Figure 5: Impromptu meeting notification by mass texting using Wiggio for IET 400 class

6. Accommodating Disabled Students

Accommodating disabled students is very challenging in face to face teaching format, but can be easy in online environment with the use of appropriate tools. We create transcripts of our online lectures, caption video clips, and make them available to those students.

Measuring Technology Impact

As mentioned earlier that we are still in the process of assessing the impact of various tools and techniques at our institution and hence this paper doesn’t show any specific assessment results. We are using some measures and evaluating impacts while implementing various tools and techniques in online education. Table 1 summarizes the impacts and benefits.

Table 1: Summary impact and benefits of using classroom tool & technologies

| Facilitate adaptation at other sites | • Will educate educators and establish engineering education partners  
| | • The methods, materials, and assessment tools have the potential for transferability and for offering effective formative and summative feedback for engineering programs nationwide. |
| Advance discovery in teaching and learning in undergraduate engineering education | • Will enhance student learning by increasing student interactions/engagement in course content delivery methods via innovative open-source applications.  
• Expose students to actively engage and experience different ways of thinking and learning that aid cognitive flexibility.  
• Will engage more than 75 engineering students in the proposed alternate learning environments, to enrich the limited exposure that they currently gain about these engineering topics.  
• To seek Type II proposal funding after successful completion of current project. |
| Contribution to a paradigm shift in undergraduate engineering education | • Will demonstrate the implementation of a fully developed case study for industrial engineering technology education in an alternative environment to increase student interactivity via unique applications.  
• Approx. 52 faculty, including women and junior faculty will participate in intensive technological and educational training experiences. |

**Conclusion**

There are various classroom tools and techniques available for online teaching, but not all of these technologies are appropriate matches to the subject taught and the teacher’s pedagogical style and strategies. At the same time, all classroom tools and techniques are equally effective on students learning. Most of the time universities need to prioritize the uses of classroom technologies due to their budget constraints. Research on the effectiveness of various classroom tools and best practices on implementing various classroom technologies can be helpful for those universities. This paper tried to address that very issue. A better understanding of how sophisticated technology impacts teaching and learning in engineering will emerge through our assessment and evaluation efforts once completed.

The project being implemented will also effectively evaluate and assess student-learning outcomes. It encompasses methodologies that are not only sustainable and scalable, but will also standardize the instruction process of engineering courses, that can be easily adopted in any university setting. Finally, it promotes innovative uses of classroom technologies that not only help professors in alternate course delivery but improve students’ learning through better student-faculty interactions.
REFERENCES

Assessing BS–CS Student Outcomes Using Senior Project

Mr. Norman Pestaina, Florida International University

Mr. Norman Pestaina is a Senior Instructor in the School of Computing and Information Sciences (SCIS) at Florida International University (FIU). Mr. Pestaina completed the B.Sc. in Mathematics (Special) at the University of the West Indies in 1972, and the MS in Computer Science at the Pennsylvania State University in 1979. He has been an Assistant Staff member of the Massachusetts Institute of Technology’s Lincoln Laboratory, and Lecturer in the Department of Mathematics at the Cave Hill campus of the University of the West Indies in his native Barbados, W.I.

A member of the FIU-SCIS faculty for more than 30 years, Mr. Pestaina has taught at all levels of the curriculum, receiving awards for excellence in teaching on five occasions. He served the School as an Undergraduate Advisor for 15 of those years, and has served continuously as a member of the School’s Curriculum/Undergraduate Committees. Mr. Pestaina was a principal architect of the School’s program assessment processes, and the SCIS undergraduate program Assessment Coordinator from 2006 through 2013, leading successful re-accreditation of the BS in Computer Science program in 2004, and 2010. Mr. Pestaina has been a Reader and Question Leader of the College Board’s Advanced Placement Computer Science Exam since 2000.

Ms. Tiana Solis, Florida International University

Tiana Solis is currently a Senior Academic Adviser /Team Leader at the School of Computing and Information Sciences, Florida International University. Prior to moving to Hawaii in 2007, she was an instructor and academic advisor for the School from 1994 to 2007. Ms. Solis taught different undergraduate courses and mentored several FIU students participating in the Florida-Georgia Louis Stokes Alliance for Minority Participation (FGLSAMP). She is a past adviser of the Women in Computer Science (WICS) student club. From 2008 to 2010. Ms. Solis was a programmer analyst at the Department of the Attorney General in Hawaii, a member of the team revamping the State Juvenile Justice Information System. Her research and instructional Interests include software development, programming languages, and computer ethics.

Dr. Peter J. Clarke, Florida International University

Peter J. Clarke received his BSc. degree in Computer Science and Mathematics from the University of the West Indies (Cave Hill) in 1987, MS degree from SUNY Binghamton University in 1996 and PhD in Computer Science from Clemson University in 2003. His research interests are in the areas of software testing, software metrics, model-driven software development, domain-specific modeling languages, and computer science education. He is currently an associate professor in the School of Computing and Information Sciences at Florida International University. He is a member of the ACM (SIGSOFT, SIGCSE, and SIGAPP); IEEE Computer Society; and a member of the Association for Software Testing (AST).
Assessing BS–CS Student Outcomes Using Senior Project

Abstract

Undergraduate program assessment is undertaken by many colleges world-wide in support of their continuous improvement processes. In addition to assuring stakeholders of program quality, assessment is required by major regional and national accrediting agencies. A critical part of the assessment process is the generation of useful data for analysis and evaluation yielding indicators for program improvement. Senior year capstone projects are a fertile source of such data. In this paper, we outline the Student Outcomes and Senior Project course of the BS-CS program at Florida International University (FIU). We describe and evaluate a methodology used to perform assessment of attainment of the BS-CS Student Outcomes using data from the Senior Project course.

1. Introduction

Undergraduate program assessment continues to be a significant undertaking in many institutions. It is the critical component of the continuous improvement process, and may provide stakeholders with some confidence in the quality of the programs. Also, assessment and continuous improvement practices are invariably prerequisites for program accreditation, a designation conferred by entities external to the departments in which the undergraduate programs are offered. The Bachelor of Science in Computer Science (BS-CS) program at Florida International University (FIU) is externally accredited by both the Computing Accreditation Commission of the Accreditation Board of Engineering and Technology (ABET/CAC)1, an international accreditation body, and the Southern Association of Colleges and Schools (SACS)12, a regional accreditation body.

The BS-CS at FIU is a dynamic program that implements many of the recommendations outlined in successive ACM/IEEE Computing Curricula for Computer Science 2. Resulting from an internal review of the BS-CS program undertaken in 2006, the Senior Project course became a required component of the BS-CS program in Fall 2007. Since the BS-CS program structure is topic-based, the Senior Project course was designed as a capstone experience. It enables students to integrate knowledge units from several CS courses such as data structures, database management, operating systems, etc. in a holistic way. The Senior Project also allows students to complete a significant project encompassing both design and implementation, and requiring elements of professional practice such as teamwork, presentation skills and ethical considerations. Introduction of the Senior Project into the BS-CS curriculum also meant that there would now be an excellent source of direct summative assessment information.

The contributions of the paper are:
1. To present our approach to assessing attainment of the BS-CS Student Outcomes based on the Senior Project course.
2. To present an evaluation of the approach using data collected from its application over two years.
The paper is organized as follows: Section 2 presents the related work; Section 3 an overview of the BS-CS program; Section 4 our approach for assessment; Section 5 results of a case study; and Section 6 the conclusion.

2. Related Work

In this section we describe the research most closely related to the work presented in this paper.

2.1 Assessing BS-CS Programs

Sanders et al.\(^{10}\) investigated the tools that were being used to assess ABET/CAC accredited CS programs in the US. They found that ten (10) different tools were used including senior exit surveys, alumni surveys, employer surveys, written exams and portfolios (maintained by the department and the students), among others. Bailie et al.\(^{6}\) describe how rubrics for two programming courses can be used to measure the student learning outcomes derived from the ABET’s Program Outcomes. In addition the authors describe how the data collected was used to improve the process.

In our work we show how to use the senior capstone project to evaluate attainment of Student Outcomes (SOs) through the BS-CS curriculum. Our approach assumes a mapping of our SOs to the ABET/CAC 2010 Student Outcomes\(^{1}\), and evaluates them by application of a rubric. However, our rubric has a different structure to the one presented by Bailie et al.\(^{6}\).

2.2 Assessment of Senior Projects

Clear et al.\(^{7}\) present a report that assists instructors to design, implement and assess capstone courses. The report is a valuable resource for departments starting a new capstone course. Richards\(^{9}\) describes the key design choices of a project-based course, focusing on the composition of the groups and the issues surrounding assessment. Both Clear et al.\(^{7}\) and Richards\(^{9}\) provide comprehensive descriptions of the assessment of projects but there is no description of how the capstone project may be used to assess the SOs of a computer science program. Farrell et al.\(^{8}\) describe an approach that attempts to develop a system for the fair allocation of course grades to the members of the senior project team. In grading our senior projects we use some of the ideas presented by Farrell et al.\(^{8}\), e.g., peer group assessment and evaluating meeting minutes.

Ahmad et al.\(^{5}\) performed a study of the undergraduate software capstone project at 19 Pakistani universities and provides generic support for quality assessment of capstone projects at the undergraduate level. The study investigated the current practices followed for assessment of computer science and software engineering capstone projects and the formulation of generic rubrics for quality assessment to minimize variation in quality.
3. BS Computer Science Program

In this section we describe the SOs for the BS-CS program at FIU, identify the principal enabling courses in the program, and describe the rationale for our Senior Project course.

3.1 The BS-CS Student Outcomes

The Student Outcomes (SOs) of an academic program are statements of the general characteristics of the program’s graduates. Typically, they express abilities of graduates that are enabled by students’ progress through the program. Additionally the SOs are intended to make the broader Program Objectives realizable, and are often adjusted as the Program Objectives evolve.

To complete the program of study for the BS-CS, every student will

a) Demonstrate proficiency in the foundation areas of Computer Science including discrete structures, logic and the theory of algorithms.

b) Demonstrate proficiency in various areas of Computer Science including data structures and algorithms, concepts of programming languages and computer systems.

c) Demonstrate proficiency in problem solving and application of software engineering techniques.

d) Demonstrate mastery of at least one modern programming language and proficiency in at least one other.

e) Demonstrate understanding of the social and ethical concerns of the practicing computer scientist.

f) Demonstrate the ability to work cooperatively in teams.

g) Demonstrate effective communication skills.

h) Have experience with contemporary environments and tools necessary for the practice of computing.

Figure 1: FIU BS-CS Student Outcomes

The current iteration represented in Figure 1 is already scheduled for update during the 2012-13 academic year.

3.2 Enabling Courses

The principal enabler of the SOs is the curriculum. For the BS-CS at FIU, the required courses most closely aligned with each SO are listed:

a) **Foundations**: Discrete Mathematics, Logic for CS, Theory of Algorithms;

b) **CS Core**: Data Structures, Principles of Programming Languages, Database Management, Computer Organization, Operating Systems Principles, required CS electives;

c) **Software Development**: Software Engineering I, Senior Project;

d) **Programming**: Computer Programming I, II & III;

e) **Ethics**: Ethics and Social Issues in Computing;

f) **Teamwork**: Software Engineering I, Senior Project;
g) **Communication**: Professional and Technical Writing; Business and Professional Communication; Senior Project.

h) **Computing Tools**: Operating Systems Principles, Database Management, Software Engineering I, Senior Project.

It should be noted that a recent BS-CS curriculum review has resulted in several changes effective in Fall 2012.

### 3.3 Senior Project

Our Senior Project course CIS 4911 was designed as a capstone experience for our graduating seniors. It is a required course of the BS in CS curriculum:

<table>
<thead>
<tr>
<th>Catalog Description: Students work on faculty supervised projects in teams of up to 5 members to design and implement solutions to problems utilizing knowledge obtained across the spectrum of Computer Science courses. This course should be taken during the semester in which the student completes all the CS courses required for the CS major.</th>
</tr>
</thead>
</table>

Figure 2: Catalog description of CIS 4911 Senior Project

Figure 2 shows the catalog description of CIS4911, the complete operational CIS 4911 syllabus may be viewed from: [http://www.cis.fiu.edu/programs/undergrad/courses/CIS_4911.pdf](http://www.cis.fiu.edu/programs/undergrad/courses/CIS_4911.pdf)

Our BS-CS program is organized following a traditional topic-based approach. The knowledge units are delivered in disjoint subject-focused courses: data structures, database management, operating systems, etc. Although there is a prerequisite structure, the intersection of knowledge units between courses is minimal. It is now generally accepted that successful practice in a computing field requires a holistic competence. One role of the capstone course in a topic-based curriculum is to foster an appreciation of how the knowledge units relate, and to forge that holistic competence. Inclusion of a capstone course into a topic-based curriculum is recommended in ACM Curricula 2001².

Currently, two flavors of the Senior Project are offered; one with a software engineering focus and the other with a systems focus. Both require development of a software system and include requirements specification, design, implementation and some form of validation. Students are required to submit four (4) deliverables during the semester and make an oral presentation of each deliverable. All presentations are recorded so that students and instructors can review the presentation performances.

### 4. Approach for Assessment

In an educational context, assessment is a process for gathering information about elements of an educational program, analyzing the information to extract indicators of program effectiveness, and evaluating the indicators against program expectations: i) to make inferences about program effectiveness, ii) to identify signposts for improving the program, iii) to make adjustments to the assessment process itself. The SOs are one set of program markers. Precisely because of the
holistic nature of a well-designed capstone, all SOs can reasonably be expected to be manifested via completion of a capstone project. This confluence accounts for the suitability of the capstone or Senior Project course as a medium for assessment of the program.

Our BS-CS program utilizes assessment of SOs in the Senior Project course as one component of its assessment plan. Other components include course-embedded assessment via quizzes and application of specialized rubrics to course artifacts; in addition, indirect measures are obtained by surveying students, instructors, alumni and employers. In the remainder of this section, the structure of the Senior Project assessment process is described.

4.1 Data Collection

One source of assessment raw data is the set of completed projects – presentations and artifacts. The data are extracted by applying rubrics to the presentations and artifacts. The agents applying the rubrics are evaluators, faculty members or industry clients, and student team-members.

Unsurprisingly, the master rubric, *Senior Project Rubric* has eight sections, one for each of the eight Student Outcomes, see Section 3.2:

- The first five sections, (a) Foundations, (b) CS Core, (c) Software Development, (d) Programming, and (e) Ethics are check-lists completed by the evaluators. In each section, the sum of the check-marks (1 if checked, 0 if not checked), up to a limit of 5, provides a rating for attainment of the Student Outcome of that section. See the rubric for the Foundations Student Outcome in Appendix A.

- The principal component of section (f) Teamwork is the *Teamwork Peer Rating Rubric*. It is completed by each team-member to rate participation of each of their other team members. Five team-work facets are rated against two criteria each. These ten ratings provided by each team-member are averaged to obtain ten criterion team ratings in the range 1 to 5. Each team-work facet earns a check-mark only if both of its criteria receive an average rating of at least 4. The evaluators provide check-marks for two other team-work facets. The sum of the check-marks, up to a limit of 5, provides a rating of the Teamwork Student Outcome. See the rubric for the Teamwork Student Outcome in Appendix B.

- The principal component of section (g) Communication is the *Presentation-Skills Rubric*. It is completed by the evaluators to rate the oral presentation skills of each team member. Each presenter is rated on five presentation facets using traditional criteria. For each facet, the presenters’ ratings are averaged to obtain a team rating for that facet. The facet earns a check-mark if the team rating is at least 3 from a maximum of 4. The evaluators provide check-marks for two other facets. The sum of the check-marks, up to a limit of 5, provides a rating of the Communication Student Outcome.

- Section (h) Computing Tools is entirely a *Computing Tools Rubric* where each project team, collectively, enumerates the tools and environments employed by the team, and provides self-ratings of their proficiency with each environment or tool. The tools are categorized by application domain: Modeling, Project Management, DBMS, etc. The domain earns a check-mark if the team’s self-rating of competence with the domain tools is above the *novice* level.
The sum of the check-marks, up to a limit of 5, provides a rating of the Computing Tools Student Outcome.

At the top level, the 1st Tier, the methodology is enumerative; it checks for utilization of SO knowledge area facets in the project implementation. A lower level, the 2nd Tier, provides more fine-grained information that might be useful for focusing continuous improvement efforts.

4.2 Analysis

Each Senior Project is independently rated by two evaluators via application of the Senior Project Rubric. This yields two ratings of attainment of each SO in the range 1 to 5. The ratings are discrete numbers. The paired ratings are combined to obtain a single project-rating for each SO:

- When the paired ratings are identical, the common rating is the SO rating.
- When the paired ratings differ by 1, their average is the SO rating.
- When the paired ratings differ by more than 1, a mediated rating is provided by a third evaluator (mediator).

For each SO, the project-rating from all Senior Projects in the semester are averaged to obtain a semester rating of attainment of the SOs. These ratings are reported in the semester summary of direct assessment data. The data from all semesters in an academic year are reported in an annual summary.

4.3 Evaluation

There is frequent misunderstanding of what outcomes assessment is about; that is, what is being evaluated. In this instance, it is not the students, or their projects, or the quality of mentorship, or the curriculum. What is being evaluated is the efficacy of the program; the assessment process described here seeks to answer the question “How well does the BS-CS program enable its graduates to attain the program’s SOs”. The data analysis yields metrics that are interpreted as answers to this question for each SO.

To evaluate using metrics, a standard or minimal acceptable rating value must be established. The default standard adopted by our program, for all measures, is 75% of the maximum rating. In this case the standard is 3.75; that is the threshold at which the Senior Project Student Outcome ratings are deemed acceptable, and below which a rating raises a red flag.

Example: the AY2011-12 summary reports Senior Project ratings of 4.56 (91%) for SO (b), and 2.72 (54%) for SO (e). Taken alone, these suggest good attainment of the SO (b), but poor attainment of SO (e). In practice, each SO is evaluated using several metrics derived via course-embedded assessment as well as surveys. There are good reasons for this seeming duplication of effort:

- Having multiple indicators increases the potential for meaningful evaluation.
- Especially for continuous improvement purposes, it is helpful to have both formative and summative indicators.
• Some assessment methods may be poorly suited to evaluating a particular outcome.
• A particular assessment instrument (rubric, survey, and quiz) may be poorly designed, or awkward to execute.
• An agent executing a particular assessment instrument may do so carelessly, or with intent to mislead.

In the particular case of our Senior Project Rubric\textsuperscript{11}, any questions of applicability and consistency should be resolved in order to have confidence in the metrics it provides. It must also be flexible enough to accommodate modifications or additions of SOs. The timing of this study is, in part, dictated by impending changes to the BS-CS SOs for closer alignment with the ABET SOs.

5. Case Study

As described earlier, CIS 4911 is a capstone course coming towards the end of the students’ program of study. This makes CIS 4911 a prime source for summative assessment of attainment of SOs, and accounts for the significant effort invested in development of the Senior Project Rubric\textsuperscript{11}.

5.1 Research Questions

The overarching question is whether to adapt the existing Senior Project Rubric, with some (hoped) improvements, to a revised set of SOs. To that end, some desired attributes of the Senior Project Rubric are evaluated:

RQ1 (Applicability): Does the rubric’s methodology measure attainment of all of the SOs in a meaningful way? If not all, to which SOs is it not applicable?
RQ2 (Extendibility): Is the rubric methodology(s) applicable to varied SO categories, and does it lend itself to fine tuning?
RQ3 (Consistency): Is there reasonable expectation that repeated application of the rubric to identical data should yield identical metrics? When there is not, can this shortcoming be ameliorated?

5.2 Method

The sources of the data for this study are 22 capstone projects completed between Fall 2010 and Spring 2012, encompassing the efforts of about 82 students. The numeric data were obtained by applying the Senior Project Rubric described earlier to the presentations and artifacts of these 22 projects. Informal reactions and feedback from evaluators applying the Senior Project Rubric provide useful insight. Thus, some anecdotal experiential information is utilized.

For each SO, evaluators determine a rating of attainment on a scale of 1 to 5, based on evidence obtained by examination of the project artifacts, or by observation of oral presentations. For each SO, the rubric provides a check-list of at least 7 rubric-points that may be earned. The number of
checked rubric-points, up to a maximum of 5, provides the rating. The evaluator may choose to record an \textit{n/a} rating when there is insufficient evidence of a particular outcome in the artifacts. For SO (f) Teamwork and SO (h) Computing Tools, the ratings rely on rubrics completed by the students themselves (see section 4.1). For SO (g) Communication, the ratings are obtained by completing the \textit{Oral Presentation Rubric}. For all remaining SOs, (a) through (e), the evaluator must examine the project artifacts. To assist the evaluator in locating the relevant sections, each project team completes an \textit{Outcomes Check-List} of entries that locate the available relevant artifact documentation.

\textbf{RQ1 (Applicability):}

Intuitively, the knowledge areas represented by SOs (b) CS Core, (c) Software Engineering, (d) Programming, (f) Teamwork, (g) Communication, and (h) Computing Tools are intrinsic to any team project with a presentation component. SOs (a) Foundations, and (e) Ethics, are less obviously essential. However, the Ethics component is listed among the CIS 4911 course outcomes, and the ABET SO (j) explicitly includes “ability to apply mathematical foundations” as part of the mix in the design of computer-based systems. Consequently, the \textit{Senior Project Rubric} does enumerate examples of such features that might reasonably be expected in a capstone or life-scale project.

\textbf{RQ2 (Extendibility):}

The principal strategy employed in the \textit{Senior Project Rubric} is enumeration of project features within each SO knowledge area. When qualitative assessment is desired, suitably designed subordinate rubrics yield metrics to be compared against pre-determined standards to yield check-marks. Thus, the \textit{Senior Project Rubric} is two-tiered, and is extendible:

- By adding additional check-lists for new outcomes (1\textsuperscript{st} Tier)
- By implementing subordinate rubrics to generate metrics to determine check-marks as described above (2\textsuperscript{nd} Tier)

\textbf{RQ3 (Consistency):}

The rubric must allow consistent application with an expectation of identical results when applied conscientiously by different agents. Because the \textit{Senior Project Rubric} has been applied to each completed Senior Project by at least two evaluators, the frequency of identical or near-identical comparisons may yield a significant measure of the rubric’s consistency.

\textbf{5.2 Results}

In the preceding section the \textit{Senior Project Rubric} was characterized in terms of the following attributes:
RQ1 (*Applicability*):

Table 1 presents the distribution of Senior Project Rubric ratings over 22 completed projects, for each of the SOs. SO (a) is the only outcome to receive a rating of *n/a*, and only SOs (a) and (e) received any ratings of 1. The metrics for SO (a) and SO (e) are essentially counts of the number of instances of project features that lie within the domains of these outcome areas. Unless project specifications explicitly require application and documentation of features from the domains outside of the project focus-areas, students will not include them.

<table>
<thead>
<tr>
<th>Rating</th>
<th>(a)</th>
<th>(b)</th>
<th>(c)</th>
<th>(d)</th>
<th>(e)</th>
<th>(f)</th>
<th>(g)</th>
<th>(h)</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>1</td>
<td>10</td>
<td>21</td>
<td>14</td>
<td>5</td>
<td>19</td>
<td>21</td>
<td>18</td>
</tr>
<tr>
<td>4</td>
<td>2</td>
<td>6</td>
<td>1</td>
<td>17</td>
<td>6</td>
<td>11</td>
<td>1</td>
<td>4</td>
</tr>
<tr>
<td>3</td>
<td>7</td>
<td>3</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>3</td>
<td>1</td>
<td>0</td>
<td>5</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>7</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td><em>n/a</em></td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

Table 1: Rating Frequency for each SO, All Data

The anecdotal evidence suggests that project clients have little interest in non-focus-area aspects. There is also evidence of this counter-productive tendency in the data for SO (b) CS Core. Incorporation and documentation of these aspects are not included unless project specifications, or mentors, require it.

As depicted in Table 1, the rubric provides good indicators of the utilization (whether high or low) of all SO areas. The rubric’s methodology is broadly applicable to obtain data on any SO whose subject area is utilized in the project implementation.

RQ2 (*Extendibility*):

The rubric is extendible in two ways. First, it can be extended to provide quantitative evaluation of a new SO by simply adding a 1<sup>st</sup> Tier check-list specific to the added SO. Second, it is may be extended to provide qualitative evaluation by assigning standards (minimum acceptability) for each bullet of an existing check-list, and designing a 2<sup>nd</sup> Tier rubric to provide qualitative ratings. These may be compared against the standards to provide check-marks for the high level 1<sup>st</sup> Tier assessment.

The single project presented in Summer 2011 was assessed using the Spring 2011 rubric, so the Summer 2011 data are counted with the Spring 2011 data. There were no rubric changes between Fall 2011 and Spring 2012.

The *Senior Project Rubric* was adjusted prior to application in Spring 2011, and again prior to Fall 2011. Table 2 shows the data for four semesters, $\mu$ – mean and $\sigma$ – standard deviation. Specifically:
• Spring 2011 – Addition of specific examples in check-mark criteria for both SO (a) Foundations, and SO (b) CS Core. This may account for improved statistics for SO (b) post Fall 2010.

• Fall 2011 - Incorporation of the 2nd Tier Teamwork Peer Rating Rubric, to seed the SO (f) Teamwork ratings. The change in the rating probably reflects a more realistic assessment in Fall 2010 and Spring 2011

• Fall 2011 - Incorporation of the 2nd Tier Presentation Skills Rubric, to seed the SO (g) Communication ratings. Informal feedback indicates altering the format for easier application.

• Fall 2011 - Incorporation of the 2nd Tier Computing Tools Rubric, to seed the SO (h) Computing Tools ratings. The rubric now more accurately reflects the breadth of students’ experience with a variety of computing tools.

<table>
<thead>
<tr>
<th>Student Outcome (SO)</th>
<th>(a)</th>
<th>(b)</th>
<th>(c)</th>
<th>(d)</th>
<th>(e)</th>
<th>(f)</th>
<th>(g)</th>
<th>(h)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fall 2010 (N = 7)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>( \mu )</td>
<td>3.0</td>
<td>3.7</td>
<td>4.9</td>
<td>4.9</td>
<td>3.9</td>
<td>5.0</td>
<td>4.9</td>
<td>4.6</td>
</tr>
<tr>
<td>( \sigma )</td>
<td>0.96</td>
<td>1.25</td>
<td>0.17</td>
<td>0.17</td>
<td>1.37</td>
<td>0.00</td>
<td>0.17</td>
<td>0.44</td>
</tr>
<tr>
<td>Spring &amp; Summer 2011 (N = 6)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>( \mu )</td>
<td>2.9</td>
<td>4.0</td>
<td>5.0</td>
<td>4.7</td>
<td>4.1</td>
<td>4.9</td>
<td>5.0</td>
<td>4.8</td>
</tr>
<tr>
<td>( \sigma )</td>
<td>1.48</td>
<td>1.15</td>
<td>0.00</td>
<td>0.47</td>
<td>0.73</td>
<td>0.19</td>
<td>0.00</td>
<td>0.37</td>
</tr>
<tr>
<td>Fall 2011 (N = 6)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>( \mu )</td>
<td>1.1</td>
<td>4.5</td>
<td>5.0</td>
<td>4.7</td>
<td>2.5</td>
<td>4.8</td>
<td>5.0</td>
<td>5.0</td>
</tr>
<tr>
<td>( \sigma )</td>
<td>0.67</td>
<td>0.58</td>
<td>0.00</td>
<td>0.24</td>
<td>0.76</td>
<td>0.37</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td>Spring 2012 (N = 3)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>( \mu )</td>
<td>1.8</td>
<td>4.7</td>
<td>5.0</td>
<td>4.5</td>
<td>3.2</td>
<td>4.3</td>
<td>5.0</td>
<td>5.0</td>
</tr>
<tr>
<td>( \sigma )</td>
<td>0.24</td>
<td>0.47</td>
<td>0.00</td>
<td>0.71</td>
<td>0.62</td>
<td>0.94</td>
<td>0.00</td>
<td>0.00</td>
</tr>
</tbody>
</table>

Table 2: Comparison of Senior Project Data by Semester

RQ3 (Consistency):

This important attribute can be measured by employing paired-agent application. We define two consistency measures:

• \( \delta=0 \) – # of identical paired ratings
• \( \delta=0,1 \) – # of paired ratings that differ by 0 or by 1

The results, over all 22 applications, are summarized in Table 3:

<table>
<thead>
<tr>
<th></th>
<th>FL 10</th>
<th>SP 11</th>
<th>FL 11</th>
<th>SP 12</th>
<th>ALL</th>
</tr>
</thead>
<tbody>
<tr>
<td>#Pairs</td>
<td>56</td>
<td>48</td>
<td>48</td>
<td>24</td>
<td>176</td>
</tr>
<tr>
<td>( \delta=0 )</td>
<td>57.1%</td>
<td>72.9%</td>
<td>70.8%</td>
<td>75.0%</td>
<td>67.6%</td>
</tr>
<tr>
<td>( \delta=0,1 )</td>
<td>87.5%</td>
<td>87.5%</td>
<td>93.8%</td>
<td>87.5%</td>
<td>89.2%</td>
</tr>
</tbody>
</table>

Table 3: Consistency Ratings by semester
\( \delta=0,1 \) is consistently just below 90%, but \( \delta=0 \) has improved from an initial 57% to be consistently around the 70% mark. From Fall 2011 and after, the SO (h) ratings are entirely determined by student self-ratings, and SO (f) almost entirely by student peer ratings. Nonetheless, the semester \( \delta=0,1 \) levels indicate that the rubric can be applied consistently.

Table 4 shows that at the outcome level, there is too high inconsistency for SO (a) and moderate inconsistency for SO (b) and for SO (e):

<table>
<thead>
<tr>
<th>Student Outcome (SO) (%)</th>
<th>(a)</th>
<th>(b)</th>
<th>(c)</th>
<th>(d)</th>
<th>(e)</th>
<th>(f)</th>
<th>(g)</th>
<th>(h)</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \delta=0 )</td>
<td>27.3</td>
<td>50.0</td>
<td>90.9</td>
<td>59.1</td>
<td>45.5</td>
<td>90.9</td>
<td>86.4</td>
<td>86.4</td>
</tr>
<tr>
<td>( \delta=0,1 )</td>
<td>68.2</td>
<td>77.3</td>
<td>100.0</td>
<td>95.5</td>
<td>81.8</td>
<td>95.5</td>
<td>95.5</td>
<td>100.0</td>
</tr>
</tbody>
</table>

Table 4: Consistency Ratings by Student Outcome

The semester trends for these outcomes are shown in Table 5:

<table>
<thead>
<tr>
<th>( \delta=0,1 )</th>
<th>FL_10</th>
<th>SP_11</th>
<th>FL_11</th>
<th>SP_12</th>
</tr>
</thead>
<tbody>
<tr>
<td># Pairs</td>
<td>7</td>
<td>6</td>
<td>6</td>
<td>3</td>
</tr>
<tr>
<td>SO (a)</td>
<td>85.7%</td>
<td>50%</td>
<td>83.3%</td>
<td>33.3%</td>
</tr>
<tr>
<td>SO (b)</td>
<td>57.1%</td>
<td>66.7%</td>
<td>100.0%</td>
<td>100.0%</td>
</tr>
<tr>
<td>SO (e)</td>
<td>100%</td>
<td>83%</td>
<td>66.7%</td>
<td>66.7%</td>
</tr>
</tbody>
</table>

Table 5: SOs (a), (b), (e): \( \delta=0,1 \) consistency by semester

The trend for SO (b) is encouraging; the consistency seems to have responded positively to the rubric adjustments. However, the fluctuation in the consistency rating of SO (a), and the decline in consistency of SO (e) are not reassuring. The anecdotal evidence of evaluators is helpful. When not explicitly required, students fail to document SO (a), (b) and (e) aspects. A conscientious evaluator finds aspects that are not located by the students’ check-lists; casual evaluators do not check for them.

6. Conclusion

We have described an approach to assessing attainment of Student Outcomes (SOs) of CS programs in capstone courses at Florida International University. Our approach is a two-tiered rubric. It allows a high-level 1st Tier enumerative methodology using check-lists that optionally may be supported by a 2nd Tier qualitative methodology. Our implementation of the Senior Project Rubric was initially single-tiered, but was soon supplemented by addition of a second tier to support assessment of three of the eight SOs.

This study has utilized the metrics produced by our rubric to demonstrate that the method is broadly applicable, and flexible to accommodate scaling and fine-tuning. The consistency of the metrics yielded by our instrument is often good. The exceptions underscore the prerequisite of purposeful mentorship that reinforces the capstone dimensions of the projects.
In the near term, the Senior Project Rubric will be modified to accommodate changes to our SOs. This study provides enough confidence in the applicability and scalability of the method, and consistency of its metrics, to suggest that retention of the instrument’s methodology is practicable.

References


APPENDIX

A. Senior Project Rubric for Outcome (a)

Student Outcome (a): *Demonstrate proficiency in the foundation areas of Computer Science including discrete structures, logic and the theory of algorithms*\textsuperscript{11}

___ Project incorporates elements of mathematical reasoning or proof (Lemma, Theorem, Propositional Logic, First Order Logic, Mathematical Induction)

___ Project utilizes elements of discrete mathematics (Set Theory, Boolean Algebras, Combinatorics, Graph Theory)

___ Project utilizes some statistical procedure(s) to represent or summarize test data (Mean, Standard Deviation, Stem Plot/Histogram, Box Plot/Percentile-Graph)

___ Project utilizes some statistical measure(s) of system behavior or performance (Probability Distributions, Confidence Intervals, Hypothesis Testing)

___ Project design utilizes finite state diagrams to model system behavior

___ Project utilizes some aspect(s) of formal computer science (Automata, Turing Machines, Recursive Function Theory, Recursive Unsolvability)

___ Project utilizes some technique(s) of numerical analysis (Error Estimation, Interpolation, Numerical Calculus, Linear Systems, Matrix Algebra)

B. Senior Project Rubric for Outcome (f)

Student Outcome (f): *Demonstrate the ability to work cooperatively in teams* \textsuperscript{11}

To be completed by an evaluator:

___ Project presentation(s) included all team members equally

___ Project team activity is appropriately and adequately documented

To be completed from the data obtained from team members’ peer evaluations:
Each team member rates each of the other members of their team individually on each criterion listed below on a scale of 1 to 5. The mean of all ratings for each criterion is recorded.

*The rubric item is checked only if the project (mean) score >= 4.0 for each of the 2 criteria.*
Team members’ roles were clearly defined and executed

<table>
<thead>
<tr>
<th>Criterion</th>
<th>Mean Score</th>
</tr>
</thead>
<tbody>
<tr>
<td>1: <em>Team members had clear understanding of expectations</em></td>
<td></td>
</tr>
<tr>
<td>2: <em>Team members maximized the use of their individual skill sets</em></td>
<td></td>
</tr>
</tbody>
</table>

Project team set out and followed a schedule for timely completion

<table>
<thead>
<tr>
<th>Criterion</th>
<th>Mean Score</th>
</tr>
</thead>
<tbody>
<tr>
<td>3: <em>Team members complied with mechanisms to track progress</em></td>
<td></td>
</tr>
<tr>
<td>4: <em>Team members completed assignments in a timely fashion</em></td>
<td></td>
</tr>
</tbody>
</table>

Project team negotiated consensus when needed

<table>
<thead>
<tr>
<th>Criterion</th>
<th>Mean Score</th>
</tr>
</thead>
<tbody>
<tr>
<td>5: <em>Team members showed respect for other team members opinions</em></td>
<td></td>
</tr>
<tr>
<td>6: <em>Team members were able to negotiate and compromise</em></td>
<td></td>
</tr>
</tbody>
</table>

Project completion evidences equitable participation by team members

<table>
<thead>
<tr>
<th>Criterion</th>
<th>Mean Score</th>
</tr>
</thead>
<tbody>
<tr>
<td>7: <em>Team members contributed ideas and viewpoints</em></td>
<td></td>
</tr>
<tr>
<td>8: <em>Team members did their fair share of the work</em></td>
<td></td>
</tr>
</tbody>
</table>

Team members shared responsibility for success and failure

<table>
<thead>
<tr>
<th>Criterion</th>
<th>Mean Score</th>
</tr>
</thead>
<tbody>
<tr>
<td>9: <em>Team members actively sought &amp; shared information from each other</em></td>
<td></td>
</tr>
<tr>
<td>10: <em>Team members were adaptable to changing requirements</em></td>
<td></td>
</tr>
</tbody>
</table>
Hybrid MPI-OpenMP versus MPI Implementations: A Case Study

Mr. Osvaldo Mangual, Honeywell Aerospace

Osvaldo Mangual has a Bachelor degree in Computer Engineering and a Master degree in Electrical Engineering from Polytechnic University of Puerto Rico in the area of Digital Signal Processing. He currently works at Honeywell Aerospace, PR, as a FPGA and ASIC Designer.

Dr. Marvi Teixeira, Polytechnic University of Puerto Rico

Dr. Teixeira is a Professor at the Electrical and Computer Engineering and Computer Science Department at Polytechnic University of Puerto Rico. He holds a Ph.D. and MSEE degrees from University of Puerto Rico at Mayaguez and a BSEE degree from Polytechnic University. Professor Teixeira is an IEEE Senior Member, a Registered Professional Engineer and a former ASEE-Navy Summer Faculty Fellow.

Mr. Reynaldo Lopez-Roig, Polytechnic University of Puerto Rico

Mr. Lopez received his B.S. in Computer Engineering from the Polytechnic University of Puerto Rico in 2013. His work as an undergraduate research assistant was related to the implementation and benchmarking of parallel signal processing algorithms in clusters and multicore architectures. Mr. Lopez is currently working at the Jet Propulsion Laboratory as a Software Systems Engineer.

Prof. Felix Javier Nevarez-Ayala, Polytechnic University of Puerto Rico

Felix Nevarez is an Associate Professor in the Electrical and Computer Engineering Department at the Polytechnic University of Puerto Rico. He earned a BS in Physics and a MS in Electrical Engineering from the University of Puerto Rico at Mayaguez and a BS in Electrical Engineering from the Polytechnic University of Puerto Rico. His research interests are in numerical simulations and parallel and distributed programming.
Hybrid MPI-OpenMP versus MPI Implementations: A Case Study

Osvaldo Mangual*, Marvi Teixeira*, Reynaldo Lopez*, Felix Nevarez*
+ Honeywell Aerospace, Aguadilla, PR., *Polytechnic University of Puerto Rico, #Jet Propulsion Laboratory, Pasadena, CA.

Abstract

In this paper we explore the performance of a hybrid, or mixed-mode (MPI-OpenMP), parallel C++ implementation versus a direct MPI implementation. This case-study provides sufficient amount of detail so it can be used as a point of departure for further research or as an educational resource for additional code development regarding the study of mixed-mode versus direct MPI implementations. The hardware test-bed was a 64-processor cluster featuring 16 multi-core nodes with four cores per node. The algorithm being benchmarked is a parallel cyclic convolution algorithm with no inter-node communication that tightly matches our particular cluster architecture. In this particular case-study a time-domain-based cyclic convolution algorithm was used in each parallel subsection. Time domain-based implementations are slower than frequency domain-based implementations, but give the exact integer result when performing very large, purely integer, cyclic convolution. This is important in certain fields where the round-off errors introduced by the FFTs are not acceptable. A scalability study was carried out where we varied the signal length for two different sets of parallel cores. By using MPI for distributing the data to the nodes and then using OpenMP to distribute data among the cores inside each node, we can match the architecture of our algorithm to the architecture of the cluster. Each core processes an identical program with different data using a single program multiple data (SPMD) approach. All pre and post-processing tasks were performed at the master node. We found that the MPI implementation had a slightly better performance than the hybrid, MPI-OpenMP implementation. We established that the speedup increases very slowly, as the signal size increases, in favor of the MPI-only approach. Even though the mixed-mode approach matches the target architecture there is an advantage for the MPI approach. This is consistent with what is reported in the literature since there are no unbalancing problems, or other issues, in the MPI portion of the algorithm.

Introduction

In certain fields, where the round-off errors introduced by the FFTs are not acceptable, time domain-based implementations of cyclic convolution guarantee the exact integer result when performing very large, purely integer, cyclic convolution. The trade-off is that these implementations are slower than frequency domain-based implementations. Applications that can benefit from this approach include multiplication of large integers, computational number theory, computer algebra and others.1,2 The proposed, time domain-based, parallel implementation can be considered as complementary to other techniques, such as Nussbaumer convolution3 and Number Theoretic Transforms4, which can also guarantee the exact integer result but could have different length-restrictions on the sequences.

Parallel implementations in cluster architectures of time domain-based, purely integer, cyclic convolution of large sequences resulted much faster than the direct, time domain-based, O(N^2) implementation in a single processor. This is not the case for frequency domain-based implementations where the direct implementation in a single processor is usually faster than the
parallel formulation, and therefore preferably, unless memory limitations or round-off errors become an issue as it happens with the mentioned applications.

The algorithm being benchmarked is a parallel cyclic convolution algorithm with no interprocessor communication. We selected this algorithm because it strongly matches our particular cluster architecture and, at the same time, is amenable to a mixed-mode (MPI-OpenMP) implementation as well as to a direct MPI implementation. In the past, different variants for this algorithm were developed\textsuperscript{5,6} and its use within different hardware implementations was proposed\textsuperscript{7,8,9}. We have found no studies regarding the implementation of this algorithm in cluster architectures. While further benchmarks and scalability studies will be reported elsewhere, in this paper we are focusing in a MPI-only versus a mixed-mode (MPI-OpenMP) parallel implementation, including performance and scalability studies, carried out in our 16-node, 64 processor cluster.

Based on the prime factor decomposition of the signal length this algorithm, which is based on a block diagonal factorization of the circulant matrices, breaks a one-dimensional cyclic convolution into shorter cyclic sub-convolutions. The subsections can be processed, independently, either in serial or parallel mode. The only requirement is that the signal length, N, admits at least an integer, \( r_0 \), as a factor; \( N = r_0s \). The Argawal-Cooley Cyclic Convolution algorithm, has a similar capability but requires that the signal length can be factored into mutually prime factors; \( N = r_0s \) with \( (r_0,s) = 1 \). Since the block pseudocirculant algorithm is not restricted by the mutually prime constraint, it can be implemented recursively using the same factor\textsuperscript{6,7}. The parallel technique, compounded with a serial recursive approach in each parallel subsection, provides a sublinear increase in performance versus the serial-recursive implementation in a single core.

For our scalability studies we first used 16 cores at four cores per node. We accessed the 16 cores directly using MPI and then, for the hybrid approach, we accessed four nodes using MPI followed by using OpenMP to access the four cores in each node. We repeated the computation for several signal lengths. We then used 64 cores. We accessed the 64 cores directly using MPI and then, for the hybrid approach, we accessed 16 nodes using MPI followed by using OpenMP to access the four cores in each node. Again, several signal lengths were used. At each parallel core the algorithm was run in a serial-recursive mode until the recursion became more expensive than directly computing the sub-convolution for the attained sub-length. The details of the serial-recursive implementation will be reported elsewhere.

We start by providing, as stated in the literature\textsuperscript{6}, the mathematical framework for the algorithm using a tensor product formulation. We then develop a block matrix factorization that includes pre-processing, post-processing and parallel stages. The parallel stage is defined through a block diagonal matrix factorization\textsuperscript{6}. The algorithm block diagram is clearly developed through a one-to-one mapping of each block to the algorithm block matrix formulation. We follow by stating the benchmark setup, benchmark results and conclusions.

**Algorithm Description**

Here we briefly describe the sectioned algorithm used for this benchmark as reported in the literature\textsuperscript{5,6}. In this particular implementation, the sub-convolutions are performed using a recursive time domain-based cyclic convolution algorithm in order to avoid round-off errors. The
The proposed algorithmic implementation does not require communication among cores but involves initial and final data distribution at the pre-processing and post-processing stages. Cyclic convolution is a established technique broadly used in signal processing applications. The Discrete Cosine Transform and the Discrete Fourier Transform, for example, can be formulated and implemented as cyclic convolutions.\textsuperscript{6,7} In particular, parallel processing of cyclic convolution has potential advantages in terms of speed and/or access to extended memory, but requires breaking the original cyclic convolution into independent sub-convolution sections. The Agarwal-Cooley cyclic convolution algorithm is suitable to this task but requires that the convolution length be factored into mutually prime factors, thus imposing a tight constraint on its application\textsuperscript{3}. There are also multidimensional methods but they may require the lengths along some dimensions to be doubled\textsuperscript{3}. There other recursive methods, which are also constrained in terms of signal length\textsuperscript{3}. When the mentioned constraints are not practical, this algorithm provides a complementary alternative since it only requires that the convolution length be factorable\textsuperscript{5,6}. This technique, depending on the prime factor decomposition of the signal length, can be combined with the Agarwal-Cooley algorithm or with other techniques.

By conjugating the circulant matrix with stride permutations a block pseudocirculant formulation is obtained. Each circular sub-block can be independently processed as a cyclic sub-convolution. Recursion can be applied, in either a parallel, serial or combined fashion, by using the same technique in each cyclic sub-convolution. The final sub-convolutions at the bottom of the recursion can be performed using any method. The basic formulation of the algorithm as stated in the literature is as follows\textsuperscript{6},

The modulo-N cyclic convolution of the length-N sequences $x[n]$ and $h[n]$ is defined by,

$$y[n] = \sum_{k=0}^{N-1} x[k] h[((n - k) \mod N)]$$

which can be formulated in matrix form as,

$$y = H_N x$$

where $H_N$ is the circulant matrix,

$$H_N = \begin{bmatrix}
    h_0 & h_{n-1} & h_{n-2} & \cdots & h_2 & h_1 \\
    h_1 & h_0 & h_{n-1} & \cdots & h_3 & h_2 \\
    h_2 & h_1 & h_0 & \cdots & h_4 & h_3 \\
    \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\
    h_{n-2} & h_{n-3} & h_{n-4} & \cdots & h_0 & h_{n-1} \\
    h_{n-1} & h_{n-2} & h_{n-3} & \cdots & h_1 & h_0
\end{bmatrix}$$
Given \( N = r_0s \), and conjugating the circulant matrix, \( H_N \), with stride-by-\( r_0 \) permutations we obtain,

\[
y = H_N x
\]

\[
P_{N,r_0} y = P_{N,r_0} H_N P_{N,r_0}^{-1} P_{N,r_0} x
\]

\[
H_{p_0} = P_{N,r_0} H_N P_{N,r_0}^{-1}
\]

The decimated-by-\( r_0 \) input and output sequences are written as,

\[
y_{r_0} = P_{N,r_0} y, \quad x_{r_0} = P_{N,r_0} x
\]

The conjugated circulant matrix has the form of a Block Pseudocirculant Matrix\(^6\), represented as \( H_{p_0} \). *Block Pseudocirculant Matrices* have the circulant sub-blocks above the diagonal multiplied by a cyclic shift operator. Equation (5) can be re-written using (6) and (7) as,

\[
y_{r_0} = H_{p_0} x_{r_0}
\]

The Block Pseudocirculant in (8) is written as\(^5,6\),

\[
y_{r_0} = \begin{bmatrix}
Y_0 \\
Y_1 \\
Y_2 \\
\vdots \\
Y_{r_0-2} \\
Y_{r_0-1}
\end{bmatrix} = \begin{bmatrix}
H_0 & S_{N/r_0}H_{r-1} & S_{N/r_0}H_{r-2} & \ldots & S_{N/r_0}H_{r-1} & S_{N/r_0}H_1 & X_0 \\
H_1 & H_0 & S_{N/r_0}H_{r-1} & \ldots & S_{N/r_0}H_2 & S_{N/r_0}H_1 & X_1 \\
H_2 & H_1 & H_0 & \ldots & S_{N/r_0}H_3 & S_{N/r_0}H_2 & X_2 \\
\vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\
H_{r_0-2} & H_{r_0-3} & H_{r_0-4} & \ldots & H_0 & S_{N/r_0}H_{r_0-1} & X_{r_0-2} \\
H_{r_0-1} & H_{r_0-2} & H_{r_0-3} & \ldots & H_1 & H_0 & X_{r_0-1}
\end{bmatrix}
\]

\( S_{N/r_0} \) is the *Cyclic Shift Operator Matrix* defined by,

\[
S_{N/r_0} = \begin{bmatrix}
0 & 0 & 0 & \ldots & 0 & 1 \\
1 & 0 & 0 & \ldots & 0 & 0 \\
0 & 1 & 0 & \ldots & 0 & 0 \\
\vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\
0 & 0 & 0 & \ldots & 0 & 0 \\
0 & 0 & 0 & \ldots & 0 & 1
\end{bmatrix}
\]
All sub-convolutions, represented by the sub-blocks in the block pseudocirculant matrix, can be processed in parallel. The cyclic shifts above the diagonal represent a circular redistribution of each sub-convolution result.

**Example:** For \( N = 4, r = 2, H_N = H_4 \), (5) becomes,

\[
P_{4,2}y = P_{4,2}
\begin{bmatrix}
y_0 \\
y_1 \\
y_2 \\
y_3
\end{bmatrix}
= P_{4,2}
\begin{bmatrix}
h_0 & h_2 & h_1 & h_0 \\
h_3 & h_0 & h_1 & h_2 \\
h_2 & h_1 & h_0 & h_3 \\
h_1 & h_2 & h_1 & h_0
\end{bmatrix}
\begin{bmatrix}
x_0 \\
x_1 \\
x_2 \\
x_3
\end{bmatrix}
\tag{11}
\]

Where \( P_{4,2} \) is a stride-by-2 permutation matrix,

\[
P_{4,2} = P_{4,2}^{-1} =
\begin{bmatrix}
1 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 \\
0 & 1 & 0 & 0 \\
0 & 0 & 0 & 1
\end{bmatrix}
\]

Applying (6) and (7), (11) becomes,

\[
y_{10} = y_2 =
\begin{bmatrix}
y_0 \\
y_1 \\
y_2 \\
y_3
\end{bmatrix}
= P_{4,2}
\begin{bmatrix}
h_0 & h_2 & h_1 & h_0 \\
h_3 & h_0 & h_1 & h_2 \\
h_2 & h_1 & h_0 & h_3 \\
h_1 & h_2 & h_1 & h_0
\end{bmatrix}
\begin{bmatrix}
x_0 \\
x_1 \\
x_2 \\
x_3
\end{bmatrix}
\tag{12}
\]

where the circulant matrix \( H_4 \) in (11) has become blocked in a pseudocirculant fashion and can be written as,

\[
y_{r0} = y_2 =
\begin{bmatrix}
Y_0 \\
Y_1
\end{bmatrix}
= H_0 + S_2H_1H_0X_0
\tag{13}
\]

where \( S_2 \) is the cyclic shift operator matrix,

\[
S_2 =
\begin{bmatrix}
0 & 1 \\
1 & 0
\end{bmatrix}
\]

and the blocked cyclic sub-convolutions are defined as,
Further examples can be found in the literature \(^6,^7\). It is clear that each sub-convolution can be separately processed, followed by a reconstruction stage to provide the final result. Parallel algorithms can be developed by factorization of the Block Pseudocirculant matrix shown in (9), (12) and (13) into diagonal blocks. The general tensor product formulation for a block diagonal factorization of the Block Pseudocirculant Matrix is \(^6,^7\),

\[
y_{r_0} = R_{r_0} (A_{r_0} \otimes I_{N/r_0}) D H_{r_0} (B_{r_0} \otimes I_{N/r_0}) x_{r_0}
\]

(15)

where \(x_{r_0}\) and \(y_{r_0}\) are the decimated-by-\(r_0\) input and output sequences and \(A_{r_0}\) and \(B_{r_0}\) are the post/pre-processing matrices, which are determined by each particular factorization. Matrix \(R_{r_0}\) take account of all the cyclic shift operators and \(D H_{r_0}\) is the diagonal matrix with blocked subsections that are suitable to be implemented in parallel. At each parallel sub-section the same algorithm can be applied again. The general form of these matrices for different variants of this algorithm can also be found in the literature \(^6,^7\). We are using the simplest approach, which is the Fundamental Diagonal Factorization \(^6\) that implies \(r_0^2\) independent parallel sub-convolutions that uses straight forward pre-processing and post-processing stages. This formulation is the best match to our target architecture and it was used to benchmark the mixed-mode, MPI-OpenMP, implementation versus the MPI implementation.

For an N-point cyclic convolution, where \(N = r_0 s\), a radix-2 \((r_0 = 2)\) fundamental diagonalization gives \(r_0^2 = 4\) diagonal subsections. Equations (5) and (15) with \(r_0 = 2\) can be written as,

\[
y_2 = \begin{bmatrix} Y_0 \\ Y_1 \end{bmatrix} = \begin{bmatrix} H_0 & S_{N/2} H_1 \\ H_1 & H_0 \end{bmatrix} \begin{bmatrix} X_0 \\ X_1 \end{bmatrix}
\]

(16)

\[
\begin{bmatrix} Y_0 \\ Y_1 \end{bmatrix} = \begin{bmatrix} I_{N/2} \\ 0 \\ 0 \\ I_{N/2} \end{bmatrix} R \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & H_0 \end{bmatrix} \otimes I_{N/2} \begin{bmatrix} H_0 & 0 & 0 & 0 \\ 0 & H_1 & 0 & 0 \\ 0 & 0 & H_0 & 0 \\ 0 & 0 & 0 & H_0 \end{bmatrix} \otimes I_{N/2} \begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} X_0 \\ X_1 \end{bmatrix}
\]

(17)

The block diagram realization of the algorithm embodied in (17) is shown in Fig. 1. The parallel realization is given by the diagonal matrix; \(D\), while the pre-processing distribution is given by matrix \(B\) and the post-processing cyclic shifts and sums are given by matrices \(R\) and \(A\). Note that, following (5), (6) and (7), \(X_0\) and \(X_1\) are the decimated subsections yielded by the stride permutation applied to the input sequence. We also need to apply the inverse stride permutation to the output vector, formed by \(Y_0\) and \(Y_1\) in order to reorder the output and give the final result. In the present implementation the parallel cyclic sub-convolutions are performed at the cores/nodes and the pre-processing and post-processing stages are computed at the master node.
The implementation in each core is time domain-based and follows a serial recursive approach, to be discussed elsewhere using a variant of this algorithm termed Basic Diagonal Factorization (BDF)⁶.

![Diagram](image)

Figure 1. Parallel cyclic convolution realization based on a **Fundamental Diagonal Factorization** \( (r_0 = 2) \) of the Block Pseudocirculant Matrix. Flow graph stages B, D, A, R directly map to same name matrices in equations (15), (17).

**Hybrid, MPI – OpenMP, versus a Direct MPI Implementation**

MPI is capable of transmitting the data to the nodes but it is not necessarily efficient distributing the data to the cores inside the multi-core nodes. OpenMP, on the other hand, is optimized for shared memory architectures. In theory, by using MPI for distributing the data to the nodes, and then using OpenMP for distributing the data inside the multi-core nodes, we can match the architecture of our algorithm to the architecture of the cluster.

Usually, there is a hierarchical model in mixed-mode implementations: MPI parallelization occurs at the top level and OpenMP parallelization occurs at the low level, as it is shown in Figure 2. This model¹⁰ matches the architecture of our cluster. For example, codes that do not scale well as the MPI processes increase, but scale well as OpenMP processes increase, may take advantage of using mixed-mode programming. Nonetheless, previous studies show that the mixed-mode approach is not necessarily the most successful technique on SMP systems and should not be considered as the standard model for all codes¹⁰.

If the parallel subsections shown in Figure 1 are cores, the cyclic sub-convolutions are directly computed at each core. If the parallel subsections are multi-core nodes, the algorithm can be applied again in a parallel-recursive fashion at each node to redistribute data among the cores within the node. The cyclic sub-convolutions are again computed directly at each core. The former scheme lends itself to a direct MPI implementation while the latter scheme is amenable to a hybrid, MPI-OpenMP implementation as shown in Figure 2. Note that, in all cases, the final sub-convolutions at the cores can be performed through a serial-recursive approach using the **BDF variant**⁶ of this algorithm. The serial-recursive process stops at some recursive step when continuing the recursion becomes more expensive than directly computing the sub-convolution. The techniques used to tackle the serial-recursive approach will be reported elsewhere.
16-Processor Direct MPI Implementation

For the direct MPI implementation we partitioned the cyclic convolution into 16 parallel subsections of one-fourth the original convolution length using a radix-4 approach \((r_o = 4)\). This implementation was tackled using MPI where the data was distributed from the master node to 16 parallel cores. In each core a serial approach was used by recursive application of the same algorithm until the final cyclic sub-convolutions were performed using a straightforward, time-domain-based, convolution algorithm. We repeated the overall process for eight different signal lengths \((2^{15} \text{ to } 2^{22})\). The block diagram realization is shown in Figure 3.

16-Processor Hybrid, MPI-OpenMP Implementation

We then implemented a hybrid, MPI-OpenMP, approach. Using this technique, data of one-half the original length was distributed from the master node to four parallel nodes using MPI following the architecture in Figure 1 (Radix-2, \(r_o = 2\)). Then, under OpenMP, the structure in Figure 1 was applied again (Radix-2, \(r_o = 2\)) in a parallel-recursive fashion at each of the four nodes. The final processing length is one-fourth of the original signal length and both methods use 16 processors. In each core, as in the previous case, a serial-recursive approach is used to compute the final sub-convolutions. We repeated the overall process for eight different signal lengths \((2^{15} \text{ to } 2^{22})\). The block diagram realization is shown Figure 4.

64-Processors MPI versus Hybrid, MPI-OpenMP, Implementations

The procedure described in the past two paragraphs was repeated for 64 processors using MPI for the direct implementation (radix-8, \(r_o = 8\)). The hybrid implementation used MPI to distribute data among 16 multi-core nodes (radix-4, \(r_o = 4\)) followed by OpenMP to distribute data among the four processors within each node (radix-2, \(r_o = 2\)). In both approaches the data length, after parallel distribution, was one-eighth of the original signal length. As with the 16-processor implementation the parallel sub-convolutions in each core were computed using a serial-recursive approach. The block diagram for the direct implementation is similar to the one shown in Figure 3, but now distributing data to 64 cores using MPI. The block diagram for the hybrid implementation is similar to the one shown in Figure 4, but now we are distributing data to 16 multi-core nodes using MPI followed by further data distribution to the four cores within each node using OpenMP. We repeated both processes for 10 different signal-lengths \((2^{15} \text{ to } 2^{24})\).
Figure 3. MPI Implementation using Radix-4 (16-Core). Stages B, D, A, R directly map to the same name matrices in equations (15) and (17). Note that no node intercommunication occurs.
Figure 4. Mixed-Mode, MPI-OpenMP Implementation Recursively using Radix-2 (16-Cores).

Stages B, D, A, R in the signal flow graph map to matrices B, D, A, R in equations (15) and (17).
Hardware Setup

Hardware Setup: 16-Nodes, 64 Processors, Dell Cluster

The current benchmark was carried out using a 16-node cluster with multi-core nodes. Each node has two, dual-core processors for a total of 64 cores. The cluster is somewhat dated but still useful for comparative and scalability studies. The cluster hardware specifications are:

Master Node:
- Brand: Dell PowerEdge 1950
- Processor: 2 X Intel Xenon Dempsey 5050, 80555K @ 3.0 GHz (Dual-Core Processor). 667MHz FSB. L1 Cache Data: 2 x 16KB, L2 Cache: 2 x 2MB.
- Main Memory: 2GB DDR2-533, 533MHz.
- Secondary Memory: 2 X 36GB @ 15K RPM.
- Operating System: CentOS Linux 5.6 64-bit.

Compute Nodes:
- Brand: Dell PowerEdge SC1435
- Processor: 2 X AMD 2210 @ 1.8GHz (Dual-Core Processor). L1 Cache Data: 2 x 128KB, L2 Cache: 2 x 1MB.
- Main Memory: 4GB DDR2-667, 667MHz.
- Secondary Memory: 80GB @ 7.2K RPM.
- Operating System: Linux 64-bit.

Software: C++

Benchmark Results

a) Direct MPI versus Hybrid, MPI-OpenMP, Implementations: 4-Nodes, 16-Cores

Table 1. Speedup of direct MPI over the Hybrid Mode for 16 Cores. Signal Length 2^M.

<table>
<thead>
<tr>
<th>M</th>
<th>Hybrid Mode</th>
<th>MPI Mode</th>
<th>Speed Up</th>
</tr>
</thead>
<tbody>
<tr>
<td>15</td>
<td>0.1325</td>
<td>0.1225</td>
<td>1.081632653</td>
</tr>
<tr>
<td>16</td>
<td>0.555</td>
<td>0.3375</td>
<td>1.644444444</td>
</tr>
<tr>
<td>17</td>
<td>1.0425</td>
<td>1.0575</td>
<td>0.985815603</td>
</tr>
<tr>
<td>18</td>
<td>3.59</td>
<td>2.775</td>
<td>1.293693694</td>
</tr>
<tr>
<td>19</td>
<td>9.1125</td>
<td>8.03</td>
<td>1.134806974</td>
</tr>
<tr>
<td>20</td>
<td>25.81</td>
<td>22.1925</td>
<td>1.16300552</td>
</tr>
<tr>
<td>21</td>
<td>75.305</td>
<td>64.1225</td>
<td>1.174392764</td>
</tr>
<tr>
<td>22</td>
<td>223.218</td>
<td>188.06</td>
<td>1.186950973</td>
</tr>
</tbody>
</table>
Figure 5. Speedup of Direct MPI over the Hybrid Mode for 16 Cores. Signal Length $2^M$.

b) Direct MPI versus Hybrid, MPI-OpenMP, Implementations: 16-Nodes, 64-Cores

Table 2. Speedup of Direct MPI over the Hybrid Mode for 64 Cores, Signal Length $2^M$.

<table>
<thead>
<tr>
<th>$M$</th>
<th>Hybrid Mode (Seconds)</th>
<th>MPI Mode (Seconds)</th>
<th>Speedup</th>
</tr>
</thead>
<tbody>
<tr>
<td>15</td>
<td>0.11</td>
<td>0.25</td>
<td>0.44</td>
</tr>
<tr>
<td>16</td>
<td>0.305</td>
<td>0.2575</td>
<td>1.184466019</td>
</tr>
<tr>
<td>17</td>
<td>0.7775</td>
<td>0.665</td>
<td>1.169172932</td>
</tr>
<tr>
<td>18</td>
<td>1.97</td>
<td>1.545</td>
<td>1.275080906</td>
</tr>
<tr>
<td>19</td>
<td>4.2825</td>
<td>3.835</td>
<td>1.116688396</td>
</tr>
<tr>
<td>20</td>
<td>10.58</td>
<td>9.7975</td>
<td>1.079867313</td>
</tr>
<tr>
<td>21</td>
<td>28.69</td>
<td>26.4</td>
<td>1.086742424</td>
</tr>
<tr>
<td>22</td>
<td>80.1725</td>
<td>72.3175</td>
<td>1.108618246</td>
</tr>
<tr>
<td>23</td>
<td>233.21</td>
<td>203.01</td>
<td>1.148761145</td>
</tr>
<tr>
<td>24</td>
<td>736.34</td>
<td>597.607</td>
<td>1.232147548</td>
</tr>
</tbody>
</table>
We can see that as the signal length increases the direct MPI implementation becomes faster than the mixed-mode implementation, Figure 5, 6, Table 1, 2. The fact that the hybrid implementation is slower than the MPI implementation appears to be mainly because of the compute time at the sub-convolution stage within the nodes. Considering that the absolute time is slightly larger for the mixed-mode approach, and looking to Figures 7, 8, it can be concluded that the transfer times are similar for both methods. There could be some particularity in the OpenMP portion of the implementation that is causing a relative slowdown in the compute time for the mixed-mode approach. If such a problem could be detected, further code optimization may be needed if we were to improve the performance of the hybrid implementation versus the MPI-only approach.

c) Transfer Time Studies

Figure 7 shows the transfer times for the direct MPI implementation. The transfer time percentage is slightly less for the hybrid MPI-OpenMP parallel implementation shown in Figure 8. The percentage of transfer time, with respect to the total time, increases as we increase the number of processors. This is because the total time for a fixed signal size decreases as more parallel processors are added. For a fixed number of processors as the signal length increases the percentage of transfer time, compared to the total execution time, decreases. Note that the mixed-mode approach needs a minimum of 16 parallel processors using the proposed formulation. The four-processor MPI implementation is shown just for reference since we are comparing the 16 and 64-processor implementations using the MPI approach versus the mixed-mode approach.
Figure 7. Transfer Time Percentage of Total Execution Time versus Number of Processors (MPI Implementation)

Figure 8. Transfer Time Percentage of Total Execution Time versus Number of Processors (Mixed-Mode Implementation)
d) Absolute Speedup of a Parallel Implementation versus a One-Core Serial Implementation.

For completeness we provide the speedup of the MPI-only parallel-recursive implementation versus the direct serial-recursive implementation, Fig. 9, Fig. 10. Since the parallel technique proposed in this paper uses a serial-recursive approach to perform each parallel sub-convolutions it can be benchmarked against the performance of a serial-recursive implementation in a single core. This benchmark shows that, as the signal length increases, the use of additional parallel processors increases the speedup of the parallel approach versus the serial approach. The increase in performance provided by this parallel scheme is sublinear.

![Graph](image)

**Figure 9.** 16-Core Parallel-Recursive versus a One-Core Serial-Recursive Cyclic Convolution Implementation Speedup using C++

![Graph](image)

**Figure 10.** 64-Core Parallel-Recursive versus a One-Core Serial-Recursive Cyclic Convolution Implementation Speedup using C++
Educational Component

The reported case-study represents an example of a complex algorithm that can be reformulated in order to more closely conform to the underlying target architecture. Since case-studies provide a focused framework for the application of engineering knowledge\textsuperscript{11}, we have provided sufficient amount of detail so it can be used as a point of departure for further research. It can also be used as an educational resource in algorithm manipulation, and/or further code development, regarding the choice of mixed-mode (MPI-OpenMP) versus direct MPI implementations.

Future Work

As a future work we plan to benchmark for memory efficiency, where the mixed-mode approach could have an advantage, and to advance our code optimization efforts seeking an increased performance for both techniques.

Conclusions

By using MPI for distributing data to the nodes and then using OpenMP to distribute data among the cores inside each node, we matched the architecture of our algorithm to our target cluster architecture. Each core processes an identical program with different data using a single program multiple data (SPMD) approach. All pre and post-processing tasks were performed at the master node.

We found that the MPI implementation had a slightly better performance than the mixed-mode, MPI-OpenMP implementation. We established that the speedup increases very slowly, as the signal size increases, in favor of the MPI-only approach. Looking at Figure 3 and 4 it becomes clear that the MPI portion of both implementations should not suffer from load unbalancing problems.

The transfer time, as percentage of total time, is slightly less for the mixed-mode parallel implementation than for the MPI implementation. Since the total time for the hybrid approach is slightly higher, the transfer times for both approaches are basically the same. Given that the transfer times are similar, there could be some particularity in the OpenMP portion of the implementation that is causing a relative slowdown in the compute time versus the MPI approach. If such a problem can be detected further code optimization may be needed if we were to improve the performance of the mixed-mode implementation versus the MPI implementation.

When there are no unbalancing problems or other issues in the MPI portion of the algorithm the direct MPI option may be adequate\textsuperscript{10}. This could account for the slightly less performance exhibited by the mixed-mode approach despite its tight match to the target architecture. This underscores the fact, as stated in the literature\textsuperscript{10}, that the mixed-mode approach is not necessarily the most successful technique on SMP systems and should not be considered as the standard model for all codes.
As expected, when the signal length increases the use of additional parallel processors sublinearly increases the performance of the parallel-recursive approach versus the serial-recursive implementation in a single core.

The presented case-study can be used as point of departure for further research or can also be used as an educational resource in algorithm manipulation, and/or additional code development, regarding the choice of mixed-mode (MPI-OpenMP) versus direct MPI implementations. The authors will gladly provide supplementary information or resources.

Acknowledgement

This work was supported in part by the U.S. Army Research Office under contract W911NF-11-1-0180 and by Polytechnic University of Puerto Rico.

References

Using Interdisciplinary Game-based Learning to Develop Problem Solving and Writing Skills

Dr. Reneta Davina Lansiquot, New York City College of Technology

Reneta D. Lansiquot is Associate Professor of English and Assistant Director of the Honors Scholars Program where she earned her first degrees, an A.A.S. in Computer Information Systems and a B. Tech in Computer Systems, New York City College of Technology, City University of New York. She earned her Ph.D. in Educational Communication and Technology at New York University after completing her M.S. in Integrated Digital Media at Polytechnic University (now The Polytechnic School of Engineering, or NYU Engineering). Her mixed-methodology research focuses on interdisciplinary studies. She has presented her research at numerous national and international conferences in Austria, Canada, Greece, Japan, and Portugal and has published peer-reviewed book chapters and journal articles on technical writing, game design, virtual reality, and problem-solving across the disciplines. Her book is entitled Cases on Interdisciplinary Research Trends in Science, Technology, Engineering, and Mathematics: Studies on Urban Classrooms.

Dr. Ashwin Satyanarayana, New York City College of Technology

Ashwin Satyanarayana earned his M.S. and Ph.D. in Computer Science from the University at Albany, State University of New York. He has been a Research Software Developer at Microsoft, where he worked on Microsoft’s search engine Bing. His research interests include data mining, machine learning, data preparation, information theory, and applied probability with applications in real-world learning problems.

Dr. Candido Cabo, New York City College of Technology/CUNY

Candido Cabo earned the degree of Ingeniero Superior de Telecomunicacion from the Universidad Politecnica de Madrid in 1982, and a Ph.D. in Biomedical Engineering from Duke University in 1992. He was a post-doctoral fellow at Upstate Medical Center, State University of New York, and a research scientist in the Department of Pharmacology at the College of Physicians and Surgeons of Columbia University. In 2000, he joined New York City College of Technology, City University of New York (CUNY) where he is a Professor in the Department of Computer Systems Technology. Since 2005, he has been a member of the doctoral faculty at the CUNY Graduate Center. His research interests include computer science and engineering education and the use of computational models to understand and solve problems in biology.
Using Interdisciplinary Game-based Learning to Develop Problem Solving and Writing Skills

Abstract

Students majoring in computing and engineering fields generally perceive that courses in their major are not related to the general education (liberal arts and sciences) courses required for their degree. This separation prevents the transfer of skills between courses in the major and general education courses that could result in mutually beneficial synergies. To challenge their preconceptions and to help students develop connections between major and general education courses, we developed a learning community that links two courses in Computer Systems (one course is an introductory course to problem solving and computer programming, CS1, and the second course is an introduction to the field of computer systems, CS0) and English Composition (EG1).

In this paper we describe an innovative approach to the teaching of computing and writing to first-year students majoring in a Computer Systems degree at a college of technology. The theme of the learning community is the development of narratives (a plot or schematic structuring of temporal actions) and their implementation as a video game prototype. Common student learning objectives and general education student learning outcomes for our courses include: use creativity to solve problems; understand and navigate systems; work productively within and across disciplines; use the tools needed for communication, inquiry, creativity, and analysis; gather, interpret, evaluate, and apply information discerningly from a variety of sources; and communication in diverse settings and groups, using writing (both reading and writing), oral (both speaking and listening), and visual means.

In the English composition class, students write original video game narratives in groups; in their CS1 computer programming class students implement these stories using Alice, a computer programming environment that supports the creation of three-dimensional animations; and, in the CS0 survey course, students explore architectural and hardware issues to describe a possible game delivery platform. The concepts and skills introduced in the computer courses are contextualized by a problem (game design) that is relevant to students and connected to concepts and skills developed in the writing course. Moreover, traditional English composition is taught to connect to the computing courses that first-year students take. The common student assignment across the three courses in this learning community is a game design document which includes analysis (background and problem description, target audience, review of existing projects and media selection), design (user characteristics, goals and objectives, and description of the delivery platform), and project description (narrative of project design, review of relevant literature, flowchart of the entire project, and storyboards). When given the chance to work on a meaningful project of their own choosing, students collaboratively created video game prototypes by leveraging their problem-solving, programming, and writing abilities gained in these three courses.
1. Introduction

Students majoring in computing and engineering fields at our institution must take a combination of courses within their discipline (about two-thirds of the total credits required for a degree) and general education courses (liberal arts and sciences, comprising the remaining one-third of the total credits) in order to graduate. Each degree program has well-defined learning outcomes that are mapped to the learning outcomes of each individual course within the program. Students are expected to perceive the different courses required for graduation as part of a single whole (i.e., the degree program), and also to establish connections and synergies between the different components (courses) of the curriculum.

Unfortunately, the transfer of concepts and skills between courses is lacking. Students tend to perceive courses in their major and general education courses as unrelated to each other; as a consequence, they are not able to transfer skills between those courses. This lack of transfer is not a problem unique to our institution. In fact, there is abundant evidence that transfer of skills between courses is relatively rare.\(^2,^3,^6\) The problem occurs not only in the crossover between general education and technologically oriented courses, but even between different courses of the same type. The lack of transfer is likely due to multiple factors. Students may have forgotten some of the material learned in a previous course; students may not perceive the connections; students may see the connections but are unable to use the material in meaningful ways in a different context; or the pedagogical approach used by instructors may not be conducive to transfer.\(^3\)

Approaches used to facilitate transfer of learning include the use of reflective writings, contextualization of learning experiences, and application of learning to real life. Multiple strategies have been suggested to encourage transfer\(^3\): making the need for transfer of learning explicit to students, advising students to take courses in the appropriate sequence, emphasizing in each course the material that students need to transfer to other courses, “practicing” transfer by inviting guest lecturers, development of metacognitive skills, and reinforcing concepts by using them often and in different contexts. Regardless of the strategies used, it seems apparent that transfer of learning does not occur automatically and that curriculum and course design should intentionally emphasize the connection between courses to stimulate transfer.

EG1, English Composition, is required of all students at the college. Likewise, CS1, Problem Solving with Computer Programming, is a required course for first-year students, and with a companion course, CS0, Introduction to Computer Systems, is a prerequisite for all other computer courses in the major. These gateway courses lay a foundation, generally, for the rest of the academic courses in college, and specially, for more advanced work in computer programming, database, and networking courses. To make more obvious to students the connections between those three courses, and to facilitate transfer of skills between those courses and beyond, we have developed a learning community (LC) that links the three courses as part of our innovative approach to the teaching of computing and writing to first-year computer systems majors at a college of technology. The theme of the LC is programming narratives—that is, having students write narratives that they will then implement, using computer programming, as a video game prototype. We believe that the LC approach incorporates and builds on many of the suggestions in the literature on how to facilitate transfer; moreover, it makes a statement, early in
the students’ academic careers, about the importance of connecting courses in the major and those in general education so as to facilitate transfer. This LC also builds on previous research showing that introducing narrative elements into problem-solving courses improves student performance in general as well as in computer programming-related problem-solving skills.4,9

We begin this paper by introducing the concept of a first-year LC, along with the learning outcomes and objectives of the three courses which are part of our LC. Next we describe the common assignment shared between the three courses, followed by the evaluation criteria for the common assignment. We then present data comparing the performance of the LC students with other students who took the same courses but were not part of the LC. The paper concludes with a discussion of the findings.

2. First-Year Learning Community (LC)

A LC is a group of students who enroll in two or more courses, generally in different disciplines that are linked together by a common theme, in an academic semester.8 LCs are one of the ten high-impact educational practices recognized nationally to improve student persistence using data from assessment to increase retention.8 First-year students are enrolled in two or three courses where faculty work together to provide a supportive structure through collaborative learning experiences and peer mentoring fosters better academic achievement.

Our institution is one of the most racially, ethnically, and culturally diverse institutions of higher education in the northeast United States: 31.5% of our students are African American, 33.8% are Latino, 20% are Asian or Pacific Islanders, 11.3% are Caucasian, and 0.6% are Native Americans. The College’s fall 2013 enrollment was 16,861.

At the College, LC students are recruited randomly and then given a list of linked course options in which enrollment is first-come, first-served. Students enrolled in LCs at the College are provided with social and education networks to support learning. Activities include a Welcome Orientation, registration workshops, study rooms, a mid-semester social event, and participation in a peer program. Faculty members are trained to implement cooperative learning, alternative assessment in the classroom, cross-disciplinary writing assignments, and critical thinking activities. They also learn how to make use of the campus’s counseling, library, and other educational resources as well as how to incorporate technology in the learning process. We have implemented LCs at our institution for more than 10 years, and the academic performance of students participating in LCs reflects the national trends. When compared to the general population at the College, students in LCs earn higher GPAs, have higher retention rates, and demonstrate greater satisfaction1.

With these results in mind, we developed a LC entitled “Story-Telling in Role-Playing and Action-Adventure Games.” In this LC, students gain insight into how modern video games are designed (both hardware and software) using narratives. As part of a LC—which includes CS0 (Introduction to Computer Systems), CS1 (Problem Solving with Computer Programming) and EG1 (English Composition I)—students leverage the problem-solving, computer programming, and writing skills gained in these three courses to produce a videogame prototype. This prototype
includes a sample game world, characters, their interactions with their setting, and a narrative establishing non-linearity.

2.1 English Composition I (EG1)

At the College, a typical EG1 course does not have a video game narrative theme and students do not have the opportunity to implement their stories; however, both courses help students to develop their ideas by using rhetorical modes including narration, comparison/contrast, analysis, and argumentation.

In this composition course, students write a series of assignments that result in a coherent exposition or narrative (i.e., they relate a set of events that follow a logical and coherent direction so as to form a story). The first third of this fifteen-week semester course provides instruction on basic elements of a story, using examples from different literary and media genres. Students focus first on the elements of drama in plays and short stories, then explore examples of the plot structure of films, and are finally introduced to the idea of the mythic hero and the hero’s journey, which can be depicted as a circle. The explanation of the hero’s quest begins at the top of the circle with the Ordinary World and then moves counterclockwise through a series of stages: The Call to Adventure; The Refusal of the Call; Meeting the Mentor; Crossing the Threshold; Tests, Allies, and Enemies; Approach; The Ordeal; The Reward; The Road Back; The Resurrection; and Return with the Elixir—which brings the hero back to the Ordinary World. At this point, the connection to video game narratives is made and the importance of an engaging protagonist is emphasized.

During the middle five weeks of the semester, students use their knowledge of the hero’s journey first to write (individually) an original video game background story, and then to pitch their story idea to their classmates to choose the best narratives to implement as a computer program, and they then revise the selected story as a group. Students (again working individually) develop an engaging character side quest, including a rationale for the importance of such a quest to the protagonist as well as to the target audience of the game (see Appendix). The detailed assignment description is provided below:

- **Create the background story for a video game, including the protagonist and antagonist:** In a three-page narrative, explain what your central character wants and what the expertise of this hero is. Why is this character on this important quest? Be sure to describe the setting—the world of your game—clearly. Provide sufficient descriptions so that it is clear what the player needs to do to “win” your game. Your video game background story should end where game play begins. Attach, on a separate page, a well-developed paragraph describing the game play. Be sure to address the following: Who is your target audience? Why should the player care about the protagonist? Why is your story socially relevant or engaging to your proposed target audience? You will present your video game plot outline to the class, and the best ideas will be chosen for further development. Therefore, consider a rationale for your game or how you would “sell” the game idea to your target audience—in this case, your classmates.

- **Revise the selected video game background story with your group:** Refine this story and broaden its scope. Each group member should contribute one additional page. The final version of this collaborative story should be at least two full pages longer than the original
and should include screenshots from Alice (www.alice.org), a 3D programming environment.

- Development of a side-quest: In a three-page narrative, develop a character side quest based on your game world and the team of characters that you have created. How will game artifacts come into play? Explain how your character interacts with the world and what this character contributes to the team. Attach, on a separate page, a well-developed paragraph explaining why this side quest will be engaging to your audience. Also attach your concept map—that is, a diagram that maps the relationships among elements of the story.

Finally, during the last five weeks, students develop their game design document as a group.

2.2 Problem Solving with Computer Programming (CS1)

The second course of this LC is CS1, Problem Solving with Computer Programming. This course is designed to introduce the student to concepts of problem solving using constructs of logic inherent in computer programming languages, including procedural programming and object-oriented programming. The student learns the nature of problems, common solution approaches, and analysis techniques. During the first two weeks of the fifteen-week semester, the emphasis is on solving problems in a context known to the students—for example, navigation of mazes or games such as tic-tac-toe. Several computer programming constructs such as sequencing, selection structures, and repetition loops are introduced to solve various problems using pseudocode. During the next three weeks students solve problems with flowchart interpreters (Visual Logic). In the following eight weeks of the semester, students use Alice, which allows them to further develop their problem-solving skills. Alice is also used to introduce object-oriented programming concepts as the students create animations or interactive computer games. In the final two weeks of the semester, students are exposed to an IDE (Integrated Development Environment, such as NetBeans or Eclipse, which helps programmers to write, compile, and test computer programs) and basic Java programming. This exposure facilitates students’ transition from the real objects that they have created and manipulated in Alice (i.e., characters in the game world) to the more abstract objects (like the objects found in a graphic user interface window, such as buttons, textboxes, and labels) that they will manipulate in subsequent and more advanced programming courses.

The LC common assignment (described further in the section entitled “The Common Assignment: A Game Design Document” below) is a crucial component enabling students to achieve and reinforce the learning outcomes for this course. As part of that assignment, students implement the background story for a video game developed in the EG1 class as a computer program in Alice. The assignment is organized around several milestones, including: (1) preparation of a flowchart of the story; (2) creating the setting of the video game with Alice objects; (3) creating the characters for the video game using Alice objects; (4) implementing the characters’ interactions among themselves and with the setting, i.e., programming the story in Alice using methods and events; (5) developing individually a character side quest within the video game developed by the group; and (6) integrating the main story and the side quest. Through this assignment, students demonstrate their ability to solve problems with a computer, using constructs of logic inherent in computer programming languages, including procedural programming (sequencing, decision and repetition structures) and object-oriented programming (use of objects and their associated methods). Their work implementing the video game narrative
in *Alice* applies to a different context the computer problem-solving skills that they acquired earlier in the course using pseudocode and a flowchart interpreter (*Visual Logic*), and it provides preparation for and a transition to learning a more formal programming language such as Java.

Incorporating narrative elements in a problem-solving course builds on previous findings that the introduction of narrative elements into problem-solving courses improves student performance in general and programming-related problem-solving skills specifically. All sections of CS1 taught in our department now incorporate narrative elements in problem-solving computer courses through the use of *Alice*. The linking of the CS1 problem-solving course in an LC with EG1 further integrates narrative elements into computer problem-solving courses; this integration should result in improved and more transferable computer problem-solving skills.

### 2.3 Introduction to Computer Systems (CS0)

In this foundational course for Computer Systems, students engage in an overall inspection of the world of computing. As part of this course, students also learn introductory concepts related to the inner workings of the computer, such as operating systems, networks, and database systems. This overview of machine architecture, software development, data organization, ethics, computer security, and the theory of computing is presented to introduce students to the key threads that recur within other computer systems courses. Thus, one of the main goals of this course is to enable students to develop critical thinking skills by designing solutions to given problems using pseudocodes and algorithms.

During this class, as part of their game design document common assignment (see below), students conduct a review of relevant literature, develop pseudocode to guide them with the implementation in *Alice* of the video game prototype, and describe the delivery platform for their video game. These components of the common assignment are crucial to enable students to achieve and reinforce the learning outcomes for this course. As part of the assignment, students are asked to:

- **Review of existing projects and identify target audience**: Reviewing involves identifying what is already known, in order to bring together results from different studies and provide a starting point for students to start on their common gaming project. This review of other games will help in identifying trends and patterns in others’ research findings in the world of video games. As part of this assignment, students also identify the target audience for their particular game.

- **Media selection**: Media selection is the first step that students should consider after they come up with their video game narrative. As part of the CS0 course, students learn how video, images and audio are stored in a computer. Using this knowledge, they then address what multimedia resources they would need for their gaming project and what memory constraints these needs would present. They are also required to consider the resulting performance impact on the computer system.

- **Artificial intelligence (AI) in gaming**: As part of designing the game, students are required to complete a literature survey and identify how AI is used in their game. As part of the CS0 course, an introduction to AI is presented, including search strategies
used in AI (e.g., semantic trees), robotics, and natural language interpretation. Students are then required to address these concepts in their game design.

- **Pseudocode**: Once they have their concept map ready, students write pseudocode that describes all the concrete and sequential steps in their plot. This pseudocode may include control structures (e.g., IF/ELSE, WHILE) that would be used in the implementation of the game.

- **Delivery platform**: As part of the CS0 course, students are taught about the functions of operating systems and are presented with an overview of different operating systems. Students are required to identify the delivery platform for their game.

3. **The Common Assignment: A Game Design Document**

3.1 **Description of the Common Assignment**

The common assignment across the three courses is a game design document. The common assignment is a crucial component to enable students to achieve and reinforce the learning outcomes for the individual courses that are part of the learning community. Common student learning objectives and general education student learning outcomes related to this assignment include:

- Use creativity to solve problems
- Understand and navigate systems
- Work productively within and across disciplines
- Use the tools needed for communication, inquiry, creativity, and analysis
- Gather, interpret, evaluate, and apply information discerningly from a variety of sources
- Communicate in diverse settings and groups, applying written (both reading and writing), oral (both speaking and listening), and visual means

This game design document contains three sections: analysis, design, and project description. These three sections in turn encompass the following subsections; the course in which students will complete each subsection is noted in parentheses.

- **Analysis**: background and problem description (EG1); target audience, review of existing projects, and media selection (CS0).
  In this section students first introduce their game, providing a summary of the background story and side quests and identifying their ideal player. Students compare and contrast their game to similar games. Finally, they select a delivery platform (e.g., a specific game console or computer). The problem description is added later, after students complete a literature review on artificial intelligence and human-computer interaction (HCI) and formulate a thesis related to this game design project.

- **Design**: User characteristics, content analysis, goals and objectives (EG1); description of the delivery platform (CS0).
  This section of the document notes the player characteristics; students also learn characteristics of their chosen genre (e.g., action-adventure or role-playing games). Game design elements related to these player characteristics are highlighted. The goal and objectives of the game are also stated; for instance, a student may express an intention “to
offer players a chance to immerse themselves in a fantasy RPG [role-playing video game] that has a compelling plot while experiencing the classical game-play system that has endeared itself to the hearts of thousands of gamers in their childhood.” The hardware items needed to properly run the proposed game, such as specific memory requirements, graphics, and sound cards, are also presented here.

- **Project Description:** Narrative of project design, flowchart (CS1); review of relevant literature, pseudocode (CS0); concept map and storyboards (EG1). A step-by-step plan of how students plan to implement their game in *Alice* is the main component of this section. This includes their flowchart, pseudocode, concept map, key characters, and representative setting screen shots from *Alice*. In addition, each member of the group provides a summary of an article on AI or HCI, and all articles are synthesized to form a relevant literature review.

### 3.2 Assessment of the Common Assignment

A literature review assessment rubric is used to evaluate students’ understanding of current developments in the field of gaming, including their grasp of relevant research on AI and HCI. This rubric describes multiple domains and the measurement criteria for each one, including completeness of the literature review, critique of sources, synthesis of sources, writing style, and adherence to an appropriate documentation format. With regard to synthesis, the rubric measures students’ ability to evaluate and report on the main ideas. They are expected not to summarize each article, but rather to organize and present the information with emphasis on its relevance to the larger topic of gaming development. Synthesizing information is not summarizing each article, but rather organizing the information by how it relates to each main point of the larger topic.

A concept map assessment rubric is used to evaluate students’ understanding of the plot structure of narratives. This rubric measures important concepts and describes domains on multiple levels with regard to the breadth of net; embeddedness and interconnectedness with other concepts; and the use of descriptive links to succinctly and accurately describe all relationships. In addition, the rubric measures efficiency of the link construction in terms of how each link type is distinct from all others, clearly describing consistent relationships. Finally, it measures layout in terms of how one map is contained on a single page, whether it contains multiple clear hierarchies, how well it is laid out, and whether it provides a sufficient number of relevant examples with links.

A target audience assessment rubric is used to evaluate students’ understanding of their research on identifying the target market for their game. The rubric measures how succinctly the students have defined the reasoning for choosing their specific audience (e.g., experienced game console players vs. cellular game users), characterized the size of the available market (with factual citations from magazine articles or an industry analysts), and identified other competitors who address the same needs as this product as well as any known customers for this product.

Media selection should include a thorough description of how audio, video, and images are represented in the game. Details with regard to specific file formats and software used for multimedia are expected.
A delivery platform rubric is used to determine the students’ understanding of what type of gaming platform their game would need. This rubric measures how the design would consider the downloading and installation of the game on Windows, Macintosh and Linux computers.

Pseudocodes and flowcharts are evaluated on how they mimic the elements of the actual computer code, but with more focus on readability and less on technical requirements. Evaluation involves determining how the dimension variables and control structures are defined. Finally, we assess whether all the applicable tasks and elements from the narrative are contained in the pseudocode.

Students also prepare a narrative on the entire project, describing how they implemented their stories in *Alice* (implementation of the characters and setting of the story as programming objects, and their interactions using events, functions and methods). This section is evaluated based on whether the reader can reproduce what was done based on the description provided.

3.3 A Case Study

In the Appendix, we highlight assignments of one student group as a case study, providing a summary of the video game background story and side quests, a concept map of the entire game, an *Alice* screen shot showing some of the programming objects and methods used in their coding of the video game, and storyboards. All those elements were part of the group’s Game Design Document.

4. Assessment of Student Performance and Satisfaction

4.1 Assessment Goals

Institutional results on LCs at our College\(^1\) compare student success and satisfaction to the general student population, which makes it difficult to isolate the effect of our particular LC, on student success and performance in the EG1, CS1 and CS0 courses. Therefore, we had two assessment goals: to gain insight into the effect of our particular LC on students’ academic performance in EG1, CS1 and CS0, and to qualitatively measure student satisfaction with the LC educational approach.

4.2 Design Methods

To quantify student performance, we compared the grades of students in the three courses of our LC (\(n =22\)), with the grades of students taking the same three courses simultaneously but not as part of a LC in fall 2013 (\(n = 70\)). Fall 2013 was the first semester we implemented this LC with the approach described in the paper. All students taking those three courses were majors in our Computer Systems degree. For our analysis, we converted letter grades to the numeric quality points (0-4) used to calculate student GPAs. For the statistical analysis, group means differences were considered significant at the 0.05 level.

We qualitatively measured student satisfaction with the LC educational approach with a survey at the end of the fall 2013 semester.
4.3 Assessment Results

Figure 1 shows the means and standard deviations for students in the LC, and not in the LC (non-LC), for all three courses. One-way analysis of variance showed no differences between course means for students in the LC group (p > 0.05), but that there were differences between course means for students in the non-LC group (p = 0.001). Post-hoc multiple comparisons in the non-LC group showed that grades in EG1 were significantly lower than grades in CS1 (p = 0.001) and CS0 (p = 0.013). There were no differences between the grade averages for each individual course between the LC and non-LC groups.

The data in Figure 1 suggests two factors contributing to student performance: differences in performance between the computer and non-computer courses and whether students took those courses as part of a LC or not. To further study the interactions between course performance and LC, we performed two-way analysis of variance with repeated-measures. The LC factor had two levels (being part of the LC or not), and the course factor had three levels (the three grades for each student in the three courses, EG1, CS1 and CS0). The results showed that both the course factor (p = 0.005) and the interaction between the course and the LC (p = 0.023) are statistically significant. This suggests that differences in performance in the three courses are modulated by whether or not the student is taking those courses as part of a LC. These results reinforce the earlier analysis suggesting that the LC context helps students to improve their performance in EG1 to a level comparable to their performance in CS1 and CS0.
Figure 2. Results of a satisfaction survey of LC students (n=21).

Students in the LC were generally satisfied with the educational approach (Figure 2). About 67% agree or strongly agree with the statement “I enjoyed being part of the Learning Community,” and about the same percentage feel better prepared for other college classes and that they improved their writing skills.

4.4 Discussion of the Assessment Results

Based on this data we can conclude that students who are not part of a LC (i.e. the majority of our students) and majoring in Computer Systems perform significantly better in the computer courses (CS1 and CS0) than in the English composition course (EG1). In contrast, for students in the LC group, performance in the EG1 course increases to the level of performance achieved in computer courses. Therefore, we propose that the increase in performance is the result of the contextualization of the learning experience that occurs in the LC: students apply writing and narrative concepts and skills to problems which are relevant to their interests and to their major. A student satisfaction survey (Figure 2) shows that students are generally satisfied with the LC educational approach. Those results are in line with institutional results on other LCs at our College.1

A number of factors should be considered when interpreting our results. In our analysis of student performance we have used final grades as an estimator of performance in the different courses. That is necessarily a summative indicator of performance in different types of assessments, including tests and quizzes, projects, homework assignments and student class participation. Moreover, LC courses and non-LC courses were not taught by the same instructors. Therefore, it is possible that there is some variability on grading styles by different instructors, even in courses like EG1, which have uniform finals. Further studies will be necessary to understand the effect of all those factors in our results, and to assess and compare performance in more specific concepts and skills both in writing and computing for students taking those courses in the context of a LC and outside a LC.
In this paper we describe an innovative approach to the teaching of computing and writing to first-year students majoring in a Computer Systems degree at a college of technology serving mostly underrepresented minorities. We provide a framework to link introductory courses in computer systems, problem solving with computer programming, and English composition in a learning community. The theme of the learning community is the development of video game narratives and their implementation as a video game prototype using computer programs, and it provides a mechanism for concept and skill transfer between the three courses. Students in the LC develop video game narratives in the English composition course, and they implement video game prototypes in the computer courses while acquiring computer concepts and skills that will be the foundation for other courses in their major. A deliverable for all students in the LC is a game design document that is a crucial component enabling students to achieve and reinforce the learning outcomes for the three courses.

Our students tend to perceive courses in their major and general education courses as unrelated to each other and, as a consequence, they are not able to transfer skills between those courses. One of the goals of the pedagogical approach described here is to make more obvious to students the connections between the three foundational courses in the LC, contextualizing the learning experience to facilitate transfer of skills between those courses and beyond: by providing learners with meaning, concepts are embedded within a web of related concepts. Concepts with numerous connections to other concepts have greater meaning and can act as a recall cue for connected concepts. We believe that the LC approach incorporates and builds on many of the suggestions in the literature on how to facilitate transfer. It also makes a statement, early in the students’ academic careers, about the importance of connecting courses in the major and those in general education so as to facilitate transfer.

A second goal of our study is to investigate the effect of linking computing and writing courses on student performance in those courses. Our results show that students majoring in Computer Systems perform significantly better in introductory computer courses than in English composition courses. This may be a consequence of the students’ view that computer courses and general education courses are unrelated, and that the computer courses are more “important” for their degree. However, when the English composition course is linked with introductory computer courses in a LC, students’ performance in English composition improves to the same level as their performance in the computer courses. It is possible that the intentional interdisciplinary contextualization of learning that occurs in the LC helps students making connections between writing and computer courses and facilitates transfer of concepts and skills with the consequent improvement in academic performance. An earlier study by Goldfine (with a different student population) showed that students with advanced skills in computer programming do not naturally make a connection between writing code and writing English, unless that connection is intentionally emphasized by the instructor, further illustrating that knowledge transfer between disciplines is rare and it must be intentionally emphasized by instructors.

While the pedagogical approach described here, using LCs to link computer and writing courses, seems to be beneficial for our student population, particularly underrepresented minorities, we
are not claiming that our results are general and can be easily extrapolated to other student populations and institutions. Further studies in other institutions will need to be carried out to determine how this approach might work in a different context. Moreover, it should be noted that LCs could be focused in different domains (e.g. green energy application, or integrating data from social media, etc.), which could provide students with other interdisciplinary experiences which may also pique their interest.

6. Conclusions

We can conclude: 1) Intentional interdisciplinary approaches to writing and problem solving allow students to purposefully connect and integrate knowledge and skills from across the disciplines. This integration will support students in solving problems; synthesizing and transferring knowledge across disciplines; becoming flexible, reflective thinkers who are comfortable with complexity; and thinking critically, communicating effectively, and working collaboratively, skills that will prepare our students to be lifelong learners in their careers; 2) Students majoring in Computer Systems perform better in introductory computer courses than in English composition courses; 3) Linking English composition courses with computer courses in the interdisciplinary context of a LC results in the improvement of students’ performance in English composition to the same level of their performance in the computer courses.

7. References


Acknowledgements

We would like to thank Learning Communities Director, Estela Rojas, for providing general data on the program at the College as well as Coordinated Undergraduate Education (CUE) First-Year Program Director, Lauri Shemaria-Aguirre for taking time out of her busy schedule to meet
with us to discuss the recruitment process and support structure for learning communities. Moreover, we would also like to thank Peer Mentors, Tamrah Cunningham, Elaine Green, and Walter Rada, all former students in our learning community, for helping students adjust to college life. In addition, we would like to extend our gratitude to the Assessment and Institutional Research (AIR) office for providing requested data. Finally, we would like to thank Hong Li, Chair of the Computer Systems Technology Department, and Nina Bannett, Chair of the English Department, for their support.

Appendix

Below is a summary of student Noel Melendez’s video game background story *Schrödinger’s Workshop*, which was chosen by his classmates to be further developed and implemented in Alice. With help from Noel’s group members, Tykila McCray, Hilario Salas, and Olajide Odunaikhe, his original background story was revised and each member of the group wrote side quest narratives, which is also summarized below, and represented in a sample concept map (Figure A1). References to corresponding screenshots from Alice have been added throughout the summary for purposes of illustration.

*Schrödinger’s Workshop*
Noel Melendez, Tykila McCray, Hilario Salas, and Olajide Odunaikhe

During the prologue of the video game prototype called Schrödinger’s Workshop, the player learns that he or she is in control of the main protagonist, Lloyd Deteget, a teenage boy genius. The name “Lloyd” is a play on the word “alloy,” or the solution of two or several metals; the last name, Deteget, means “to unveil” in Latin. By following Lloyd on a typical afternoon, the player learns that he is studying at the most prestigious engineering college in the city. He or she also learns that Lloyd has a habit of leaving the safety of his home to collect parts and metal scrap for his strange inventions. The lack of parental supervision throughout the prologue suggests that Lloyd is an orphan.

When our hero takes a brisk walk outside, the player sees that the world of Schrödinger’s Workshop is a vast utopia contained in a tight metropolitan setting, littered with advanced technologies such as flying vehicles, hover cars, and brilliant lights (see Figure A2). The city is protected by a large, cascading metal wall that shields it from the mysterious, foggy environment lying beyond it. The story line explains that, if someone were to exit the city’s safety and venture into the fog, he or she would be snatched by men robed in white, never to be seen again. This myth has been passed down from generation to generation, but Lloyd has always thought that it was a hoax and wants to explore the outside world.

On his way to the junk yard in search of scrap metal, Lloyd finds that his headphones are malfunctioning. Soon radio transmissions are interrupted and LCD screens displayed in window shops flicker violently. Lloyd hears a message announced by a disappointed female voice: “We are with heavy hearts today as bring this news to you. We are pulling the plug; this experiment was unsuccessful.” The sun suddenly dims, all technological devices cease to function, and vehicles drop from the sky like flies. The city is now in darkness, except for fires from the
quickly accumulating accidents (see Figure A3). One falling aircraft damages the city wall, creating an escape opportunity for Lloyd, who ventures out into the fog (see Figure A4).

Outside the city, the terrain seems bouncy, plastic as if manufactured, and unnaturally flat. Eventually Lloyd encounters a large, intimidating wall that stretches in all directions, with an entrance to a building called the Cypress Laboratory, which will be the main focus of the game (see Figure A5). Peering inside the laboratory, Lloyd sees men and women robed in white, scurrying around and recording data from LCD screens that litter the cold and sterile halls.

It turns out that the laboratory has been conducting an experiment on Lloyd’s town, which they called Schrödinger’s Workshop. The “sun” was an artificial source of light, and the city has been observed for several centuries through a series of monitors in the laboratory until the power was cut off.

The game has several side quests. Each one has an effect on the game and also reinforces the idea that the player has control over the fate of many people, not just Lloyd. For example, in “Dean’s Lament,” Lloyd visits his alma mater and finds that the college dean, named Hunnivant, is in physical peril. Hunnivant reveals to Lloyd that he was involved with the man responsible for conducting the grand experiment on Lloyd’s town. The player must help the dean to receive medical attention and press him for information. Whether the player ignores Hunnivant or helps him with his dilemma influences subsequent results, including the difficulty level of the game’s final stage. Other side quests introduce Lloyd’s parents (who themselves had traveled through the fog and met each other Cypress Laboratory) and involve Lloyd’s attempt to save a neighbor family who had contributed to his life.

As the game’s final stage approaches, the player faces a fork in the road. What should Lloyd do about the female voice, “Sophia,” that has plagued him since his journey began? If the player chooses to listen to Sophia, she will guide Lloyd to the person responsible for the experiments and become a valuable ally. Her guidance makes game play easier for the player, and Sophia also describes the city’s past and explains prior experiments that were considered failures. Alternatively, the player can choose to ignore Sophia and try to find the answers to Lloyd’s questions without her help, providing a more challenging game experience.
Figure A1. This is a student concept map that illustrates the aforementioned “Sofia” side quest.

Figure A2. This is the opening of the game. The camera will pan through Lloyd’s world as he carries out his normal day. The giant wall can be seen in the background of this image.
Figure A3. The “accident” occurs and technology comes to a halt. It is dark and smoke-filled, and the only light comes from the fires that spread throughout the city.

Figure A4. Alice screen shot showing the programming objects and methods used. Lloyd assesses damage up close. He is paralyzed and tries to reason with himself; this is Lloyd’s refusal of the call.
Figure A5. This image is the end of the prologue in the game. The student, Noel Melendez, used Alice’s fog function and varied subparts to create a convincing laboratory gate.
Transition from Concepts to Practical Skills in Computer Programming Courses: Factor and Cluster Analysis

Dr. Candido Cabo, New York City College of Technology/CUNY

Candido Cabo earned the degree of Ingeniero Superior de Telecomunicacion from the Universidad Politecnica de Madrid in 1982, and a Ph.D. in Biomedical Engineering from Duke University in 1992. He was a post-doctoral fellow at Upstate Medical Center, State University of New York, and a research scientist in the Department of Pharmacology at the College of Physicians and Surgeons of Columbia University. In 2000, he joined New York City College of Technology, City University of New York (CUNY) where he is a Professor in the Department of Computer Systems Technology. Since 2005, he has been a member of the doctoral faculty at the CUNY Graduate Center. His research interests include computer science and engineering education and the use of computational models to understand and solve problems in biology.
Transition from Concepts to Practical Skills in Computer Programming Courses: Factor and Cluster Analysis

Abstract

Computer programming courses are gateway courses with low passing grades, which may result in student attrition and transfers out of engineering and computer science degrees. Barriers to success include a good understanding of programming concepts and the ability to apply those concepts to write viable computer programs.

In this paper, we analyze the determinants of the transition from concepts to skills in computer programming courses using factor and cluster analysis. The purpose of this study is to answer the following questions related to computer programming teaching and learning: 1) Which are the correlations and interdependencies in student understanding of different computer programming concepts?; 2) Which are the cognitive challenges that students find when learning programming concepts?; 3) How the understanding of different programming concepts relate to practical skills in computer programming; 4) What determines a successful transition from understanding the concepts to the ability to write viable computer programs?

After several computer programming concept assessments in this first Java Programming course, we grouped the students' performance into seven different categories: assignment operators, repetition structures, selection structures, program design using methods, arrays, classes and Java syntax. Factor analysis identified two factors (components) grouping the interdependencies and correlations between programming concept categories. The first component correlated with the repetition and selection categories, and could be referred to as the “algorithmic” component. The second component correlated with the methods, arrays and assignment categories, and could be referred as the “structural” component. Student performance in conceptual categories related to the “algorithmic” factor was significantly better than in conceptual categories related to the “structural” factor. Cluster analysis showed that student performance in the “structural” conceptual component is predictive of the student’s ability to solve practical computer programming problems.

We conclude that a strong emphasis in the structural components of computer programming (i.e. program design using methods, use of the assignment operator, and use of data structures like arrays) is necessary for a successful transition from concepts to skills in computer programming courses.

1. Introduction

First year problem-solving and/or computer programming courses are gateway courses with low passing rates, which may account for student attrition and transfers out of computer science degrees. A number of challenges have been identified over the years by the computer science education community. It has been shown that an understanding of the problem domain to be solved by implementing a computer program should be a prerequisite for writing the computer program itself. Students’ inability to create a mental model of a given problem domain hinders their ability to develop problem-solving skills and write computer programs.
Another difficulty encountered by novice programmers is the syntax of computer programming languages, which is often overwhelming to students who get distracted from solving problems by the obscurity of the statements and program organization. This difficulty was recognized early in computer programming education, and different strategies including graphical languages and animations of program states were developed. One approach to increase success in first-year programming courses is a shift from teaching programming to teaching problem-solving skills. This approach has been successful and avoids some of the problems that hinder progress in the development of thinking skills that are important for computer programming. However, this approach has also been criticized because the translation of a problem solution to a computer program is not obvious. The challenges faced by students and educators in learning and teaching computer programming have been summarized in a recent review.

Following earlier findings in computer education research we require our students to take a problem-solving course before their first programming course. It has been showed that introducing narrative elements in pre-programming problem-solving courses (a pedagogical approach that has been called programming narratives) is more effective than traditional approaches using a full-fledge programming language as a tool to help students develop computer programming problem-solving skills. To facilitate the implementation of programming narratives we currently use Alice (www.alice.org), a programming environment that allows learners to create interactive animations while learning computer programming concepts. However, despite the benefit of using programming narratives to help students develop problem-solving skills, the transition from pre-programming problem-solving courses to courses where students should master a full-fledge programming language remains a challenge.

Two barriers to success in computer programming courses include a good understanding of programming concepts and the ability to apply those concepts to write viable computer programs. The purpose of this study is to answer the following questions related to computer programming teaching and learning: 1) Which are the correlations and interdependencies in student understanding of different computer programming concepts?; 2) Which are the cognitive challenges that students find when learning programming concepts?; 3) How the understanding of different programming concepts relate to practical skills in computer programming; 4) What determines a successful transition from understanding the concepts to the ability to write viable computer programs?

2. Methods

2.1 Participants

Our institution is one of most racially, ethnically, and culturally diverse institutions of higher education in the northeast: 31.5% of students are African American, 33.8% Latinos, 20% Asian/Pacific Islander, 11.3% Caucasian, and 0.6% Native Americans. At project initiation, the College spring 2013 enrollment was 16,208.
We report data from performance assessments from 22 students who took a Programming Fundamentals course in spring 2013. In this course, students use Java as the programming language of choice to help develop their conceptual and practical programming skills. For all students, this is the first programming course in their curriculum. However, before this course, all students had taken a Problem-Solving course in which they used pseudocode, flowcharting and Alice (www.alice.org) to learn basic procedural and object-oriented programming concepts. The goal of the Problem-Solving course is to teach programming concepts without the burden of learning a full-fledge programming language. However, basic Java programming is introduced in the last three weeks of the Problem Solving course to facilitate the transition to the Programming Fundamentals course.

2.2 Exploratory Factor Analysis

After several computer programming concepts assessments in the first Java programming course, we grouped students’ performance into seven different categories: assignment, repetition (for/while structures), selection (if/else structures), methods, arrays, classes and general syntax. Student performance in concepts and skills was assessed at three different times during the semester.

Exploratory factor analysis is a data reduction technique that aims at finding hidden correlations and interdependencies between different variables and grouping them in a number of overarching factors or components. Estimating the number of factors is tricky, and therefore to estimate the number of factors in the factor analysis we used different criteria. We used SPSS to extract the number of factors using the Kaiser-Guttman 6 (number of eigenvalues greater than one) and Cattell’s scree test 2. We also used FACTOR 12 to estimate the number of factors using Horn’s parallel analysis 7. The Kaiser-Meyer-Olkin measure of sampling adequacy was 0.674, above the suggested minimum of 0.5. Interpretation of the extracted factors can be made easier by orthogonal factor rotation. We used the varimax rotation method with Kaiser normalization.

2.3 Cluster Analysis

After the factors or components underlying the different conceptual categories have been identified, it is possible to derive scores for each student on each factor. We used hierarchical cluster analysis, using the Euclidian distance as a proximity measurement, to classify students’ factor scores and to group students in different clusters reflecting their responses to conceptual assessments. The number of clusters was determined by inspection of the dendrogram, a display representing visually the distances at which clusters are combined.
3. Results

3.1 Student Performance in Programming Concepts and Practical Skills

Figure 1 shows the individual performance of students in concepts and skills assessments (range 0-100). Performance in concepts can be mapped to the first two levels of Bloom’s taxonomy learning structure (knowledge and comprehension level), and performance in skills can be mapped to levels three and four (application and analysis level). We considered 70% (equivalent to a C grade) an acceptable (“passing”) performance in the assessments (vertical and horizontal dashed lines in Figure 1). Forty-one percent of the students did not have an acceptable performance either in concepts or in skills. (For reasons that will become clear later, we represent those students in Figure 1 with a solid circle.) Thirty-six percent of the students had an acceptable performance both in concepts and skills (students represented in Figure 1 with an unfilled circle). Twenty-three percent of students had an acceptable performance in concept assessments but not in practical skill assessments (students represented in Figure 1 with crossed squares).

![Figure 1. Overall student performance in programming concepts and skills (n = 22). Dashed lines mark an acceptable performance in computer programming concepts and skills assessments (70%). Unfilled circles represent students with acceptable performance in concepts and skills; crossed squares represent students with acceptable performance in concepts but not in skills; solid circles represent students with poor performance in concepts and skills. The solid line is the regression line (Skills = 1.34 *Concepts – 37, with r^2 = 0.71). Arrow shows the intercept of the regression line with the x-axis.](image-url)
Figure 1 illustrates the two barriers faced by novice computer programming students: a good understanding of programming concepts and the ability to apply those concepts to write viable computer programs. About 59% (13 out of 22) of students had an acceptable understanding of programming concepts. And about 61% (8 out 13) of the students with a good understanding of programming concepts were able to transfer that knowledge into practical programming skills.

Our analysis of Figure 1 relies in a somehow arbitrary threshold that marks the difference between satisfactory and poor performance in concepts and skills (70%, dashed lines in Figure 1). However, an analysis of the regression line (which is not dependent on arbitrary thresholds) can lead us to similar conclusions. Linear regression (solid line in Figure 1, $r^2 = 0.71$) indicates that performance in programming concepts is correlated with performance in practical programming skills (see also Figure 2 below). Also, most students perform better in concepts than in practical skills (shown by the positive intercept of the regression line with the concept-axis; arrow in Figure 1), indicating a barrier in students’ ability to apply concepts to the solutions of practical computer programming problems.

Figure 2. Average scores (range 0-10) in an assessment of computer programming concepts for students with acceptable performance in both concepts and skills (unfilled circles), for students with an acceptable performance in concepts but not in skills (crossed squares), and for students with poor performance in both concepts and skills (solid circles). Symbols are the same as in Figure 1.

There were no students with an acceptable performance in skills but poor performance in concepts (Figure 1). This suggests that without a good grounding in the understanding of the
concepts, it is very unlikely to develop acceptable practical programming skills. But, is there an acceptable (minimum) level of conceptual understanding to be able to develop acceptable practical programming skills? Figure 2 shows that students having an acceptable performance in both concepts and skills (unfilled circles) have a better overall performance in concepts than the other two groups (crossed squares, solid circles). This suggests that there might be a minimum level of conceptual understanding that is necessary in order to succeed in the development of practical programming skills. On the other hand, Figure 2 shows that the performance in the three groups is similar in some concepts (for example, events), while in other concepts the performance is markedly different (for example, methods and arrays). So, which are the more important concepts (or group of concepts) that students should master to develop acceptable practical programming skills? Is the understanding of all concepts equally important for the development of practical programming skills?

3.2 Exploratory Factor Analysis

To further understand the nature of students’ understanding of computer programming concepts, and the hidden correlations and interdependencies between programming concepts in those seven different categories, we performed exploratory factor analysis.

We grouped student performance in computer programming concepts assessments in seven different categories: assignment, repetition, selection, methods, arrays, classes and syntax. Factor analysis identified two factors or components grouping the interdependencies and correlations between programming concept categories. Figure 3 shows a plot of the factor loadings in the orthogonally rotated space with iso-loading factor lines, which illustrate the percent of correlation of the different conceptual categories with a given factor or component.

Factor loadings are the correlations between the different categories and the extracted factors (components), and therefore their value is between +1 and -1. The first component correlated with the repetition (correlation 0.9) selection (correlation 0.83), and classes (correlation 0.61) categories, and, since it involves concepts necessary to implement computer algorithms, it could be referred to as the “algorithmic” component. The second component correlated with the methods (correlation 0.89), arrays (correlation 0.81) and assignment (correlation 0.64) categories, and, since it involves concepts on data structures, data assignment and program organization, it could be referred to as the “structural” component. The correlation of the syntax category with any of the factors was < 0.60.

Student performance in concept categories related to the “algorithmic” factor was significantly better (paired t-test, p < 0.05) than student performance in concept categories related to the “structural” factor (Figure 4). This finding suggests that students have more difficulty with the structural components of computer programming which, therefore, need more attention and emphasis in the classroom.
Figure 3. Factor plot illustrating the correlations of the seven conceptual categories with the “algorithmic” and “structural” factors. Solid iso-correlation lines show the correlation of the different conceptual categories with the “structural” factor. Dashed iso-correlation lines show the correlation of the different conceptual categories with the “algorithmic” factor. For clarity, only three isolines (0.4, 0.6 and 0.8) are shown.

Figure 4. Performance in concept categories (range 0-100) related to the “algorithmic” factor (repetition, selection and classes) and in concept categories related to the “structural” factor (assignment, methods and arrays).
3.2 Cluster Analysis

After the factors or components underlying the different conceptual categories have been identified, it is possible to derive scores for each student on each factor. Figure 5 shows a plot of factor scores on the “algorithmic” and “structural” factors for all 22 students. Hierarchical cluster analysis of the factor scores indicated that students could be grouped in three clusters (dashed lines separate the different clusters in Figure 5). Students in the same cluster are similar with respect to their factor scores and are dissimilar to students in other clusters. Note that the clustering is mostly determined by the factor scores on the “structural” factor (Figure 5). For example, students with a factor score on the “structural” factor > 1 belong to cluster #3, regardless of the factor score on the “algorithmic” factor. Likewise, students with a factor score on the “structural” factor < 0, belong to cluster #1, regardless of the factor score on the “algorithmic” factor. For each cluster there is a similar range of variation for the factor scores on the “algorithmic” factor.

Figure 5. Plot of factor scores for all 22 students. The dashed lines separate the different clusters formed by hierarchical cluster analysis based on the Euclidian distances of the factor scores. The symbols representing the different students (solid circles, unfilled circles, crossed squares) are the same as in Figure 1.

It is instructive to compare Figure 5 to Figure 1. Note that, students belonging to cluster #1 in Figure 5 overlap considerably (7 out of 9, or 78%) with students with a poor performance in both
concepts and practical skills (solid circles in Figure 1). Also note that, students belonging to cluster #3 in Figure 5 overlap considerably with (5 out of 8, or 62%) with students with a satisfactory performance in both programming concepts and skills (unfilled circles in Figure 1). In comparing the grouping of students in Figures 5 (using cluster analysis of factor scores on concept assessments) and in Figure 1 (using performance in both concepts and skills assessments), it should be kept in mind, that the boundaries used in Figure 1 (70% performance, dashed lines) are somehow arbitrary. A different acceptable performance could have been chosen.

Overall, these results suggest that student clustering based on the factor scores on the “algorithmic” and “structural” factors, which are measurements based only on performance in conceptual assessments is a good predictor of student performance in practical programming skills. Since the clustering is mostly determined by the factor scores on the “structural” factor, we can conclude that student performance in the “structural” conceptual component is predictive of the student’s ability to solve practical computer programming problems. This is also consistent with the results in Figure 4 which indicate that students have more difficulty with concepts related with the structural factors of computer programming than with concepts related to algorithmic factors.

4. Discussion

Using factor analysis we have identified two factors or components grouping the interdependencies and correlations in student understanding of programming concept categories. The first component correlated with the repetition and selection categories, and could be referred to as the “algorithmic” component. The second component correlated with the methods, arrays and assignment categories, and could be referred as the “structural” component. Student performance in conceptual categories related to the “algorithmic” factor was significantly better than in conceptual categories related to the “structural” factor. Cluster analysis showed that student performance in the “structural” conceptual component is predictive of the student’s ability to solve practical computer programming problems.

Others before us have identified different factors that are important for student success in computer programming. Earlier work emphasizes the distinction between problem-solving skills and programming skills for student success as two different sets of cognitive skills, and suggest to teach problem solving before teaching actual programming. In our curriculum we use that approach, by requiring students to take a problem-solving course before they take their first programming course (see Methods/Participants). Still, to be able to write viable computer programs, students need both problem-solving skills and programming skills. In that context, it is possible that the “algorithmic” factor identified by factor analysis may relate to problem-solving skills. On the other hand, since cluster analysis indicates that the “structural” factor is predictive of students’ ability to write viable computer programs, it is also possible that the “structural” factor may relate to the programming skills proposed by others. Beyond the identification of both sets of conceptual categories, our results provide further insight on how different concept categories relate to the students’ ability to write viable computer programs.
Our students performed better in conceptual categories related to the “algorithmic” factor than in conceptual categories related to the “structural” factor (Figure 4). This may have been a consequence of the fact that before taking the Programming Fundamentals course (the subject of this study), our students are required to take a problem solving course that emphasizes algorithms to solve problems independently of a programming language. This is consistent with what others have shown that shifting from teaching programming to teaching problem-solving has been shown to increase success in first-year programming courses 4, 5, 9.

However, despite the benefits of an approach teaching problem-solving skills first, the transition from pre-programming problem-solving courses to courses in which students should master a full-fledge programming language remains a challenge 14, 18. This is reflected in the number of students (41%) who did not have an acceptable performance in both concepts and skills (Figure 1). Even though those students had passed a previous problem solving course, they find the transition to a learning environment that uses a full-fledge programming language like Java difficult. Our results show that a good understanding of concepts related to the “structural” factor may determine whether students would be able to write viable computer programs or not. So, additional emphasis should be placed in concepts related to program structure and organization to facilitate the students’ transition from concepts to practical skills.

According to Mayer 13, in addition to the cognitive and metacognitive aspects of problem solving, other aspects like motivation and engagement are also important determinants of student success in problem solving. We believe that student motivation and engagement is an important factor that contributes to the effectiveness of incorporating programming narratives in pre-programming problem solving courses 10, 11. Therefore, it is likely that pedagogical approaches that motivate and engage students will also facilitate their transition from concepts to practical skills in programming courses, with the concomitant effect on student success.

In interpreting our results, it is important to consider our student population, consisting mostly of underrepresented minorities (see Methods/Participants). Further studies in other institutions will need to be carried out to determine if our results apply to a different context.

5. Conclusions

We can conclude: 1) There are two barriers for student success in computer programming courses: a good understanding of programming concepts and the ability to apply those concepts to write viable computer programs; 2) Factor analysis shows that student understanding of computer programming concepts falls in two metaconceptual groups: an “algorithmic” and a “structural” factor; 3) Students have a better understanding of concepts that relate to the “algorithmic” factor than of concepts that relate to the “structural” factor; 4) Student performance in the “structural” conceptual component is predictive of the student’s ability to solve practical computer programming problems; 5) Strong emphasis in the structural components of computer programming (assignment, methods, arrays) is necessary for a successful transition from concepts to skills in computer programming courses.
6. References


